



- (51) **International Patent Classification:**
G06F 21/50 (2013.01)
- (21) **International Application Number:**
PCT/US2015/027595
- (22) **International Filing Date:**
24 April 2015 (24.04.2015)
- (25) **Filing Language:**
English
- (26) **Publication Language:**
English
- (30) **Priority Data:**
14/320,520 30 June 2014 (30.06.2014) US
- (63) **Related by continuation (CON) or continuation-in-part (CIP) to earlier application:**
US 14/320,520 (CON)
Filed on 30 June 2014 (30.06.2014)
- (71) **Applicant: EBAY INC.** [US/US]; 2145 Hamilton Avenue, San Jose, California 95125 (US).
- (72) **Inventors: WARDMAN, Bradley**; c/o EBAY INC., 2145 Hamilton Avenue, San Jose, California 95125 (US). **CARRICO, Ryan A.**; c/o EBAY INC., 2145 Hamilton Avenue, San Jose, California 95125 (US). **PRATT, Nate Robert**; c/o EBAY INC., 2145 Hamilton Avenue, San Jose, California 95125 (US).
- (74) **Agent: CHEN, Tom**; Haynes And Boone LLP, 2323 Victory Avenue, Suite 700, Dallas, Texas 75219-7673 (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.

[Continued on next page]

(54) **Title:** CLASSIFYING A PROGRAM INTERACTING WITH A SYSTEM USING QUESTIONING AND FINGERPRINTING

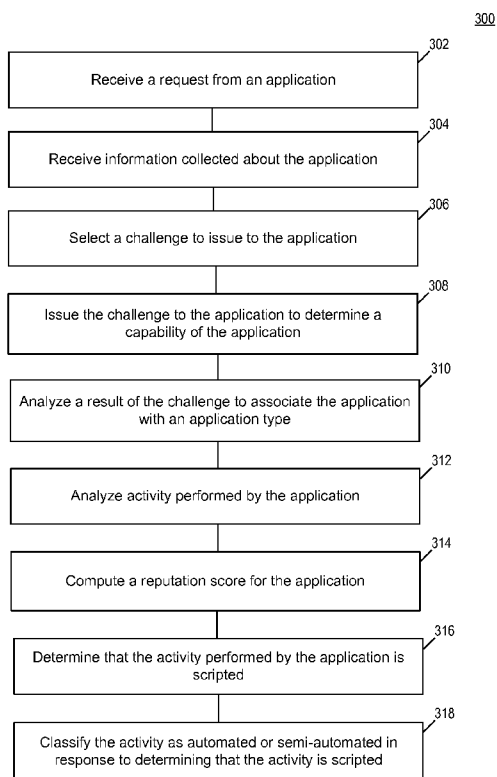


FIG. 3

(57) **Abstract:** Methods, systems, and computer program products for online content referral are provided. A computer-implemented method may include issuing a challenge to an application interacting with a computer system to determine whether activity performed by the application is scripted, analyzing data associated with a result of the challenge to determine whether the result is at least partially correct, and computing a reputation score for the application based on the result of the challenge. A computer-implemented method also may include comparing the reputation score for the application to a threshold for identifying non-scripted activity, and determining that the activity performed by the application is scripted when the reputation score for the application does not meet the threshold for identifying non-scripted activity.



(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).

Published:

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

CLASSIFYING A PROGRAM INTERACTING WITH A SYSTEM USING QUESTIONING AND FINGERPRINTING

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application is a continuation of and claims priority to U.S. Patent Application No. 14/320,520, filed June 30, 2014, which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention generally relates to computer system security and, more particularly, to the detection and classification of scripted activity in computer systems.

BACKGROUND

[0003] Cybercrime is criminal activity involving computer systems. For example, a computer system may be used to perpetrate a crime or may be a target of criminal activity. Such criminal activity may include unauthorized access to personal information, credit card accounts, bank accounts, and other secure data. In addition, cybercrime also may include the propagation of computer viruses and fraud using computer systems.

[0004] A “script” generally refers to a collection of executable computer language instructions stored in a file. Scripts usually are created to automate execution of the instructions by a computer system. Automated scripts are executed much faster and in greater number than what can be accomplished using human effort. As a result, criminals routinely use automated scripts in cybercrime to minimize effort, to increase overall numbers of targets and victims, and to carry out sophisticated attacks through bypassing known security measures.

[0005] Accordingly, improved methods of detecting and classifying automated scripts can aid in the identification, prevention and mitigation of cyberattacks.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Various examples of the present disclosure will be understood more fully from the detailed description given below and from the accompanying drawings of various examples of the disclosure. In the drawings, like reference numbers may indicate identical or functionally similar elements. The drawing in which an element first appears is generally indicated by the left-most digit in the corresponding reference number.

[0007] **FIG. 1** is a block diagram illustrating a system architecture, in accordance with various examples of the present disclosure.

[0008] **FIG. 2** is a flow diagram illustrating detection of scripted activity, according to an example of the present disclosure.

[0009] **FIG. 3** is a flow diagram illustrating multifactor detection of scripted activity, according to an example of the present disclosure.

[00010] **FIG. 4** is a block diagram of an exemplary computer system that may perform one or more of the operations described herein.

DETAILED DESCRIPTION

[00011] Systems, methods, and computer program products for accurately classifying a computer program interacting with a computer system using questioning and fingerprinting are disclosed. Examples of the present disclosure use questioning and fingerprinting to categorize applications interacting with a computer system as human operated, automated scripts, or driven by a combination of human interaction and automated scripts.

[00012] In an example, an application running on a client device interacts with a web server providing access to a web-based application. The web server receives a request from the client application (e.g., web browser, computer program, computer script, etc.) and a script detection system issues one or more challenges, either in response or as part of a response, to test the client application. For example, challenges may test capabilities of the client application to determine an application type of the client application. In one example, challenges test whether the client application is a web browser or another type of computer program imitating a web browser, regardless of how the client application has identified or represented itself.

[00013] In an example, the script detection system analyzes results of challenges issued to the client application to associate the application with an application type, analyzes fingerprint information collected about the client application, and analyzes monitored activity performed by the client application. The script detection system also computes a reputation score based on one or more of these factors, which indicates trustworthiness of the client application.

[00014] In an example, the script detection system determines whether activity of the client application is scripted or manually performed based on one or more of challenge results, information collected about the application, activity performed by the application, or the reputation score of the application. The script detection system also may classify scripted activity as automated or semi-automated based on one or more of these factors. Further, one or more actions may be triggered in response to the detection and classification of scripted activity based on a security level associated with a computer system.

[00015] Accordingly, aspects of the present disclosure allow detection of scripted activity and classification of such activity as automated, semi-automated, or manual using questioning and fingerprinting. Such detection and classification may be used to prevent and mitigate unauthorized computer system activity.

[00016] **FIG. 1** illustrates an exemplary system architecture 100 in which examples of the present disclosure can be implemented. System architecture 100 includes server machine

110, a data store 180 and client machines 102A-102N connected to a network 104. Network 104 may be a public network (e.g., the Internet), a private network (e.g., a local area network (LAN) or wide area network (WAN)), or a combination thereof. In an example, network 104 may include the Internet and/or one or more intranets, landline networks, wireless networks, and/or other appropriate types of communication networks. In one example, the network 104 may comprise a wireless telecommunications network (e.g., cellular phone network) adapted to communicate with other communication networks, such as the Internet.

[00017] Data store 180 is persistent storage that is capable of storing various types of data, such as text, audio, video and image content. In some examples data store 180 might be a network-attached file server, while in other examples data store 180 might be some other type of persistent storage such as an object-oriented database, a relational database, and so forth.

[00018] Client machines 102A-102N may be personal computers (PC), laptops, mobile phones, tablet computers, or any other computing device. Client machines 102A-102N may run an operating system (OS) that manages hardware and software of the client machines 102A-102N. A browser (not shown) may run on the client machines (e.g., on the OS of the client machines). The browser may be a web browser that can access content and services provided by a web server 120 of server machine 110. Other types of computer programs and computer scripts also may run on client machines 102A-102N.

[00019] Server machine 110 may be a rackmount server, a router computer, a personal computer, a portable digital assistant, a mobile phone, a laptop computer, a tablet computer, a camera, a video camera, a netbook, a desktop computer, a media center, or any combination thereof. Server machine 110 may include a web server 120 and a script detection system 130. In some examples, web server 120 and script detection system 130 may run on one or more different machines.

[00020] Web server 120 may serve text, audio, video and image content from data store 180 to clients 102A-102N. Web server 120 also may provide web-based application services and business logic to clients 102A-102N. Clients 102A-102N may locate, access and consume various forms of content and services from web server 120 using applications, such as a web browser. Web server 120 also may receive text, audio, video and image content from clients 102A-102N that is saved in data store 180 for purposes that may include preservation and distribution of content.

[00021] In an example, web server 120 is coupled to one or more applications servers (not shown) that provide applications and services to clients 102A-102N, for example, through web server 120. For example, web server 120 may provide clients 102A-102N with access to

one or more applications, which include but are not limited to banking, electronic commerce, e-mail, social networking, etc. Such functionality also may be provided, for example, as one or more different web applications, standalone applications, systems, plugins, web browser extensions, and application programming interfaces (APIs). In some examples, plugins and extensions may be referred to, individually or collectively, as add-ons.

[00022] In an example, some clients 102A-102N may include applications associated with a service provided by server 110. In one example, one or more device types (e.g., smart phones, smart televisions, tablet computers, etc.) may use applications to access content provided by server 110, to issue commands to server 110, to receive content from server 110 without visiting or using web pages.

[00023] In an example, functions performed by server 110 and/or web server 120 also may be performed by the client machines 102A-102N, in whole or in part. In addition, the functionality attributed to a particular component may be performed by different or multiple components operating together. Server 110 also may be accessed as a service provided to other systems or devices appropriate application programming interfaces, and thus is not limited to use in websites.

[00024] In an example, script detection system 130 includes application verifier module 140, application fingerprinting module 150, activity analyzer module 160, and activity classifier module 170. In other examples, functionality associated with application verifier module 140, application fingerprinting module 150, activity analyzer module 160, and activity classifier module 170 may be combined, divided and organized in various arrangements.

[00025] In an example, application verifier module 140 receives one or more user requests from an application running on a client 102A. In response, application verifier module 140 issues a challenge to the application to determine whether or not an application is a particular application type. For example, application verifier module 140 may challenge the application to perform one or more tasks or to answer one or more questions (e.g., in a process of elimination). Application verifier module 140 then may analyze one or more results of the challenges to determine whether or not the application is a web browser, an automated script, a shell browser automated by a script, or another type of application. Thus, application verifier module 140 may determine an application type of an application based on whether the application is able or unable to perform certain tasks or answer certain questions in a manner consistent with an application type.

[00026] In an example, application verifier module 140 adaptively selects a challenge to present to the application from a collection of one or more available challenges. In one example, application verifier module 140 selects challenges randomly. Application verifier module 140 also may select challenges adaptively or by using a process of elimination to determine whether an application is associated with a particular application type.

[00027] In an example, application verifier module 140 issues one or more challenges to an application at a particular time or at various times as a client 102A application interacts with server machine 110. For example, application verifier module 140 may issue one or more challenges as part of authentication and authorization of a user login. In one example, application verifier module 140 issues challenges to an application at various points throughout a session. For example, application verifier module 140 may issue a challenge to a client 102A application at random times or when the application seeks to perform one or more predetermined activities during a session.

[00028] In an example, application fingerprinting module 150 receives information about the application. For example, application fingerprinting module 150 may actively collect information from the application or may receive such information from another source that has gathered or received the data.

[00029] In an example, application fingerprinting module 150 receives information associated with a client 102A application from one or more sources. For example, information associated with an application may be collected from cookies, supercookies, user agents, browser fingerprints, flash shared objects, network devices, etc. Such sources may provide correct or altered information about what an application is, what an application can do, a computing device executing the application, an environment where the application that executes, a location of the computing device where the application executes, etc. In one example, application fingerprinting module 150 analyzes information associated with a client 102A application from various sources to detect consistency and inconsistency as part of identifying and classifying scripted activity.

[00030] In an example, activity analyzer module 160 monitors and analyzes activity performed by a client 102a application that interacts with a server machine 110, web server 120, or other computing system. In one example, activity analyzer module 160 analyzes timing of application requests to determine the likelihood of whether the activity is associated with an automated script. For example, activity analyzer module 160 may compare timing of application requests to baseline timing of human responses to determine whether the application requests are associated with scripted activity.

[00031] In an example, activity analyzer module 160 monitors and analyzes web pages and other objects accessed by a client 102A application, navigation of a client 102A application through a web site or other type of target application, repetition in the behavior of a client 102A application, whether a client 102A application accesses multiple accounts, and other various aspects of interaction between a client 102A application and a target computer system.

[00032] In an example, activity classifier module 170 calculates a reputation score or rating for a client 102A application interacting with a server machine 110 or a web server 120. A reputation score or rating generally refers to a score, classification or ranking, which indicates trustworthiness of an application. In one example, activity classifier module 170 computes a reputation score for a client 102A application based on information provided by one or more of application verifier module 140, application fingerprinting module 150, and activity analyzer module 160. For example, a reputation score may be based in part on comparing what a client 102A application says it is to what script detection system 130 determines the client 102A application to be.

[00033] In an example, a client 102A application may indicate that it is a web browser via a user agent. However, challenges that issued to the client 102A application may reveal that the application behaves like a PERL[®] script. Therefore, such a discrepancy generally may reduce the reputation score or rating of the application. Further, activity performed by a client 102A application at a much faster rate than human-performed activity also may lower the reputation score rating of the application. In contrast, the reputation score for a client 102A application may increase, for example, incrementally when the application behaves in a manner matching an application type expected to be interacting with a computer system (e.g., a web browser), when the application behaves in a manner consistent with how it has identified itself, or when the application appears to be operated by human rather than by an automated script.

[00034] In an example, activity classifier module 170 determines whether activity performed by a client 102A application is scripted and classifies the activity as automated, semi-automated, or manually performed. In one example, activity classifier module 170 determines that client 102A application activity is scripted based on any single determination or finding provided by application verifier module 140, application fingerprinting module 150, or application analyzer module 160. Activity classifier module 170 also may use the same information or other information to classify the activity as one of automated, semi-automated or manually operated and controlled.

[00035] In an example, activity classifier module 170 also may determine that client 102A application activity is scripted based on a combination of information or determinations provided by any two or more of the application verifier module 140, application fingerprinting module 150, and activity analyzer module 160. Activity classifier module 170 also may use the same information or other information to classify the activity as one of automated, semi-automated or manually performed.

[00036] In an example, activity classifier module 170 determines that a client 102A application is associated with scripted activity when a reputation score for the application does not meet a predetermined threshold of trustworthiness for identifying non-scripted activity. In one example, activity classifier module 170 classifies client 102A application activity determined to be scripted as automated or semi-automated based on a computed reputation score for the application or information provided by any of application verifier module 140, application fingerprinting module 150, and activity analyzer module 160.

[00037] FIG. 2 is a flow diagram illustrating detection of scripted activity, according to an example of the present disclosure. The method 200 may be performed by processing logic that may comprise hardware (circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a general purpose computer system, dedicated machine, or processing device), firmware, or a combination thereof.

[00038] Method 200 begins at block 202 when script detection system 130 issues a challenge to a client 102A application to determine a capability of the application. In one example, application verifier module 140 selects a challenge to issue to a client 102A application interacting with server machine 110 from one or more available challenges or different collections of challenges designed to test capabilities of various application types. For example, application verifier module 140 may use challenges that are each configured to test capabilities of application types, such as web browsers, Java[®] applications, PERL[®] scripts, etc.

[00039] In an example, a web browser is an accepted or approved application type expected to interact with a web application provided by web server 120. Application verifier module 140 selects one or more challenges that it will issue to a client 102A application making requests to web server 120. For example, application verifier module 140 may seek to determine whether a client 102A application interacting with web server 120 is a web browser, a particular type of web browser, or another application type.

[00040] In an example, application verifier module 140 issues a challenge written in the JavaScript[®] programming language to a client 102A application that it expects to be a web

browser. For example, JavaScript[®] code may be sent to a client 102A application as part of a response to a request. In one example, web browser challenges may include asking a web browser to provide a list of installed plug-ins, to provide a list of available fonts, to perform a mathematical calculation, or to obtain a resource (e.g., a file) from a particular location. In one example, application verifier module 140 may select and issue a series of challenges randomly and/or adaptively to prove or disprove that an application is associated with an approved or non-approved application type.

[00041] In an example, each challenge may be blocking or non-blocking to client 102A application interaction with a server machine 110, web server 120, application server, software application, etc. For example, a blocking challenge may require a client 102A application to first provide a response to an issued challenge before the application is permitted to continue interacting with a computer system. In an example, a non-blocking challenge allows a client 102A application to continue interacting with a computer system even when the application does not provide a response to the challenge. In one example, a reputation score of a client 102A application is reduced or lowered when the application does not provide a response to an issued challenge.

[00042] In an example, a non-blocking challenge allows a client 102A application to continue interacting with a computer system for a limited period. For example, a client 102A application may be permitted to continue interacting with a computer system for a limited period of time or for a limited number of requests before a response to an issued challenge is required. In one example, a client 102A application session is disallowed or terminated when the application fails to respond to one or more challenges.

[00043] At block 204, the script detection system 130 analyzes a result of the challenge to associate a client 102A application with an application type. In an example, application verifier module 140 analyzes a result returned from a client 102A application in response to an issued challenge. In one example, application verifier module 140 compares a result of the challenge to an expected result for an application type tested by the challenge. For example, application verifier module 140 may issue a challenge designed to test whether a client 102A application is a web browser. A web browser specific challenge may inquire about a specific web browser details that are not available in other application types. Thus, application types other than web browsers generally would not have the capability to answer web browser specific challenges.

[00044] In an example, application verifier module 140 associates a client 102A application with an application type based on its analysis of one or more challenge results. In

one example, application verifier module 140 determines a probability that an application is associated with an application type based on results of one or more challenges. For example, application verifier module 140 may determine a probability that a client 102A application is associated with one or more different application types based on each challenge response provided by a client 102A application. Application verifier module 140 then may conclude that the application is associated with one or more application types, for example, after a certain number of challenges have been issued, based on a numeric threshold, or when a statistical level of certainty has been met.

[00045] At block 206, script detection system 130 determines whether activity performed by a client 102A application is scripted. In an example, script detection system 130 determines that activity performed by a client 102A application is scripted when the application is unable to correctly answer or perform one or more challenges or a certain number or percentage of challenges. In one example, script detection system 130 also may determine that activity performed by a client 102A application is scripted when the application correctly answers or performs one or more challenges associated with a non-approved application type for interacting with server machine 110 or web server 120.

[00046] In an example, script detection system 130 determines that activity performed by a client 102A application is scripted based on analyzing the activity. In one example, activity analyzer module 150 analyzes a speed at which the activity is performed, objects and functionality that are accessed, navigation performed by the application activity, repetition in behavior of the application activity, patterns in application activity network traffic, patterns in application activity data, etc.

[00047] In an example, activity classifier module 170 determines that activity performed by a client 102A application is scripted when it detects that the activity is performed at a faster rate than human initiated activity associated with a similar computing device, when the activity accesses certain objects or functionality, when it detects the activity is performing anomalous website navigation, when it detects a repetitive pattern in the activity, or when it detects anomalous network traffic patterns or alphanumeric character strings associated with scripted activity.

[00048] In one example, activity classifier module 170 determines that activity performed by a client 102A application is scripted when information collected about the application by application fingerprinting module 150 is inconsistent with information determined about the application by application verifier module 140 based on results of one of more challenges. For example, information about the application may be collected from cookies, supercookies,

user agents, browser fingerprints, flash shared objects, network devices, etc. In one example, a client 102A application is determined to be scripted when an application type found and information collected about the application does not match an application type that has been determined for a client 102A.

[00049] At block 208, script detection system 130 classifies the activity as automated or semi-automated when it is determined that the activity is scripted. In an example, activity classifier module 170 classifies the activity of a client 102A application determined to be scripted as automated or semi-automated activity. In one example, automated activity generally refers to application activity that is determined to be performed by a computer system. Semi-automated activity generally refers to application activity that is determined to be performed by a combination of a computer system and human effort.

[00050] In an example, activity classifier module 170 classifies activity of a client 102A application as automated. For example, activity classifier module 170 may classify a client 102A application as automated when detecting that an application type determined to be associated with the application is an unapproved application type (i.e., not expected to interact with server machine 110 or web server 120). In one example, a web browser application may be an approved application type expected to interact with web server 120. However, an application determined to be another application type (a Java[®] application, PERL[®] script, etc.) may be unapproved for interaction with web server 120.

[00051] In an example, activity classifier module 170 also may classify an application as automated when information collected about a client 102A application is inconsistent with an application type determined to be associated with the application. For example, an application may identify itself as a web browser, but the application actually may be determined to be a different application type based on a result of one or more challenges. In one example, activity classifier module 170 also may classify an application as automated even when information collected about a client 102A application is consistent with a determined application type. For example, activity classifier module 170 may classify application activity as automated based on analysis of client 102A application activity or a corresponding determination provided by activity analyzer module 160.

[00052] In an example, activity classifier module 170 classifies activity of a client 102A application determined to be scripted as semi-automated. In one example, activity classifier module 170 classifies application activity as semi-automated when analysis of the activity or a corresponding determination provided by activity analyzer module 160 indicates that the activity is performed by some combination of computer system and human effort. For

example, a pattern or timing of activity performed by an application during a login process on server machine 110 or web server 120 may indicate that the activity is human driven.

However, a later pattern or timing of activity performed during an authenticated session resulting from the login process may indicate that another portion of the activity is performed by a computer system (e.g., via execution of automated scripts).

[00053] In an example, one or more actions are taken when scripted activity is detected and/or classified as automated or semi-automated. In one example, responsive actions are performed based a level of sensitivity or confidentiality associated with an application or associated information. Thus, actions taken in response to detecting automated or semi-automated scripts may be tailored flexibly to any organization, application, situation, etc.

[00054] In an example, detection and classification of scripted activity triggers generation of a corresponding logged event with associated information. An alert also may be generated and transmitted to notify one or more parties of the event. In one example, a security permission of a session involving the application determined to be performing scripted activity is adjusted. For example, security permissions for a user may be modified from writeable to read-only to prevent data modification. User access also may be restricted to a limited set of objects or functionality. In some examples, sessions are terminated, user accounts are disabled and access is blocked to a source of a client 102A application.

[00055] **FIG. 3** is a flow diagram illustrating multifactor detection of scripted activity, according to an example of the present disclosure. The method 300 may be performed by processing logic that may comprise hardware (circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a general purpose computer system, dedicated machine, or processing device), firmware, or a combination thereof.

[00056] Method 300 begins at block 302 when script detection system 130 receives a request from a client 102A application. In an example, an application running on a client 102A interacts with server machine 110 by sending a request to web server 120. For example, an application running on 102A may access a landing page, login page, or transactional page of a web application provided by web server 120. The web application may provide services, such as banking, electronic commerce, e-mail, social networking, e-mail, etc.

[00057] At block 304, script detection system 130 receives collected information about the client 102A application. In an example, application fingerprinting module 150 collects or receives collected information about a client 102A application that interacts with server machine 110 or web server 120. In one example, application fingerprinting module 150

collects or receives information from one or more of cookies, supercookies, user agents, flash shared objects, network diagnostics, etc. The collected information, for example, may indicate attributes of the application, aspects of an environment where the application executes, attributes of a computing device that executes the application, attributes of one of more network connections used by the application, etc.

[00058] At block 306, script detection system 130 selects a challenge to issue to the client 102A application. In an example, application verifier module 140 selects a challenge to issue to a client 102A application based on an approved or expected application type that is to interact with server machine 110 or web server 120. Application verifier module also may select a challenge to issue to the application based on information collected about the application received at block 304.

[00059] At block 308, script detection system 130 issues a challenge to the client 102A application to determine one or more capabilities of the application. In an example, application verifier module 140 issues the challenge selected at block 306 to the application to determine whether the application has the capability to perform the challenge or to provide information requested by the challenge.

[00060] In an example, application verifier module 140 issues a challenge by sending JavaScript[®] code to a client 102A application expected to be a web browser as part of a response to a request received from the application. In one example, application verifier module 140 issues the challenge to determine whether the application is a web browser by asking the application to perform an activity or to provide information that is inherent to a web browser. For example, a standard web browser usually can provide a list of installed plug-ins, provide a list of available fonts, perform a mathematical calculation, and obtain a resource (e.g., a file) from a particular location. Application verifier module 140 may select and issue a series of random or adaptive challenges to prove or disprove that an application is associated with an application type.

[00061] In an example, application verifier module 140 issues a challenge to a client 102A application asking the application to perform one or more activities using computer code provided by a computer system. For example, the computer system may provide one or more types of computer code for a client 102A application to use when responding to a challenge. In one example, a computer system provides computer code written in one or more custom or standardized computer languages (e.g., C, C++, C#, Java[®], JavaScript[®], PERL[®], extensible markup language (XML), hypertext markup language (HTML), cascading style sheets (CSS), and/or other computer languages) to an application.

[00062] In an example, application verifier module 140 issues a challenge involving a CAPTCHA (i.e., a Completely Automated Public Turing test to tell Computers and Humans Apart). For example, a challenge may include a CAPTCHA comprising text identification, numeric identification, audio identification, a task, a puzzle, a mathematical problem, image recognition, etc. Application verifier module 140 also may issue one or more similar non-CAPTCHA challenges.

[00063] In an example, application verifier module 140 issues a challenge involving word analysis. For example, a challenge may ask an application to provide a count of characters in a word or alphanumeric string. A challenge also may ask an application to provide a count of vowels or non-vowels in a word or string, one or more positional references of an identified character, numeral, or symbol in a word or string, etc. Further, a challenge may ask an application whether one or more words have been spelled correctly.

[00064] In an example, application verifier module 140 issues a challenge involving sentence analysis. For example, a challenge may ask an application to provide a count of words in a sentence. A challenge also may ask an application to identify whether a sentence is declarative, imperative, interrogatory, or exclamatory. In one example, a challenge may ask an application whether a sentence is properly formed, or to identify one or more languages represented in a sentence, etc.

[00065] In an example, application verifier module 140 issues a challenge involving one or more mathematical problems. For example, a challenge may ask an application to solve a mathematical problem that is numeric, an equation, or a mathematical problem described in words. In one example, a challenge may ask an application to provide a plurality of answers for a mathematical problem having at least two correct answers (e.g., correct positive and negative answers, etc.).

[00066] In an example, application verifier module 140 issues a challenge involving one or factual questions. For example, a challenge may ask an application to demonstrate awareness of or an ability to locate one or more pieces of common knowledge. For example, a challenge may ask an application to provide an answer to a question pertaining to science, history, geography, literature, entertainment, etc. (e.g., the capital of France, the tenth U.S. president, etc.).

[00067] At block 310, script detection system 130 analyzes a result of the challenge to associate the client 102A application with an application type. In an example, application verifier module 140 compares a response to a challenge received from a client 102A application to an expected result associated with an application type. For example,

application verifier module 140 may analyze whether a result partially or fully matches expected result associated with an approved application type or an unapproved application type to determine whether the application is or is not associated with an application type.

[00068] In an example, application verifier module 140 may issue a series of challenges to a client 102A application, for example, randomly or adaptively. In one example, application verifier module 140 determines a probability that the application is associated with each of one or more application types. For example, application verifier module 140 may compute and re-compute a probability that the application is associated with each of multiple application types based on results provided by a client 102A application in response to multiple challenges.

[00069] At block 312, script detection system 130 analyzes activity performed by a client 102A application. In an example, activity analyzer module 160 monitors activity of a client 102A application interacting with web server 120 of server machine 110. In one example, activity analyzer module 160 may analyze timing (e.g., speed, latency, cadence, etc.) of client 102A application requests and responses to determine whether the activity is initiated by a human, automated script, or semi-automated script. For example, activity analyzer module 160 may compare timing of client 102A application activity to a preexisting timing baseline of similar activity as performed by a human to determine whether the application requests are associated with the scripted activity. In one example, activity analyzer module 160 analyzes website navigation performed by the activity, repetition in the behavior of the activity, whether the activity accesses or attempts to access multiple accounts, etc. In some examples, a reputation score of a client 102A application is adjusted (e.g., positively or negatively) based on one or more results of script detection system 130 analysis of client 102A application activity.

[00070] At block 314 script detection system 130 computes a reputation score for the client 102A application. In an example, activity classifier module 170 computes a reputation score or rating indicating trustworthiness of a client 102A application interacting with web server 120 of server machine 110. In one example, activity classifier module 170 computes a reputation score for an application based on a correlation between an application type that is determined to be associated with the application and information collected about the application. For example, the more closely a client 102a application behaves to how it identifies itself may result in a higher reputation score.

[00071] In an example, a reputation score for a client application also may be influenced positively or negatively based on behavior of the application. For example, application

activity associated with anomalous timing or behavioral patterns may result in a reduction in reputation score where non-suspicious activity may result in a consistent or improved reputation score. In one example, a reputation score may be initially computed and updated any number of times as new information is observed or learned about a client 102A application.

[00072] In an example, a reputation score is computed or adjusted based on one or more fonts associated with a client 102A application. In one example, application fingerprinting module 150 determines one or more fonts associated with a client 102A application, such as a web browser, based on application fingerprint information. Application verifier module 140 also may ask an application to provide one or more associated or installed fonts.

[00073] In an example, activity classifier module 170 computes or adjusts an application's reputation score based on comparing one or more fonts installed on or associated with the application to one or more fonts expected to be installed on or associated with the application. For example, an application that is missing one or more expected fonts may receive a reduction in reputation score. On the other hand, an application having one or more expected fonts may receive a reputation score increase.

[00074] In an example, a reputation score is computed or adjusted based on one or more add-ons associated with a client 102A application. For example, application fingerprinting module 150 may determine one or more add-ons installed on a web browser application based on fingerprint information. Application verifier module 140 also may ask an application to provide a list of one or more add-ons installed on or available to an application. In one example, a reputation score of a client application is increased when an expected add-on is identified as installed on the application. A reputation score also may be reduced or penalized when a client application is missing an add-on that is expected to be installed.

[00075] In an example, activity classifier module 170 computes or adjusts a reputation score of a client 102A application based on analysis of one or more cached objects associated with the application. For example, application verifier module 140 may ask an application to provide a list of one or more cached objects associated with the application. Application fingerprinting module 150 may determine a list of one or more cached objects associated with an application from application fingerprint information. In one example, activity classifier module 170 analyzes the one or more cached objects and adjusts a reputation score for an application based on whether one or more of the cached objects are determined to be legitimate (e.g., non-malicious) cached user-objects, or whether one or more of the cached objects are associated with a malicious source, malware, spyware, viruses, etc.

[00076] In an example, activity classifier module 170 computes or adjusts a reputation score of a client 102A application based on header information provided by the application. For example, activity classifier module 170 may analyze header information provided by a client 102A during standard communication, in response to an issued challenge, in fingerprint information, etc. In one example, activity classifier module 170 analyzes attributes, values, lengths, and other characteristics of the client transmitted headers.

[00077] In an example, activity classifier module 170 increases or decreases a client 102A application reputation score based on the correctness and authenticity of header information provided by the application. For example, activity classifier module 170 may increase a reputation score when header information provided from a web browser application is complete, correct, accurate, meets standards (including but not limited to RFC 2616), etc. On the other hand, activity classifier module may deduct value from a reputation score when header information is incomplete, incorrect, inaccurate, contains a string of characters or behaviors indicative of scripted activity, etc.

[00078] In an example, activity classifier module 170 increases a client 102A application reputation score when header information provided by the application (e.g., via self-identification) matches header information received when communicating with the application. For example, when a client 102A application identifies itself as a web browser (e.g., a FireFox[®] web browser, a Safari[®] web browser, etc.) of a particular version or release, then certain aspects of header information received from that application usually match header information associated with the particular version or release of that web browser. In one example, activity classifier module 170 decreases a client 102A application reputation score when header information received from the application does not match header information associated with a self-identified application type provided by the application. For example, if a client 102A application self-identifies as a web browser and header information received from the application matches a programming language library (instead of web browser header information), then it is more likely that the application is scripted, and a reputation score of the application may be reduced accordingly.

[00079] At block 316, script detection system 130 determines that the activity performed by the client 102A application is scripted. In an example, application verifier module 140 determines that activity performed by a client 102A application is scripted when a client 102A application is unable to correctly respond to a challenge associated with an approved application type for interacting with server machine 110 or web server 120. In one example, application fingerprinting module 150 determines that activity performed by a client 102A

application is scripted based on information collected about the application. In one example, activity analyzer module 160 determines that activity performed by a client 102A application is scripted based on a pattern of anomalous timing or behavior that it detects when analyzing the activity.

[00080] In an example, activity classifier module 170 determines that application activity is scripted based on a reputation score computed for the application. In one example, activity classifier module 170 determines that a client 102A application interacting with web server 120 is scripted when a reputation score computed for the application does not meet a predetermined threshold indicating that the application is trustworthy. In one example, classifier module 170 compares a reputation score computed for an application to a predetermined minimum threshold of trustworthiness each time the reputation score is updated based on new information.

[00081] At block 318, script detection system 130 classifies the activity as automated or semi-automated in response to determining that the activity is scripted. In an example, activity classifier module 170 classifies activity of a client 102A application as automated or semi-automated activity. For example, activity classifier module 170 may classify an application as automated by determining that an application type of an application is an unapproved application type for interacting with web server 120. Activity classifier module 170 also may classify an application as automated when information collected about a client 102A application is inconsistent with an application type determined to be associated with the application. Activity classifier module 170 also may classify application activity as automated based on analysis of the activity or a corresponding determination provided by activity analyzer module 160.

[00082] In an example, activity classifier module 170 classifies activity of a client 102A application as semi-automated. In one example, activity classifier module 170 classifies activity as semi-automated when it is determined that the activity is performed by a combination of computer system and human effort. For example, application activity may be associated with human interaction at one point in a session and then may be associated automated execution by a computer system at another point in the session (or vice versa). In one example, activity classifier module 170 may classify activity as semi-automated based on a reputation score computed for the application.

[00083] In an example, one or more actions are taken when scripted activity is detected or identified as automated or semi-automated. In one example, responsive actions are performed based a level of sensitivity or confidentiality associated with an application or

associated information. Thus, actions taken in response to detecting automated or semi-automated scripts may be tailored flexibly to any organization, application, situation, etc.

[00084] In an example, one or more activities are performed in response to classifying application activity as automated or semi-automated. For example, an event may be logged, an alert may be transmitted, a security permission of a session or user may be modified, a session may be terminated, a user account may be disabled, access may be blocked for a client 102A application or for a source of the application, etc.

[00085] **FIG. 4** illustrates a diagram of a machine in the exemplary form of a computer system 400 within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In other examples, the machine may be connected (e.g., networked) to other machines in a LAN, an intranet, an extranet, or the Internet. The machine may operate in the capacity of a server or a client machine in client-server network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term “machine” shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[00086] The exemplary computer system 400 includes a processing device (processor) 402, a main memory 404 (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM), double data rate (DDR SDRAM), or DRAM (RDRAM), etc.), a static memory 406 (e.g., flash memory, static random access memory (SRAM), etc.), and a data storage device 418, which communicate with each other via a bus 430.

[00087] Processor 402 represents one or more general-purpose processing devices such as a microprocessor, central processing unit, or the like. More particularly, the processor 402 may be a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or a processor implementing other instruction sets or processors implementing a combination of instruction sets. The processor 402 may also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. The processor

402 is configured to execute instructions 422 for performing the operations and steps discussed herein.

[00088] The computer system 400 may further include a network interface device 408. The computer system 400 also may include a video display unit 410 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)), an alphanumeric input device 412 (e.g., a keyboard), a cursor control device 414 (e.g., a mouse), and a signal generation device 416 (e.g., a speaker).

[00089] The data storage device 418 may include a computer-readable storage medium 428 on which is stored one or more sets of instructions 422 (e.g., software) embodying any one or more of the methodologies or functions described herein. The instructions 422 may also reside, completely or at least partially, within the main memory 404 and/or within the processor 402 during execution thereof by the computer system 400, the main memory 404 and the processor 402 also constituting computer-readable storage media. The instructions 422 may further be transmitted or received over a network 420 via the network interface device 408.

[00090] In one example, the instructions 422 include instructions for a script detection system (e.g., script detection system 130 of **FIG. 1**) and/or a software library containing methods that call a script detection system. While the computer-readable storage medium 428 (machine-readable storage medium) is shown in an example to be a single medium, the term “computer-readable storage medium” should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term “computer-readable storage medium” shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present disclosure. The term “computer-readable storage medium” shall accordingly be taken to include, but not be limited to, solid-state memories, optical media, and magnetic media.

[00091] In the foregoing description, numerous details are set forth. It will be apparent, however, to one of ordinary skill in the art having the benefit of this disclosure, that the present disclosure may be practiced without these specific details. In some instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present disclosure.

[00092] Some portions of the detailed description have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer

memory. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[00093] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "computing", "comparing", "applying", "creating", "ranking," "classifying," or the like, refer to the actions and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (e.g., electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[00094] Certain examples of the present disclosure also relate to an apparatus for performing the operations herein. This apparatus may be constructed for the intended purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions.

[00095] It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other examples will be apparent to those of skill in the art upon reading and understanding the above description. The scope of the disclosure should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

CLAIMS

What is claimed is:

1. A system, comprising:
a non-transitory memory; and
one or more hardware processors in communication with the memory and configured to read instructions from the memory to:
issue a challenge to an application interacting with a computer system to determine whether activity performed by the application is scripted;
analyze data associated with a result of the challenge to determine whether the result is at least partially correct; and
compute a reputation score for the application based on the result of the challenge.
2. The system of claim 1, wherein the one or more hardware processors further:
compare the reputation score for the application to a threshold for identifying non-scripted activity.
3. The system of claim 1, wherein the one or more hardware processors further:
determine that the activity performed by the application is scripted when the reputation score for the application does not meet a minimum threshold.
4. The system of claim 1, wherein the challenge involves the application performing an activity using computer code provided by the computer system.
5. The system of claim 4, wherein the computer code comprises JavaScript[®] computer language code.
6. The system of claim 4, wherein the computer code comprises style sheet computer language code.
7. The system of claim 1, wherein the challenge involves a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA).
8. The system of claim 1, wherein the challenge involves performing analysis of one or more words.

9. The system of claim 1, wherein the challenge involves performing analysis of one or more sentences.

10. The system of claim 1, wherein the challenge involves solving one or more mathematical problems.

11. The system of claim 1, wherein the challenge involves solving a mathematical problem with at least two answers.

12. The system of claim 1, wherein the challenge involves demonstrating awareness of a fact.

13. The system of claim 1, wherein the one or more hardware processors further: determine one or more fonts associated with the application; and adjust the reputation score in view of the one or more fonts associated with the application.

14. The system of claim 1, wherein the one or more hardware processors further: determine one or more add-ons associated with the application; and adjust the reputation score in view of the one or more add-ons associated with the application.

15. The system of claim 1, wherein the one or more hardware processors further: analyze one or more cached objects associated with the application; and adjust the reputation score in view of analyzing the cached objects.

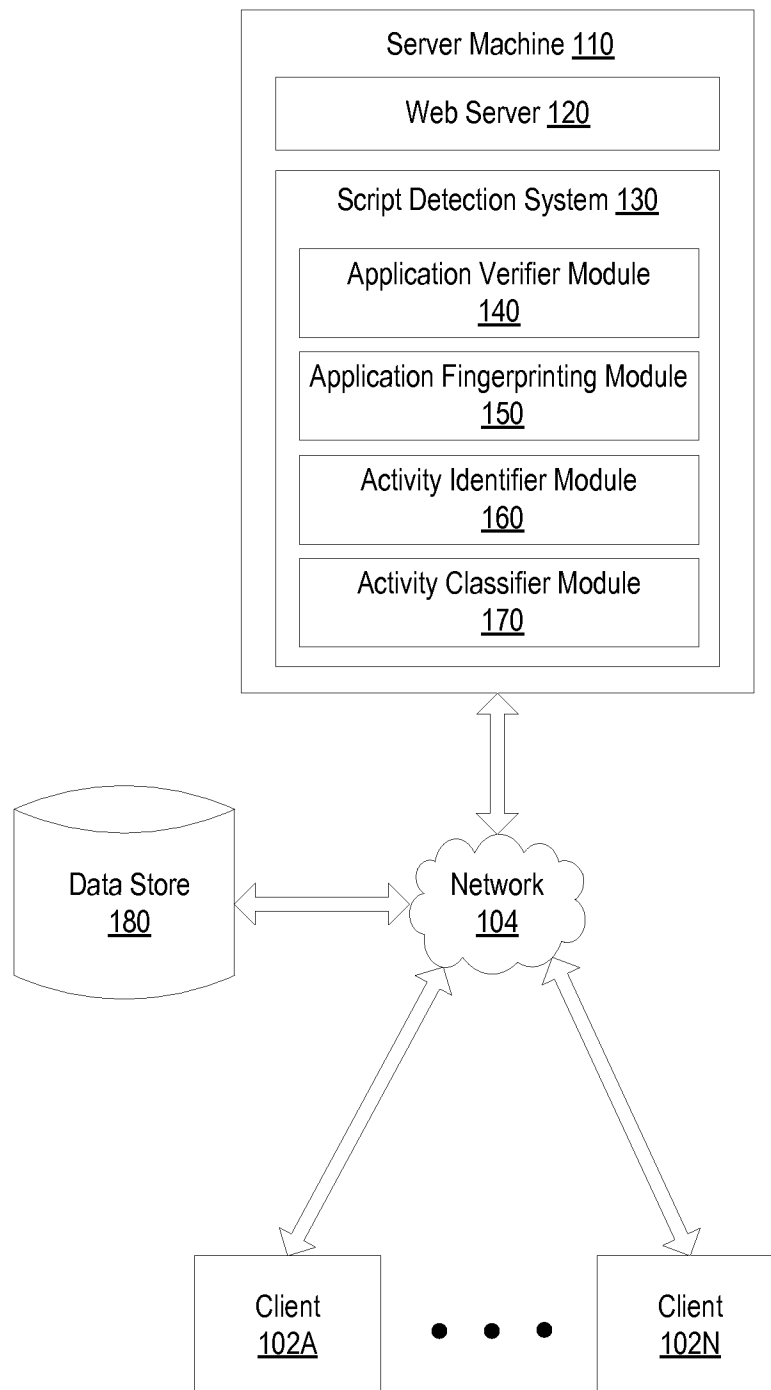
16. The system of claim 1, wherein the one or more hardware processors further: analyze speed for each of one or more responses the application provides to the computer system; and adjust the reputation score in view of analyzing the speed of the one or more responses provided by the application.

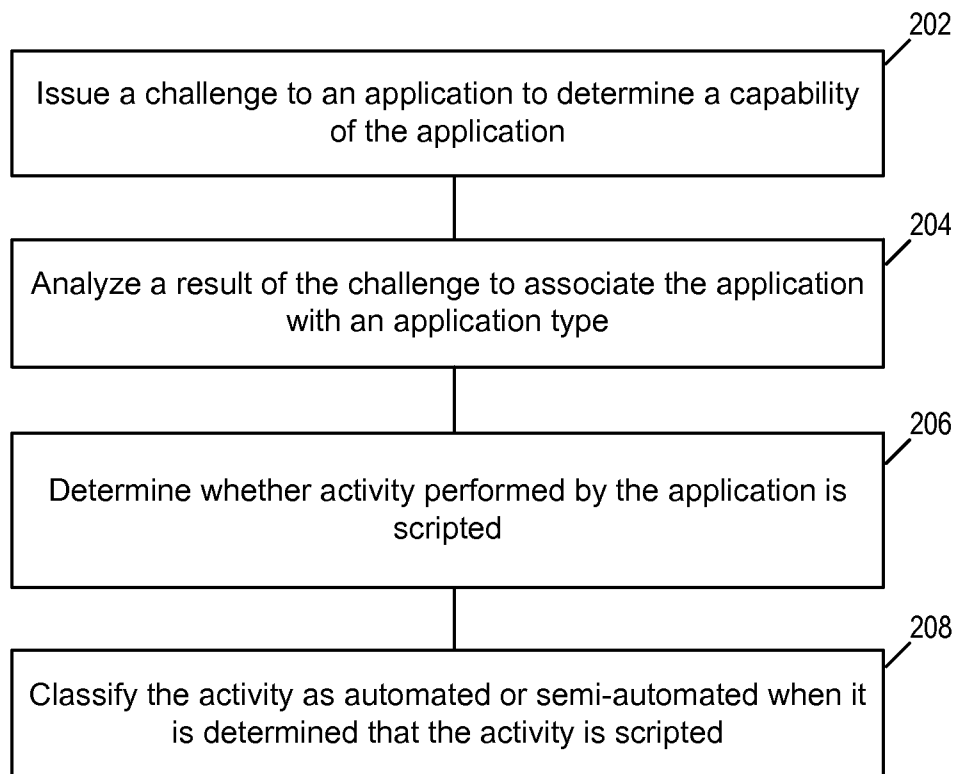
17. The system of claim 16, wherein the speed of one or more of the responses is compared to a preexisting baseline of related computer system activity performed by a human.

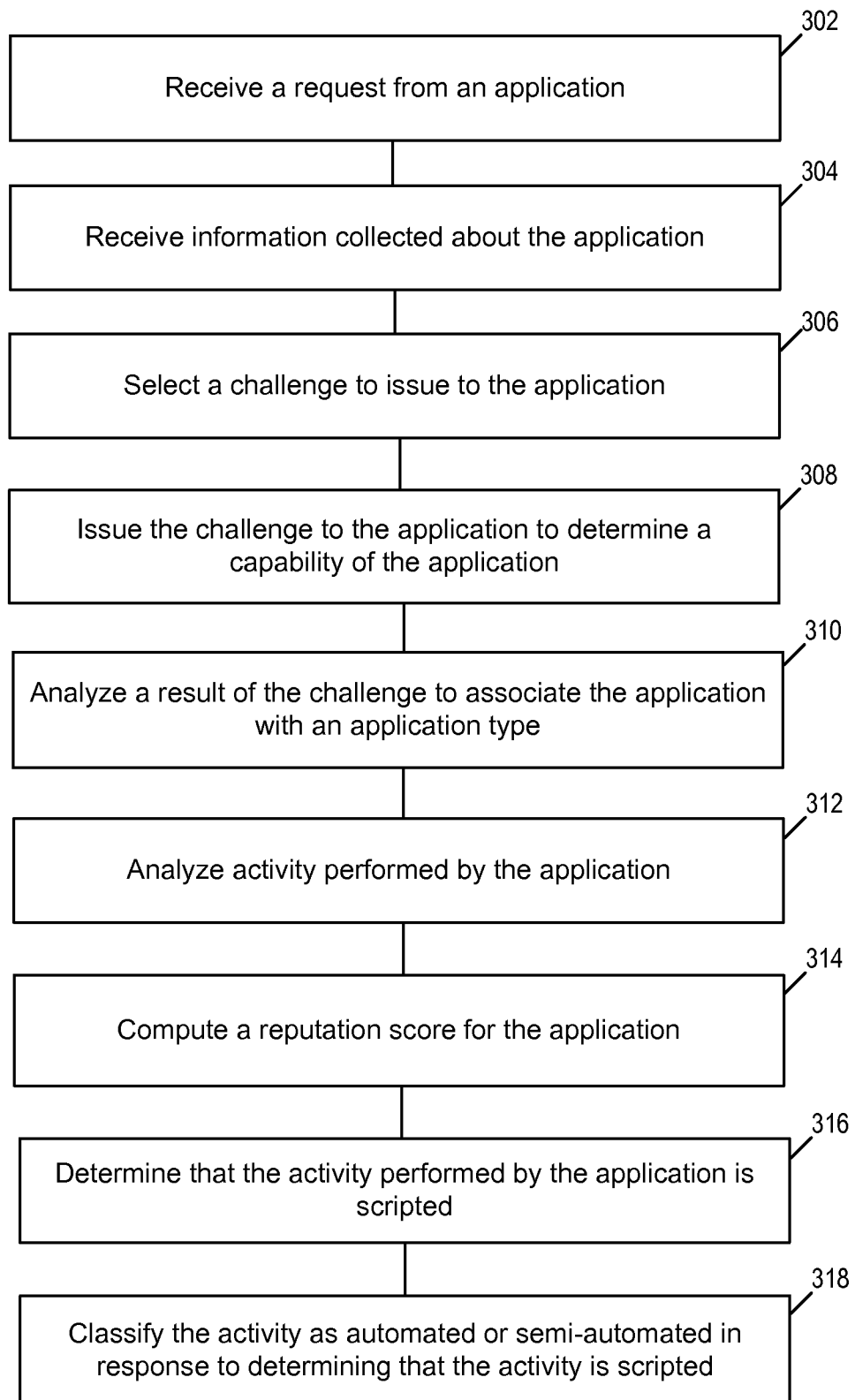
18. The system of claim 1, wherein the one or more hardware processors further:
analyze header information provided by the application; and
adjust the reputation score in view of analyzing the header information provided by the application.

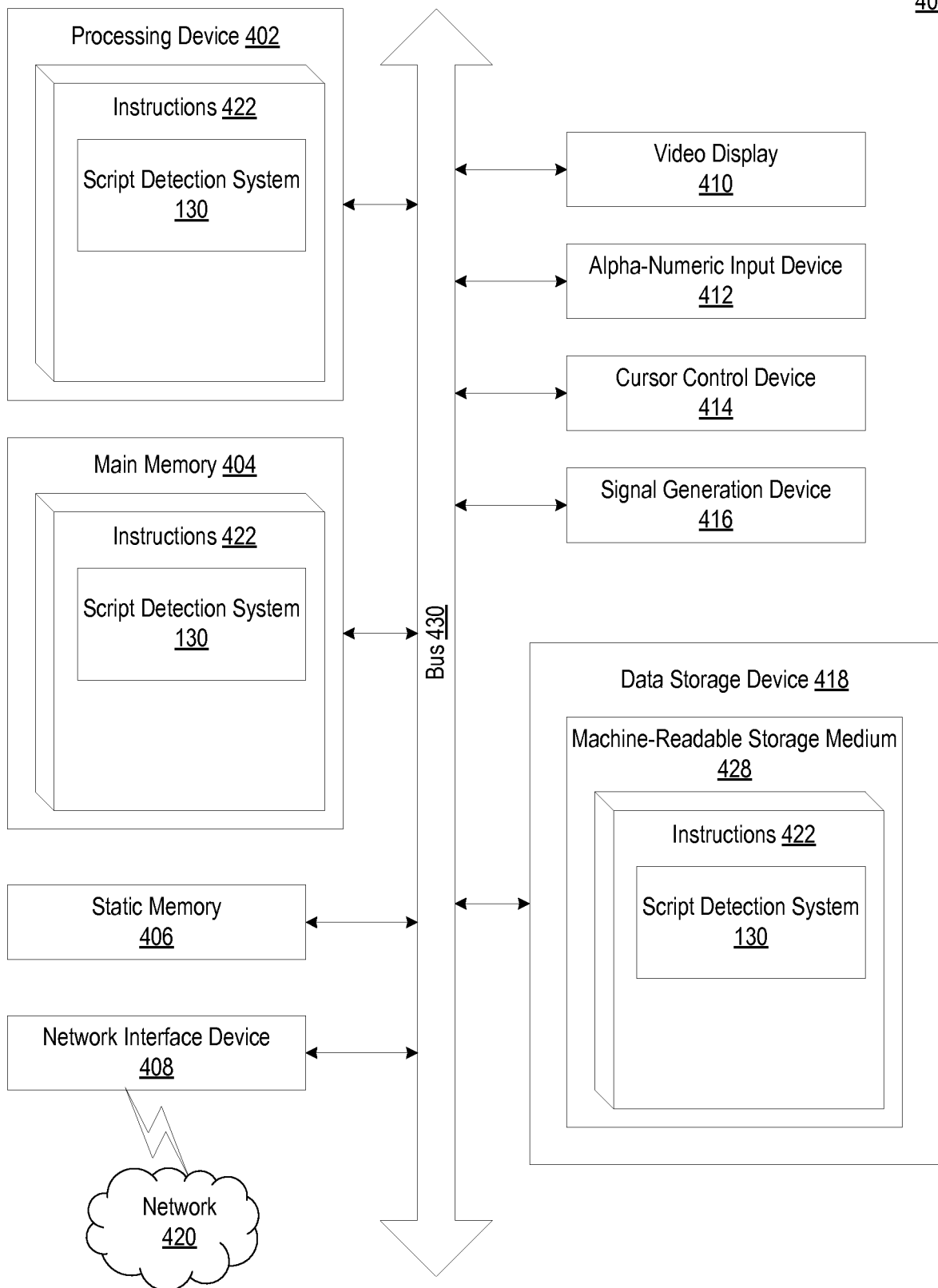
19. A computer-implemented method, comprising:
issuing, by a processor, a challenge to an application interacting with a computer system to determine whether activity performed by the application is scripted;
analyzing, by the processor, data associated with a result of the challenge to determine whether the result is at least partially correct; and
computing, by the processor, a reputation score for the application based on the result of the challenge.

20. A non-transitory computer-readable medium comprising computer-readable instructions which, when executed by a processor, cause the processor to perform operations comprising:
issuing a challenge to an application interacting with a computer system to determine whether activity performed by the application is scripted;
analyzing data associated with a result of the challenge to determine whether the result is at least partially correct; and
computing a reputation score for the application based on the result of the challenge.

100**FIG. 1**

200**FIG. 2**

300**FIG. 3**

400**FIG. 4**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US15/27595

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - G06F 21/50 (2015.01)

CPC - G06F 21/50

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)

IPC(8) Classification(s): G06F 21/50, 21/55, 21/57, 21/00, 15/173 (2015.01); CPC Classification(s): G06F 21/50, 21/55, 21/554, 21/562, 21/566, 21/57, 21/577, 21/316, 15/173; H04L 67/02, 63/14, 63/1408, 63/1416, 63/1441, 63/1466, 63/1483; G06N 7/005

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatSeer (US, EP, WO, JP, DE, GB, CN, FR, KR, ES, AU, IN, CA, INPADOC); IEEE/IEEEExplore; Google/Google Scholar; IP.com; detect, threat, attack, suspicious, malicious, analysis, assess, cyber, web, internet, network, reputation, challenge, test, classify, categorize, script, code, response, result, rank, score, probability, threshold, reference

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X ----- Y	US 2006/0253581 A1 (DIXON, C et al.) November 9, 2006; paragraphs [0007], [0015], [0087], [0101], [0131], [0140], [0254], [0298].	1-9, 12-16, 18-20 ----- 10, 11, 17
Y	US 2011/0280160 A1 (YANG, W) November 17, 2011; paragraphs [0034], [0036], [0049].	10, 11, 17
A	US 2013/0298192 A1 (KUMAR, S et al.) November 7, 2013; entire document.	1-20



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

13 July 2015 (13.07.2015)

Date of mailing of the international search report

28 JUL 2015

Name and mailing address of the ISA/

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Shane Thomas

PCT Helpdesk: 571-272-4300
PCT OSP: 571-272-7774