COMPUTER SYSTEM AND METHOD FOR SECURE ACCESS CONTROL OF AN ONLINE FORM PAGE

Technical Field

The present disclosure relates generally to access of online form pages, and in particular to a computer system and computerized method for secure access control of an online form page.

Background

With the rapid development and popularization of the Internet, more industries are shifting their offline businesses online, including the financial industry and particularly the credit industry. For example, some finance companies or banks can complete loan applications and approval processes for end users or customers purely online. However, due to the purely online loan application process, there is a greater risk of fraud that can result in financial losses for the banks and/or users.

Real users typically use web browsers to access online form pages on bank websites and fill forms in the online form pages for their loan applications. However, malicious users can use headless browsers, which are web browsers without a graphical user interface, to illegally access the online form pages and execute automated web application operations. Specifically, these headless browsers continuously perform automated operations such as filling information on the forms and submitting the filled forms for fraudulent loan applications. Further, the information filled on the loan applications may possibly have been stolen from real users illegally. Automated operations performed by headless browsers thus pose significant financial fraud risks to banks and other loan-providing companies and may also endanger the privacy and financial security of real users whose information has been stolen.

Therefore, in order to address or alleviate at least one of the aforementioned problems and/or disadvantages, there is a need to provide secure access control of online form pages.

Summary

In accordance with a first aspect of the present disclosure, a computer system and computerized method for secure access control of an online form page is disclosed. The method comprises: receiving, from a user device, an access request to access the online form page, the access request sent by a browser application executed on the user device; receiving, from the user device, browser data associated with the browser application; verifying the browser data; authorizing or denying the access request based on verification of the browser data; and sending the online form page to the user device upon authorization of the access request.

In an embodiment, the browser data comprises a browser kernel of the browser application, and wherein the access request is authorized or denied based on verification of the browser kernel.

In an embodiment, the verification of the browser kernel comprises: sending a browser type detection instruction to the user device, the browser type detection instruction predetermined based on the browser kernel; and receiving, from the user device, a browser type detection result from execution of the browser type detection instruction by the browser application, wherein the access request is authorized if the browser type detection result matches the browser kernel.

In an embodiment, the browser type detection instruction is configured to cause an exception in the browser kernel when the browser application executes the browser type detection instruction. The exception may comprise a program execution statement error or a memory exception error.

In an embodiment, the browser data comprises browser attributes of the browser application, and wherein the access request is authorized or denied based on verification of the browser attributes.

In an embodiment, the browser attributes comprise a user agent header of the browser application, and wherein the access request is denied if the user agent header comprises a keyword associated with headless browsers.

In an embodiment, the browser attributes comprise graphics data associated with rendering of graphics on the browser application, and wherein the access request is denied if the graphics data is empty. The graphics data may comprise a web graphics library and/or graphics card details of a graphic card of the user device.

In an embodiment, the browser attributes comprise a navigator interface property of the browser application, and wherein the access request is denied if the navigator interface property comprises a true value.

In an embodiment, the browser attributes comprise a window size of the browser application, and wherein the access request is denied if the window size is zero.

In an embodiment, the method further comprises receiving, from the user device, a device display resolution of the user device and an available display resolution for loading the online form page on the browser application, wherein the access request is authorized or denied based additionally on the device display resolution and available display resolution. The access request may be authorized if the device display resolution and available display resolution are different.

In accordance with a second aspect of the present disclosure, a computer system and computerized method for secure access control of an online form page is disclosed. The method comprises: sending the online form page to a user device having a browser application executed thereon for filling a form in the online form page; receiving, from the user device, a submission request to submit the filled form; receiving, from the user device, temporal data associated with filling of the form; verifying the temporal data; authorizing or denying the submission request based on verification of the temporal data; and submitting the filled form for further processing upon authorization of the submission request.

In an embodiment, the temporal data comprises a dwell time which is a time taken to fill the form, and wherein the submission request is denied if the dwell time is below a predefined dwell time threshold.

In an embodiment, the temporal data comprises a fill time which is a time taken to fill a form field in the form, and wherein the submission request is denied if the fill time is below a predefined fill time threshold.

In an embodiment, the form comprises a set of hidden form fields, and wherein the submission request is authorized or denied based additionally on the set of hidden form fields. The submission request may be denied if the hidden form fields are filled. In an embodiment, the form comprises a set of valid form fields corresponding to the set of hidden form fields, and wherein the submission request is denied if the hidden form fields and the valid form fields have the same filled content.

In an embodiment, the form comprises a set of valid form fields corresponding to the set of hidden form fields, wherein the hidden form fields have default filled content, and wherein the submission request is authorized if the hidden form fields are unchanged from the default filled content.

In accordance with a third aspect of the present disclosure, a computer system and computerized method for secure access control of an online form page is disclosed. The method comprises: receiving, from a user device, an access request to access the online form page, the access request sent by a browser application executed on the user device; receiving, from the user device, browser data associated with the browser application; verifying the browser data; authorizing or denying the access request based on verification of the browser data; sending the online form page to the user device upon authorization of the access request; receiving, from the user device, a submission request to submit the filled form; receiving, from the user device, temporal data associated with filling of the online form page; verifying the temporal data; authorizing or denying the submission request based on verification of the temporal data; and submitting the filled form for further processing upon authorization of the submission request.

A computer system and method for secure access control of an online form page are thus disclosed herein. Various features, aspects, and advantages of the present disclosure will become more apparent from the following detailed description of the embodiments of the present disclosure, by way of non-limiting examples only, along with the accompanying drawings.

Brief Description of the Drawings

FIG. 1 is a block diagram illustrating a system architecture of a computer system for secure access control of an online form page, according to embodiments of the present disclosure.

FIG. 2 is a flowchart illustrating a computerized method implementable by the system architecture of FIG. 1 for secure access control of an online form page, including authorizing an access request to access the online form page, according to embodiments of the present disclosure.

FIG. 3 is a flowchart illustrating a computerized method implementable by the system architecture of FIG. 1 for secure access control of an online form page, including authorizing a submission request to submit a filled form in the online form page, according to embodiments of the present disclosure.

FIG. 4 is a flowchart illustrating a computerized method implementable by the system architecture of FIG. 1 for secure access control of an online form page, including authorizing the access request and the submission request, according to embodiments of the present disclosure.

FIG. 5 is a block diagram illustrating a technical architecture of a computing server, according to embodiments of the present disclosure.

Detailed Description

For purposes of brevity and clarity, descriptions of embodiments of the present disclosure are directed to a computer system and computerized method for secure access control of an online form page, in accordance with the drawings. While aspects of the present disclosure will be described in conjunction with the embodiments provided herein, it will be understood that they are not intended to limit the present disclosure to these embodiments. On the contrary, the present disclosure is intended to cover alternatives, modifications and equivalents to the embodiments described herein, which are included within the scope of the present disclosure as defined by the appended claims. Furthermore, in the following detailed description, specific details are set forth in order to provide a thorough understanding of the present disclosure. However, it will be recognized by an individual having ordinary skill in the art, i.e. a skilled person, that the present disclosure may be practiced without specific details, and/or with multiple details arising from combinations of aspects of particular embodiments. In a number of instances, well-known systems, methods, procedures, and components have not been described in detail so as to not unnecessarily obscure aspects of the embodiments of the present disclosure.

In embodiments of the present disclosure, depiction of a given element or consideration or use of a particular element number in a particular figure or a reference thereto in corresponding descriptive material can encompass the same, an equivalent, or an analogous element or element number identified in another figure or descriptive material associated therewith.

References to “an embodiment / example”, “another embodiment / example”, “some embodiments / examples”, “some other embodiments / examples”, and so on, indicate that the embodiment(s) / example(s) so described may include a particular feature, structure, characteristic, property, element, or limitation, but that not every embodiment / example necessarily includes that particular feature, structure, characteristic, property, element or limitation. Furthermore, repeated use of the phrase “in an embodiment / example” or “in another embodiment / example” does not necessarily refer to the same embodiment / example.

The terms “comprising”, “including”, “having”, and the like do not exclude the presence of other features / elements / steps than those listed in an embodiment. Recitation of certain features / elements / steps in mutually different embodiments does not indicate that a combination of these features / elements / steps cannot be used in an embodiment.

As used herein, the terms “a” and “an” are defined as one or more than one. The use of “/” in a figure or associated text is understood to mean “and/or” unless otherwise indicated. The term “set” is defined as a non-empty finite organisation of elements that mathematically exhibits a cardinality of at least one (e.g. a set as defined herein can correspond to a unit, singlet, or single-element set, or a multiple-element set), in accordance with known mathematical definitions. The terms “first”, “second”, etc. are used merely as labels or identifiers and are not intended to impose numerical requirements on their associated terms.

As used in the present disclosure, the terms “component”, “module”, “system”, “interface” and the like are generally intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a module may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a controller and the controller can be a component. One or more modules may reside within a process and/or thread of execution and a module may be localized on one computer and/or distributed between two or more computers. As another example, an interface can include I/O components as well as associated processor, application, and/or API components. In the context of the present disclosure, the information processing and response generation system and its constituent parts may be implemented as hardware, software, or a combination thereof.

Furthermore, various embodiments of the present disclosure may be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof to control a computer to implement the disclosed subject matter. For instance,

the claimed subject matter may be implemented as a computer-readable medium embedded with a computer executable program, which encompasses a computer program accessible from any computer-readable storage device or storage media. For example, computer readable media can include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips, etc.), optical disks (e.g., compact disk (CD), digital versatile disk (DVD), etc.), smart cards, and flash memory devices (e.g., card, stick, key drive, etc.).

FIG. 1 is a block diagram illustrating a system architecture of a computer system 100 for secure access control of an online form page, according to various embodiments of the present disclosure. The online form page is hosted on an online website and includes at least one form and may include other page elements. Each form includes a plurality of form fields. Each form field can be configured with various attributes, including but not limited to, interaction type, name, value, size, minimum length, maximum length, check box, radio button, etc. A form field may be of various interaction types based on the input options, which may include one or more of input boxes, radio buttons, checkboxes, dropdown lists, buttons, sliders, scroll bars, and file upload boxes. The form fields can be filled or input by one or more of typing input, check box selection, radio button selection, dropdown list selection, scroll selection, etc. For example, form fields may be filled with alphanumeric text, such as single-line or multi-line text. For example, a form field may be filled with a name, password, identification number, email address, date, gender, website addresses, etc.

A user may use a user device 110, such as a computer, mobile phone, or tablet device, to communicate with the computer system 100 to access the online form page. Specifically, a user executes a browser application on the user device 110 to access the online form page. A browser application or web browser is a software application configured to access websites. A browser application may include independent browsers, such as Chrome or Firefox, and application interfaces provided based on browser kernels within various software applications.

The computer system 100 and user device 110 are communicative with each other through a communication network 120. The communication network 120 is a network

layer, medium, or environment through which content, notifications, and/or messages are communicated among various components. Suitable security protocols, such as encryption protocols, may be implemented in the communication network 120 for secure communications among the components. Some non-limiting examples of the communication network 120 include a virtual private network (VPN), wireless fidelity (Wi-Fi) network, light fidelity (Li-Fi) network, local area network (LAN), wide area network (WAN), LoRaWAN, metropolitan area network (MAN), satellite network, Internet, fibre optic network, coaxial cable network, infrared (IR) network, radio frequency (RF) network, and any combination thereof. Various components in the communication network 120 may connect to the communication network 120 in accordance with various wired and wireless communication protocols, such as Transmission Control Protocol / Internet Protocol (TCP/IP), User Datagram Protocol (UDP), 2nd to 5th Generation (2G to 5G) communication protocols, Long Term Evolution (LTE) communication protocols, and any combination thereof. Each component to the communication network 120 includes a data communication or transceiver module to communicate and transmit / receive data over the communication network 120. Some non-limiting examples of a transceiver module include an antenna module, a radio frequency transceiver module, a wireless transceiver module, a Bluetooth transceiver module, an Ethernet port, a Universal Serial Bus (USB) port, or any other module / component / device configured for transmitting and receiving data.

The computer system 100 includes a communication module 102 configured for communicating with the user device 110. Specifically, the communication module 102 provides a communication node for enabling data communication through the communication network 120. The computer system 100 further includes a verification module 104 configured for performing various verification processes to enable secure access control of the online form page.

The user may access a website by entering a URL address on the browser application executed on the user device 110. The access to the online form page hosted on the website may be executed in response to various operations, such as clicking on links

or buttons. It will be appreciated that the online form page may include links or buttons that can be clicked to attempt to access another online form page.

For example, a real human user may access an online form page, such as by clicking on the “Apply” button within the website, to fill or complete one or more forms, such as to apply for a financial loan application. Further, in the loan application, personal information may need to be filled in multiple forms in the form page or in multiple form pages. Each form page may include a “Next” button for the user to request access to the next form page.

In some situations, illegal or malicious users can use headless browsers to illegally access the online form pages and execute automated operations, such as to automate completion of fraudulent loan applications. Automated operations performed by headless browsers thus pose significant financial fraud risks and may also endanger the privacy and financial security of real users whose information has been stolen. The computer system 100 and various methods implemented on the computer system 100 seeks to address one or more of these problems.

FIG. 2 is a flowchart illustrating a computerized method 200 for secure access control of an online form page, according to various embodiments of the present disclosure. Particularly, the method 200 is implemented in the computer system 100. The method 200 outlines the key steps involved in controlling secure access to the online form page in response to an access request from the user.

At step 202, the communication module 102 receives, from the user device 110, an access request to access the online form page, wherein the access request is sent by the browser application executed on the user device 110. For example, the user may send the access request by clicking on a link or button. The access request may optionally include a timestamp of the aforementioned access operation by the user.

At step 204, the communication module 102 receives, from the user device 110, browser data associated with the browser application. For example, the browser data may include a browser kernel and/or browser attributes of the browser application.

At step 206, the verification module 104 verifies the browser data. Specifically, the verification module 104 compares the browser data against predefined conditions and determines whether the browser data satisfies these conditions.

At step 208, the verification module 104 authorizes or denies the access request based on verification of the browser data. Specifically, the access request is authorized if the browser data satisfies the predefined conditions, and conversely the access request is denied if the browser data fails to satisfy the predefined conditions.

At step 210, the communication module 102 sends the online form page to the user device 110 upon authorization of the access request. Thus, the user would be able to access the online form page after the access request has been authorized. Conversely, the user would not be able to view the online form page if the access request is denied.

FIG. 3 is a flowchart illustrating a computerized method 300 for secure access control of an online form page, according to various embodiments of the present disclosure. Particularly, the method 300 is implemented in the computer system 100. The method 300 outlines the key steps involved in controlling secure access to the online form page in response to a submission request from the user. Specifically, the submission request is sent by the user after the user has received the online form page and filled one or more forms in the online form page, such as to submit a completed loan application form.

At step 302, the communication module 102 sends the online form page to the user device 110 having the browser application executed thereon for filling a form in the online form page.

At step 304, the communication module 102 receives, from the user device 110, a submission request to submit the filled form. Particularly, the form includes multiple form fields that require input from the user.

At step 306, the communication module 102 receives, from the user device 110, temporal data associated with filling of the form. For example, the temporal data includes a dwell time or a fill time. The dwell time may be defined as the time taken fill the form. The fill time may be defined as the time taken to fill a form field in the form.

At step 308, the verification module 104 verifies the temporal data. Specifically, the verification module 104 compares the temporal data against predefined conditions and determines whether the temporal data satisfies these conditions.

At step 310, the verification module 104 authorizes or denies the submission request based on verification of the temporal data. Specifically, the submission request is authorized if the temporal data satisfies the predefined conditions, and conversely the submission request is denied if the temporal data fails to satisfy the predefined conditions.

At step 312, the communication module 102 submits the filled form for further processing upon authorization of the submission request. For example, the completed form is for a financial loan application and would be sent for further processing, such as by the relevant department of the bank to consider whether to approve the loan to the user. Conversely, the completed form would not be sent for further processing if the submission request is denied.

FIG. 4 is a flowchart illustrating a computerized method 400 for secure access control of an online form page, according to various embodiments of the present disclosure. Particularly, the method 400 is implemented in the computer system 100. The method 400 outlines the key steps involved in controlling secure access to the online form page, including authorization of both access and submission requests based on verification of the browser and temporal data.

At step 402, the communication module 102 receives, from the user device 110, an access request to access the online form page, wherein the access request is sent by the browser application executed on the user device 110. The access request may optionally include a timestamp.

At step 404, the communication module 102 receives, from the user device 110, browser data associated with the browser application.

At step 406, the verification module 104 verifies the browser data. Specifically, the verification module 104 compares the browser data against predefined conditions and determines whether the browser data satisfies these conditions.

At step 408, the verification module 104 authorizes or denies the access request based on verification of the browser data.

If at step 408 the verification module 104 determines that the browser data does not satisfy the predefined conditions, the browser data fails the verification process and step 408 proceeds to step 410. At step 410, the access request is denied, and the access request is considered an illegitimate request. This may happen if the access request came from an illegal or malicious user using a headless browser.

If at step 408 the verification module 104 determines that the browser data satisfies the predefined conditions, the browser data is successfully verified and step 408 proceeds to step 412. At step 412, the access request is authorized, and the access request is considered a legal request from a real user.

In some embodiments, the browser data includes a browser kernel of the browser application, wherein the access request is authorized or denied based on verification of the browser kernel. The browser kernel, or browser engine, is the core software component of the browser application. Different browser applications can be classified according to their browser kernels. For example, Chrome browser uses the Blink kernel and Firefox browser uses the Gecko kernel.

Verification of the browser kernel includes sending a browser type detection instruction to the user device 110, wherein the browser type detection instruction is predetermined based on the browser kernel. Verification of the browser kernel may includes receiving, from the user device 110, a browser type detection result from

execution of the browser type detection instruction by the browser application. The access request is authorized if the browser type detection result matches the browser kernel.

For example, the browser type detection instruction is configured to cause an exception in the browser kernel when the browser application executes the browser type detection instruction. The exception may include any possible exception, including but not limited to, a program execution statement error or a memory exception error. By detecting whether an exception or error occurs when the browser application executes the browser type detection instruction, it can be determined whether the actual browser kernel of the browser application in the browser type detection result matches the declared browser kernel in the browser data.

In one example of the program execution statement error, the corresponding browser type detection instruction contain a set of statements that cannot be executed by a browser application and result in syntax errors. When the browser application attempts to execute the instruction, it will result in the error because it cannot understand or execute the instruction. In one example of the memory exception error, this can occur because the corresponding browser type detection instruction requires the browser application to perform certain operations, such as allocating a large amount of memory or accessing illegal memory addresses, that exceed the current memory processing capability or access permissions of the browser application. The browser application will trigger the memory exception error when it attempts to execute the instruction.

In some embodiments, the browser data includes browser attributes of the browser application, wherein the access request is authorized or denied based on verification of the browser attributes. Browser attributes are configuration information of the browser application that may be specific to the browser kernel.

In a first example of the browser attributes, the browser attributes include a user agent header of the browser application. The user agent header is a string that identifies the user device 110 making HTTP or web requests to a server. The user agent header includes details about the user device 110 and the browser application making the

requests. More specifically, the user agent header allows servers to identify the operating system and CPU type of the user device 110, the browser application and version, browser kernel, browser language, and browser plug-ins, etc.

If the user agent header has a keyword associated with headless browsers, the access request is denied and is considered an illegitimate request. Conversely, the access request is legal and authorized if the user agent header does not have a keyword associated with headless browsers. Examples of keywords associated with headless browsers include, but are not limited to, “Headless”, “Chrome Headless”, and “PhantomJS”. In one example, the browser application is Chrome and if the detected keyword is “Chrome Headless”, this may indicate that the headless mode of the Chrome browser or a headless browser simulating Chrome is being used to request access to the online page form. Examples of headless Chrome browsers include Puppeteer, Chromeless, or a browser simulation by Selenium. In another example, PhantomJS is a headless browser based on the WebKit engine, and the user agent header contains the keyword “PhantomJS”.

In a second example of the browser attributes, the browser attributes include graphics data associated with rendering of graphics on the browser application. The graphics data may include a web graphics library and/or graphics card details of a graphic card of the user device 110. If the graphics data is empty, the access request is denied and is considered an illegitimate request. Conversely, the access request is legal and authorized if the graphics data is not empty.

The web graphics library may include WebGL which is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without plug-ins. WebGL can provide hardware-accelerated 3D rendering for HTML5 Canvas. In one example, the browser application is Firefox and if the detected WebGL data is empty or the graphics card information is empty, this may indicate that the browser application is a headless Firefox simulated by Selenium.

In a third example of the browser attributes, the browser attributes include a navigator interface property of the browser application. If the navigator interface property has a

true value, the access request is denied and is considered an illegitimate request. Conversely, the access request is legal and authorized if the navigator interface property has a false value.

The navigator interface property may include the Navigator.WebDriver property. WebDriver is a W3C standard for a platform- and language-neutral wire protocol that allows out-of-process programs to remotely instruct the behaviour of web browsers. In one example, the Navigator.WebDriver property has a true value if the browser application is a headless browser simulated by Selenium, so it can be determined that the access request from the simulated headless browser is an illegitimate request that would be denied.

In a fourth example of the browser attributes, the browser attributes include a window size of the browser application. If the window size is zero, the access request is denied and is considered an illegitimate request. Conversely, the access request is legal and authorized if the window size is greater than zero. For example, a headless browser such as Phantom JS does not have any window to function as a user interface, so the external width and height of the window are both zero. Thus, it can be determined from the window size of zero that the browser application is a headless browser and the illegitimate access request would be denied.

In some embodiments, the browser data includes the browser attributes without the browser kernel, wherein the access request is authorized or denied based on verification of the browser attributes, without verification of the browser kernel. The browser attributes may include one or more of the user agent header, graphics data, navigator interface property, and window size. It will be appreciated that verification of these browser attributes is described above and is not elaborated upon for purpose of brevity.

In some embodiments, the browser data includes the browser kernel and browser attributes of the browser application, wherein the access request is authorized or denied based on verification of the browser kernel and browser attributes. For example, the browser attributes or configuration information of the browser application are

specific to the browser kernel so that different browser attributes may be detected according to the respective browser kernel. The verification module 104 detects the browser kernel and browser attributes and proceeds to verify them against the predefined conditions, such as explained in the examples above, and determine whether the access request is an illegal request or a legitimate request.

Verifying the browser kernel declared by the browser application can prevent or reduce the risk of automated operations by headless browsers. This is because illegal users may intentionally confuse the browser kernel to avoid security detection by website hosts. Further, when the browser kernel is verified, browser attributes related to the browser kernel can be subsequently verified, effectively enhancing the effectiveness of the verification process.

In some embodiments, the method 400 further includes a step of receiving, from the user device 110, a device display resolution of the user device 110 and an available display resolution for loading the online form page on the browser application. The access request is authorized or denied based additionally on the device display resolution and available display resolution. Specifically, the access request is authorized if the device display resolution and available display resolution are different, and the access request is denied if the display resolutions are the same.

When a regular browser application is used by a real user to access websites and load webpages, the device display resolution and the available display resolution are usually different due to various factors such as window size adjustments and browser interface elements that occupy the display screen space. As for headless browsers, they typically do not have actual display devices, so the device display resolution of a headless browser and the available display resolution will be the same. Therefore, by comparing these two display resolutions, it is possible to determine whether the access request comes from a regular browser or a headless browser, thereby determining whether the access request is legal or illegitimate, respectively.

At step 410, the access request is illegitimate and is denied and the online form page would not be sent to the user device 110. Further, at step 410, various operations may

be performed in response to the access request being illegitimate. The operations may include recording the illegal access request in a database, which may include details of the user device 110 and browser application. The operations may include sending a predetermined page to the user device 110 so it may be displayed on the browser application. Displaying the predetermined page may indicate to the user, whether a real user or illegal user, that the access request has been identified as illegal and additional security measures have been triggered.

At step 412, the access request is legal and is authorized. Further, at step 412, various operations may be performed in response to the submission request being legal. The operations may include sending a predetermined page to the user device 110 so it may be displayed on the browser application.

At steps 410 and 412, the same predetermined page may be sent to the user device 110 for display on the browser application when the access request is either denied or authorized, respectively. However, different background operations may be performed by the computer system 100 depending on whether the access request is denied or authorized. For example, the predetermined page may show “Under Review” to inform the user that the access request is being processed. If the access request is illegal (i.e. step 410), the background operations may include storing the illegal access request in a database. If the access request is legal (i.e. step 412), the background operations include sending the online form page to the user device 110.

At step 414, upon authorization of the legal access request, the communication module 102 sends the online form page to the user device 110 for the user to fill a form in the online form page. It will be appreciated that the online form page may have multiple forms to be filled by the user.

At step 416, the communication module 102 receives, from the user device 110, a submission request to submit the filled form.

At step 418, the communication module 102 receives, from the user device 110, temporal data associated with filling of the form.

At step 420, the verification module 104 verifies the temporal data. Specifically, the verification module 104 compares the temporal data against predefined conditions and determines whether the temporal data satisfies these conditions.

At step 422, the verification module 104 authorizes or denies the access request based on verification of the temporal data.

If at step 422 the verification module 104 determines that the temporal data does not satisfy the predefined conditions, the temporal data fails the verification process and step 422 proceeds to step 424. At step 424, the submission request is denied, and the submission request is considered an illegitimate submission. Like the access request, this may happen if the submission request came from an illegal or malicious user using a headless browser.

If at step 422 the verification module 104 determines that the browser data satisfies the predefined conditions, the temporal data is successfully verified and step 422 proceeds to step 426. At step 426, the submission request is authorized, and the submission request is considered a legal submission from a real user.

In some embodiments, the temporal data includes a dwell time, which is the time taken in filling the form, wherein the submission request is denied if the dwell time is below a predefined dwell time threshold. Various methods may be used to determine the dwell time based on respective start and end times. It will be appreciated that the start and end times to determine the dwell time may be corrected to account for different time zones, particularly if the computer system 100 and user device 110 operate in different time zones.

In a first example, the start time is the start filling time and the end time is the form submission time, and the dwell time is determined to be the time difference between the start filling time and form submission time. The start filling time is when the user begins to fill the form. The form submission time is when the user sends the submission request to submit the completed form.

In a second example, the start time is the form sending time and the end time is the form submission time, and the dwell time is determined to be the time difference between the form sending time and form submission time. The form sending time is when the online form page is sent to the browser application. The form sending time can be easily determined by the computer system 100 and can avoid tampering by unauthorized users.

In a third example, the start time is the form loading time and the end time is the form submission time, and the dwell time is determined to be the time difference between the form loading time and form submission time. The form loading time is when the browser application completes loading the online form page. Specifically, the computer system 100 detects the time stamp when the form page loading is completed through various methods, such as by listening for browser load events or parsing relevant header information in the HTTP request sent by the browser application, such as the X-Requested-With header.

The form submission time may be defined by the time when the user clicks the link or button to trigger the submission of the completed form. For example, the form submission time is defined by the time when the “Submit” link or button is clicked. As mentioned above, the online form page may have multiple forms to be filled by the user, and clicking the “Submit” link or button or other similarly-worded function triggers the submission completed multiple forms.

Further, in certain applications, multiple forms in multiple form pages may need to be submitted. In a non-final form page having a set of forms, the completed forms may be submitted by clicking on the “Next” or similarly-worded function. This triggers submission of the completed forms in the non-final form page and also triggers a request to access the next form page, which may be another non-final form page or a final form page. It will be appreciated that it may not be necessary to verify the browser data to authorize the new access request, since the verification has been done for the first form page. It will also be appreciated that verification of the temporal data may be

done upon completion of all the forms in all the form pages to thereby authorized submission of all the completed forms.

The submission request for a completed form is authorized if the corresponding dwell time is equal to or more than the predefined dwell time threshold, and the submission request is denied if the dwell time is below the predefined dwell time threshold. This is because the form is likely to be completed in an automated operation by a headless browser if the dwell time is too short, compared to a real user who needs more time to manually fill in the form. The predefined dwell time threshold may be defined based on the mean value of the dwell times of multiple real users filling the corresponding form.

In some embodiments, the temporal data includes a fill time, which is the time taken to fill a form field of the online form page, wherein the submission request is denied if the fill time is below a predefined fill time threshold.

Preferably, the form field is a responsive form field, and the online form page includes a plurality of responsive form fields. In a regular form, the content of the form fields is often obtained by the computer system 100 from the user device 110 when the form is submitted. A responsive form field is a form field, such as an input box, wherein its filling or input status can be obtained in real-time when it is being filled or input. This allows the time duration in filling the responsive form field to be obtained in real-time. Optionally, the filling or input content of the responsive form field may also be obtained in real-time.

A responsive form field is configured to be listenable to achieve real-time monitoring of the user's operation of filling or inputting the responsive form field. The time duration in filling each responsive form field can be obtained and the respective fill time can be determined based on the respective time duration. It will be appreciated that different listening form field configurations can be set for different browser applications based on their browser kernel and attributes.

For example, the jQuery oninput script is used as an input box and the timestamps in filling the responsive form fields can be recorded in real-time and sent to the computer system 100. For example, the jQuery oninput script obtains a first timestamp for a first responsive form field, wherein the first timestamp represents when the user begins filling or inputting the first responsive form field. The jQuery oninput script then obtains a second timestamp for a second responsive form field, wherein the second timestamp represents when the user begins filling the second responsive form field. The fill time of the first responsive form field can be determined based on the difference between the first and second timestamps. Similarly, the jQuery oninput script can obtain a third timestamp for a third responsive form field, wherein the third timestamp represents when the user begins filling the third responsive form field, and that the fill time of the second responsive form field can be determined based on the difference between the second and third timestamps.

Alternatively, scripts or programs other than the jQuery oninput script may be used to achieve real-time monitoring of user input in the responsive form fields. These may include native JavaScript event listeners, such as addEventListener, that can monitor events such as input and change in form fields, thereby achieving real-time monitoring. Further, some frontend frameworks, such as React and Vue, may be used to build dynamic and interactive user interfaces for websites and these frameworks may provide their own event listening mechanisms that can be used to achieve real-time monitoring.

The submission request for a completed form is authorized if the fill time for a form field is equal to or more than the predefined fill time threshold, and the submission request is denied if the fill time is below the predefined fill time threshold. This is because the form field is likely to be completed in an automated operation by a headless browser if the fill time is too short, compared to a real user who needs more time to manually input content into the form field. The predefined fill time threshold for a form field may be defined based on the mean value of the fill times of multiple real users filling the corresponding form field. The mean value may be determined based on the time duration in filling the form field or the timestamps from multiple form fields, as described in the examples above.

In some embodiments, the temporal data includes multiple fill times for all the form fields. The submission request for a completed form is authorized if the fill time for each respective form field is equal to or more than the respective predefined fill time threshold for the respective form field. Conversely, the submission request is denied if at least one, or any other number, of the fill times is below the respective predefined fill time threshold.

In some embodiments, the temporal data includes the dwell time and one or more fill times for one or more form fields. The submission request for a completed form is authorized if the dwell time is equal to or more than the predefined dwell time threshold, and if the fill time for each respective form field is equal to or more than the respective predefined fill time threshold for the respective form field. Conversely, the submission request is denied if the dwell time is less than the predefined dwell time threshold, or if at least one, or any other number, of the fill times is below the respective predefined fill time threshold.

Verifying the fill time may be preferred instead of or in addition to the dwell time. This is because illegal users may may attempt to add delays at the online form page to intentionally lengthen in the dwell time when performing automated operations through headless browsers. For example, the headless browser intentionally waits at the online form page without filling any of the form fields. Verifying the fill time for each form field can prevent or reduce the risk of automated operations by headless browsers.

In some embodiments, listening configurations for the form fields can be used to verify the browser kernel and/or browser attributes of the browser application being used to input the form fields. If the monitoring result is exceptional, i.e. triggers can exception error, it can be determined that the actual browser kernel does not match the declared browser kernel. For example, an exception error may be the inability to monitor the dwell time and/or fill time. Various scripts or programs can be used for real-time monitoring of the browser kernel and/or browser attributes, such as the jQuery oninput script. It will be appreciated that different listening configurations can be set for different browser applications based on their browser kernel and attributes.

In some embodiments, the form includes a set of hidden form fields, wherein the submission request is authorized or denied based additionally on the set of hidden form fields. Specifically, the submission request is denied if the hidden form fields are filled.

A hidden form field is invisible to a real user as the hidden form field is not displayed on the browser application, so the real user would not be able to fill the hidden form field. However, an illegal user using a headless browser for automated operations may detect the hidden form field and attempt to fill the hidden form field. It is difficult for headless browsers to detect anomalies or differences between hidden and valid form fields, resulting in headless browsers attempting to fill both hidden and valid form fields when performing automated operations. Thus, verifying the filling status of the hidden form fields can prevent or reduce the risk of automated operations by headless browsers.

In a first example, the form has a set of valid form fields corresponding to the set of hidden form fields, wherein the submission request is denied if the hidden form fields are filled. The submission request is also denied if the hidden form fields and the valid form fields have the same filled content. For example, the form has three hidden form fields and three valid form fields corresponding to each other, each pair of hidden and valid form fields corresponding to name, email address, and phone number. A real user does not fill in these three hidden form fields because they are invisible. However, an illegal user may, when performing automated operations through a headless browser, cause the headless browser to detect the hidden form fields and input data into the hidden form fields. Moreover, due to the correspondence between these hidden form fields and valid form fields, the headless browser can input data in the valid form field, such as the name, email address, or phone number, and input the same data into the corresponding hidden form field. Therefore, the headless browser treats both the hidden and valid form fields as form fields that require data input and automatically fills each corresponding pair of form fields with the same input data.

In a second example, the form has a set of valid form fields corresponding to the set of hidden form fields, and the hidden form fields have default filled content. The submission request is authorized if the hidden form fields are unchanged from the default filled content. For example, the form has a hidden form field and a valid form field, both form fields corresponding to name. The hidden form field for name has default filled content that is unrelated to any normal name, such as a string of numbers and/or special characters. Although the hidden form field contains filled content, the default filled content is impossible to be the same as a normal name in the corresponding valid form field. Thus, when the default filled content in the hidden form field is not changed, the filled content of the valid form field and the hidden form field can be simply compared without excessive conversion. If the default filled content in the hidden form field is unchanged, this is likely because a real user is filling the form and is unable to see the hidden form field, hence resulting in authorization of the submission request.

At step 424, the submission request is illegitimate and is denied and the filled form would not be sent for further processing. Further, at step 424, various operations may be performed in response to the submission request being illegitimate. The operations may include recording the illegal submission request in a database, which may include details of the user device 110 and browser application. The operations may include sending a predetermined page to the user device 110 so it may be displayed on the browser application. Displaying the predetermined page may indicate to the user, whether a real user or illegal user, that the submission request has been identified as illegal and additional security measures have been triggered.

At step 426, the submission request is legal and is authorized. Further, at step 426, various operations may be performed in response to the submission request being legal. The operations may include sending a predetermined page to the user device 110 so it may be displayed on the browser application.

At steps 424 and 426, the same predetermined page may be sent to the user device 110 for display on the browser application when the submission request is either denied or authorized, respectively. However, different background operations may be

performed by the computer system 100 depending on whether the submission request is denied or authorized. For example, the predetermined page may show “Under Review” to inform the user that the submission request is being processed for submission. If the submission request is illegal (i.e. step 424), the background operations may include storing the illegal submission request in a database. If the submission request is legal (i.e. step 426), the background operations include submitting the filled form for further processing.

At step 428, upon authorization of the legal submission request, the communication module 102 submits the filled form for further processing. For example, the completed form is for a financial loan application and would be sent for further processing, which may include performing loan-related business operations such as approval and risk control.

As described in various embodiments herein, the computer system 100 and methods 200,300,400 implemented on the computer system 100 perform verification processes for secure access control of online page forms. Various conditions in these verification processes include the browser kernel, browser attributes, display resolution, dwell time of a user’s stay on the online form page, time taken to fill the form fields, and filling status of hidden form fields. The verification processes advantageously determine whether online business processes on the form page are being automated using headless browsers, and prevent illegal or malicious users or attackers from continuously automating online business processes. For example, automated completion of forms for financial loan applications by illegal users can be prevented, thereby mitigating financial fraud risks and protects the privacy and financial security of real users.

Embodiments of the present disclosure may be provided as a network of communicating devices (i.e. a “computerized network”). Embodiments of the invention may be also provided as a software application downloadable into a computer device to facilitate the computerized methods 200,300,400 described herein. The software application may be a computer program product, which may be stored on a non-

transitory computer-readable medium on a tangible data-storage device, such as a storage device of a server.

FIG. 5 is a block diagram illustrating a technical architecture 500 of a computing server for implementing the processes of FIG. 2 to FIG. 4, according to various embodiments of the present disclosure. The technical architecture 500 represents a computer server suitable for use as the computer system 100 according to various embodiments of the present disclosure. While a single computing server is shown, the methods may be implemented across multiple computers in a distributed computing environment.

The technical architecture 500 includes a processor 502 (which may be referred to as a central processor unit or CPU) that is in communication with memory devices including secondary storage 504 (such as disk drives), read only memory (ROM) 506, random access memory (RAM) 508. The processor 502 may be implemented as one or more CPU chips. The technical architecture may further comprise input/output (I/O) devices 510, and network connectivity devices 512.

The secondary storage 504 is typically comprised of one or more disk drives or tape drives and is used for non-volatile storage of data and as an over-flow data storage device if RAM 508 is not large enough to hold all working data. Secondary storage 504 may be used to store programs which are loaded into RAM 508 when such programs are selected for execution.

In this embodiment, the secondary storage 504 includes a component 504a comprising non-transitory instructions operative by the processor 502 to perform various operations of the methods 200,300,400. Specifically, the component 504a implements the functionality of the various components and modules illustrated in FIG. 1, including the communication module 102 and verification module 104 of the computer system 100.

The ROM 506 is used to store instructions and perhaps data which are read during program execution. The secondary storage 504, the RAM 508, and/or the ROM 506

may be referred to in some contexts as computer readable storage media and/or non-transitory computer readable media.

I/O devices 510 may include printers, video monitors, liquid crystal displays (LCDs), plasma displays, touch screen displays, keyboards, keypads, switches, dials, mice, track balls, voice recognizers, card readers, paper tape readers, or other well-known input devices.

The processor 502 executes instructions, codes, computer programs, scripts which it accesses from hard disk, floppy disk, optical disk (these various disk-based systems may all be considered secondary storage 504), flash drive, ROM 506, RAM 508, or the network connectivity devices 512. While only one processor 502 is shown, multiple processors may be present. Thus, while instructions may be discussed as executed by a processor, the instructions may be executed simultaneously, serially, or otherwise executed by one or multiple processors.

The network connectivity devices 512 enable the technical architecture 500 to communicate with external systems, which may include distributed computing resources or cloud-based services.

Although the technical architecture 500 is described with reference to a computer, it should be appreciated that the technical architecture 500 may be formed by two or more computers in communication with each other that collaborate to perform a task. For example, but not by way of limitation, an application may be partitioned in such a way as to permit concurrent and/or parallel processing of the instructions of the application. Alternatively, the data processed by the application may be partitioned in such a way as to permit concurrent and/or parallel processing of different portions of a data set by the two or more computers. In an embodiment, virtualization software may be employed by the technical architecture 500 to provide the functionality of a number of servers that is not directly bound to the number of computers in the technical architecture 500. In an embodiment, the functionality disclosed above may be provided by executing the application and/or applications in a cloud computing environment. Cloud computing may comprise providing computing services via a network

connection using dynamically scalable computing resources. A cloud computing environment may be established by an enterprise and/or may be hired on an as-needed basis from a third-party provider.

By programming and/or loading executable instructions onto the technical architecture 500, at least one of the CPU 502, the RAM 508, and the ROM 506 are changed, transforming the technical architecture 500 in part into a specific purpose machine or apparatus having the novel functionality taught by the present disclosure. It is fundamental to the electrical engineering and software engineering arts that functionality that can be implemented by loading executable software into a computer can be converted to a hardware implementation by well-known design rules. Such hardware implementations may include field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs), which can further enhance the system's performance for specific information processing tasks.

In the foregoing detailed description, embodiments of the present disclosure in relation to a computer system and computerized method for secure access control of an online form page are described with reference to the provided figures. The description of the various embodiments herein is not intended to call out or be limited only to specific or particular representations of the present disclosure, but merely to illustrate non-limiting examples of the present disclosure. The present disclosure serves to address at least one of the mentioned problems and issues associated with the prior art. Although only some embodiments of the present disclosure are disclosed herein, it will be apparent to a person having ordinary skill in the art in view of the present disclosure that a variety of changes and/or modifications can be made to the disclosed embodiments without departing from the scope of the present disclosure. It will be understood by those skilled in the art that many variations of the embodiments can be made within the scope of the following claims. Moreover, features of one or more embodiments may be mixed and matched with features of one or more other embodiments. Therefore, the scope of the disclosure as well as the scope of the following claims is not limited to embodiments described herein.

Claims

1. A computerized method for secure access control of an online form page, the method comprising:

receiving, from a user device, an access request to access the online form page, the access request sent by a browser application executed on the user device;

receiving, from the user device, browser data associated with the browser application;

verifying the browser data;

authorizing or denying the access request based on verification of the browser data; and

sending the online form page to the user device upon authorization of the access request.

2. The method of claim 1, wherein the browser data comprises a browser kernel of the browser application, and wherein the access request is authorized or denied based on verification of the browser kernel.

3. The method of claim 2, wherein verification of the browser kernel comprises:

sending a browser type detection instruction to the user device, the browser type detection instruction predetermined based on the browser kernel; and

receiving, from the user device, a browser type detection result from execution of the browser type detection instruction by the browser application,

wherein the access request is authorized if the browser type detection result matches the browser kernel.

4. The method of claim 3, wherein the browser type detection instruction is configured to cause an exception in the browser kernel when the browser application executes the browser type detection instruction.

5. The method of claim 4, wherein the exception comprises a program execution statement error or a memory exception error.

6. The method of any one of claims 1 to 5, wherein the browser data comprises browser attributes of the browser application, and wherein the access request is authorized or denied based on verification of the browser attributes.

7. The method of claim 6, wherein the browser attributes comprise a user agent header of the browser application, and wherein the access request is denied if the user agent header comprises a keyword associated with headless browsers.

8. The method of claim 6 or 7, wherein the browser attributes comprise graphics data associated with rendering of graphics on the browser application, and wherein the access request is denied if the graphics data is empty.

9. The method of claim 8, wherein the graphics data comprises a web graphics library and/or graphics card details of a graphic card of the user device.

10. The method of any one of claims 6 to 9, wherein the browser attributes comprise a navigator interface property of the browser application, and wherein the access request is denied if the navigator interface property comprises a true value.

11. The method of any one of claims 6 to 10, wherein the browser attributes comprise a window size of the browser application, and wherein the access request is denied if the window size is zero.

12. The method of any one of claims 1 to 11, further comprising receiving, from the user device, a device display resolution of the user device and an available display resolution for loading the online form page on the browser application, wherein the access request is authorized or denied based additionally on the device display resolution and available display resolution.

13. The method of claim 12, wherein the access request is authorized if the device display resolution and available display resolution are different.

14. A computer system for secure access control of an online form page, the system comprising:

a communication module configured for:

receiving, from a user device, an access request to access the online form page, the access request sent by a browser application executed on the user device;

receiving, from the user device, browser data associated with the browser application;

sending the online form page to the user device upon authorization of the access request; and

a verification module configured for:

verifying the browser data; and

authorizing or denying the access request based on verification of the browser data.

15. A computerized method for secure access control of an online form page, the method comprising:

sending the online form page to a user device having a browser application executed thereon for filling a form in the online form page;

receiving, from the user device, a submission request to submit the filled form;

receiving, from the user device, temporal data associated with filling of the form;

verifying the temporal data;

authorizing or denying the submission request based on verification of the temporal data; and

submitting the filled form for further processing upon authorization of the submission request.

16. The method of claim 15, wherein the temporal data comprises a dwell time which is a time taken to fill the form, and wherein the submission request is denied if the dwell time is below a predefined dwell time threshold.

17. The method of claim 15 or 16, wherein the temporal data comprises a fill time which is a time taken to fill a form field in the form, and wherein the submission request is denied if the fill time is below a predefined fill time threshold.

18. The method of any one of claims 15 to 17, wherein the form comprises a set of hidden form fields, and wherein the submission request is authorized or denied based additionally on the set of hidden form fields.

19. The method of claim 18, wherein the submission request is denied the hidden form fields are filled.

20. The method of claim 18, wherein the form comprises a set of valid form fields corresponding to the set of hidden form fields, and wherein the submission request is denied if the hidden form fields and the valid form fields have the same filled content.

21. The method of claim 18, wherein the form comprises a set of valid form fields corresponding to the set of hidden form fields, wherein the hidden form fields have default filled content, and wherein the submission request is authorized if the hidden form fields are unchanged from the default filled content.

22. A computer system for secure access control of an online form page, the system comprising:

a communication module configured for:

sending the online form page to a user device having a browser application executed thereon for filling a form in the online form page; and

receiving, from the user device, a submission request to submit the filled form;

receiving, from the user device, temporal data associated with filling of the form; and

submitting the filled form for further processing upon authorization of the submission request; and

a verification module configured for:

verifying the temporal data; and

authorizing or denying the submission request based on verification of the temporal data.

23. A computerized method for secure access control of an online form page, the method comprising:

receiving, from a user device, an access request to access the online form page, the access request sent by a browser application executed on the user device;

receiving, from the user device, browser data associated with the browser application;

verifying the browser data;

authorizing or denying the access request based on verification of the browser data;

sending, upon authorization of the access request, the online form page to the user device for filling a form in the online form page;

receiving, from the user device, a submission request to submit the filled form;

receiving, from the user device, temporal data associated with filling of the form;

verifying the temporal data;

authorizing or denying the submission request based on verification of the temporal data; and

submitting the filled form for further processing upon authorization of the submission request.

24. A computer system for secure access control of an online form page, the system comprising:

a communication module configured for:

receiving, from a user device, an access request to access the online form page, the request sent by a browser application executed on the user device; and

receiving, from the user device, browser data associated with the browser application;

sending, upon authorization of the access request, the online form page to the user device for filling a form in the online form page;

receiving, from the user device, a submission request to submit the filled form;

receiving, from the user device, temporal data associated with filling of the form; and

submitting the filled form for further processing upon authorization of the submission request; and

a verification module configured for:

verifying the browser data;

authorizing or denying the access request based on verification of the browser data;

verifying the temporal data; and

authorizing or denying the submission request based on verification of the temporal data.