METHOD, DEVICE AND SYSTEM FOR INVOKING LOCAL SERVICE ASSEMBLY BY BROWSER

Technical Field

The present application relates to the field of communication technologies, and in particular, to a method, an apparatus and a system for invoking a local service component by a browser.

Background

A browser plugin such as an NPAPI (Netscape Plugin Application Programming Interface) is a universal interface for executing an external application in a browser. An execution of an external application may also be referred to as an invocation of a local service component. The invocation of a local service component may include installing a digital certificate, deleting a digital certificate, verifying a digital certificate, acquiring environment information of a local machine such as a mac address of a network card, etc. Besides NPAPI, current common browser plugins having similar functions further include a BHO (Browser Helper Object), Native Messaging, JS-Ctypes, etc.

However, these browser plugins have some problems, which are specifically as follows:

Although almost all graphical interface browsers (except for IE) of desktop terminals support NPAPI, improper writing of an NPAPI plugin may cause a breakdown of a browser, or even lead to a malware attack to a system because no security consideration is made when NPAPI is designed. Moreover, the NPAPI plugin does not support mobile devices, and is on the verge of elimination at present. Currently, most of the latest browsers (such as chrome and Firefox, etc.) do not support NPAPI, and a browser plugin cannot be used to acquire local information or verify a digital certificate of a website. BHO, which is proposed by Microsoft,

is an industry standard for a browser to open an interactive interface to third-party programmers, and an interactive interface in the field of the browser may be entered through simple codes. Moreover, a number of IE browser personalizing tools are now implemented using BHO. However, BHO only supports the IE browser, and is limited in applications. Moreover, Google adds a new Native Messaging interface in a new chrome browser thereof, to replace an original NPAPI. However, Native Messaging only supports the chrome browser at present, and an extension needs to be installed in the browser. Similarly, JS-Ctypes only support the Firefox browser, and similarly, an extension needs to be installed in the browser.

Therefore, under the circumstance where NPAPI is to be eliminated but other browser plugins are not universal, a solution, which has a strong universality and a good security, and is capable of implementing invocation of a local service component by most mainstream browsers, is in urgent need currently.

Summary

An objective of embodiments of the present application is to provide a method, an apparatus and a system for implementing invocation of a local service component by a browser, to ensure most mainstream browsers to be able to invoke a local service component while taking security into account.

To achieve the above objective, in one aspect, the embodiments of the present application provide a method for implementing invocation of a local service component by a browser, which includes:

a client terminal receiving a connection request based on a cross-domain communication protocol initiated by a browser terminal;

the client terminal performing validity verification of the connection request; and

the client terminal establishing a communication connection with the browser terminal based on the cross-domain communication protocol to enable the browser terminal to invoke a local service component, if the connection request passes the validity verification.

In another aspect, the embodiments of the present application further provide an apparatus for implementing invocation of a local service component by a browser. The apparatus is a client terminal, and includes:

a connection request reception module configured to receive a request for a connection based on a cross-domain communication protocol initiated by a browser terminal;

a validity verification module configured to perform validity verification of the connection request; and

a connection control module configured to establish a communication connection with the browser terminal based on the cross-domain communication protocol to enable the browser terminal to invoke a local service component if the connection request passes the validity verification.

In still another aspect, the embodiments of the present application further provide a system for implementing an invocation of a local service component by a browser, which includes:

a browser terminal configured to initiate a connection request based on a cross-domain communication protocol to a client terminal; and

the client terminal configured to perform validity verification of the connection request, and establish a communication connection with the browser terminal based on the cross-domain communication protocol to allow the browser terminal to invoke a local

service component if the connection request passes the validity verification.

In the embodiments of the present application, the client terminal supports the cross-domain communication protocol and most existing mainstream browsers also support the cross-domain communication protocol, the client terminal is able to establish a communication channel with an existing mainstream browser. As such, under the circumstance that NPAPI is to be eliminated at present but other browser plugins are not universal, a communication solution enabling most mainstream browsers to invoke a local service component is provided. Furthermore, in the embodiments of the present application, prior to establishing a communication connection with a browser, the client terminal performs validity verification of a connection request of the browser. The communication connection for invoking a local service component can be established only when the request passes the validity verification. Therefore, the embodiments of the present application also possess a function of preventing a malicious website from utilizing a client terminal, thereby improving the security of communication for a browser to invoke a local service component.

Brief Description of the Drawings

Accompanying drawings described herein are used to provide further understanding of the embodiments of the present application, constitute a part of the present application, and are not construed as limitations to the embodiments of the present application. In the accompanying drawings:

FIG. 1 is a flowchart of an exemplary method for implementing invocation of a local service component by a browser according to an embodiment of the present application.

FIG. 2 is a logic structural diagram of an exemplary apparatus for implementing

invocation of a local service component by a browser according to an embodiment of the present application.

FIG. 3 is a logic structural diagram of an exemplary system for implementing invocation of a local service component by a browser according to an embodiment of the present application.

Detailed Description

In order to make the objectives, technical solutions and advantages of the embodiments of the present application in a more comprehensible manner, the embodiments of the present application are described herein in further detail with reference to the embodiments and the accompanying drawings. Illustrative examples and corresponding descriptions of the embodiments of the present application are used to explain the embodiments of the present application, and are not to be construed as limiting the embodiments of the present application.

Details of implementations of the embodiments of the present application are described herein in further detail with reference to the accompanying drawings.

Referring to FIG. 1, a method for implementing invocation of a local service component by a browser according to an embodiment of the present application includes the following operations.

At S101, a client terminal receives a connection request based on a cross-domain communication protocol initiated by a browser terminal. When the browser terminal needs to invoke a local service component, the browser terminal initiates a connection request based on a WebSocket protocol to the client terminal, and the client terminal receives the request correspondingly. The WebSocket protocol is a network technology of full duplex

communications. In the WebSocket protocol, communication parties only need a handshake action to quickly form a communication channel between the communication parties, and mutual data transmission may be directly conducted between the communication parties. Current mainstream browsers (such as Chrome16+, IE10+, Firefox11+, Safari6.0+, Opera12.10+, 360 Security Browser, 360 Chrome, Taobao™ Browser, Sogou Browser, UC Browser, Cheetah Browser, and Maxthon Browser) all support the WebSocket protocol.

It should be noted that a cross-domain communication protocol needs to be used between a client terminal and a browser in the embodiments of the present application, to enable most mainstream browsers to invoke a local service component. The cross-domain refers to crossing system security domains. The WebSocket protocol is preferred in the embodiments of the present application. However, the embodiments of the present application may also use other cross-domain communication protocols (such as a JSONP protocol and a CORS protocol, etc.) as an alternative. However, the JSONP protocol has a limit on the size of a data packet that is sent, and if the length of data that is sent exceeds the limit, the rest of the data may be truncated. When the CORS protocol is used, a special field needs to be added to a Response header that is returned by a client terminal to a server. The client terminal, however, cannot send a message to the server proactively, and the server needs to query the client terminal ceaselessly about whether there is data to be sent to the server. The WebSocket protocol does not have the limits of the above two protocols, and can establish a bi-directional communication channel for communication parties quickly. Moreover, when this type of communication approach of the WebSocket protocol is used, even the client terminal breaks down, a website cannot perceive this, because the client terminal can restart at the back end and resume an associated service. Therefore, the experience of the website is improved.

At S102, the client terminal performs validity verification of the connection request. If the validity verification succeeds, S103 is performed. Otherwise, the client terminal disconnects a connection with the browser terminal, to prevent a malicious website from utilizing the client terminal. In this operation, the purpose of the validity verification is to prevent a malicious website from utilizing the client terminal. In this way, the malicious website cannot obtain information of a local machine using the client terminal, thereby improving the security of communications. In this operation, the validity verification preferably performs verification by determining whether a website corresponding to the browser terminal initiating the connection request is a website included in a website white list. This is because a relative few number of websites need to invoke a local service component generally. Thus, verification using a website white list may accelerate the processing speed of validity verification of a client terminal. Apparently, the validity verification may also use other approaches, such as token verification, in this operation. The token verification refers to encryption using a non-symmetric encryption algorithm, and only a legal website can generate such token. A client terminal can identify modifications made to the token, and thus refuse a connection.

At S103, the client terminal establishes a communication connection with the browser terminal based on the WebSocket protocol to allow the browser terminal to invoke a local service component in response to the connection request passing the validity verification. After the communication connection is established, a process for the browser terminal to invoke a local service component may substantially include the following:

1) the browser terminal sending a command to the client terminal;

2) the client terminal parsing the command and delivering the command to a corresponding local service component;

3) the local service component processing the command and returning a processing result to the client terminal; and

4) the client terminal forwarding the processing result to the browser terminal.

In the embodiments of the present application, upon receiving the connection request based on the WebSocket protocol initiated by the browser terminal, the client terminal selects a port from a port list for binding according to a set port binding priority order. Moreover, the port binding priority order is consistent with a port selection priority order of the browser terminal. For example, the port list includes four selectable ports, namely, A, B, C and D, and the port binding priority order of the client terminal is A>B>C>D. Similarly, the port selection priority order of the browser terminal is also A>B>C>D. In this way, when the most preferred port A is occupied by another software, the client terminal and the browser terminal may both attempt to select the port B first, thereby being conducive to improving the success rate of establishing a connection, and shortening the time for establishing the connection.

Moreover, in the embodiments of the present application, when the client terminal breaks down, the client terminal sends a breakdown report thereof to the server, for performing corresponding improvements and upgrades on the client terminal according to the breakdown report at a later time. The breakdown report may include information such as an error code, an error location, and a current state, etc. The client terminal may obtain the breakdown report by invoking a fault diagnostic program that is built in an operating system via an Application Programming Interface (API) of the operating system.

In the embodiments of the present application, an implementation for updating the website white list at the client terminal side may be achieved using the following approach:

the client terminal querying the server periodically about whether an update of the

website white list exists, and submitting a request for acquiring the update if affirmative.

In the embodiments of the present application, since the client terminal supports the WebSocket protocol and most existing mainstream browsers also support the WebSocket protocol, the client terminal is able to establish a communication channel with an existing mainstream browser. As such, under the circumstance that NPAPI is to be eliminated at present and other browser plugins are not universal, a communication method that enables a majority of mainstream browsers to invoke a local service component is provided. Furthermore, in the embodiments of the present application, prior to establishing a communication connection with a browser, the client terminal performs validity verification of the connection request of the browser, and the communication connection will be established therewith only when the request passes the validity verification. Therefore, the embodiments of the present application also possess a function of preventing a malicious website from utilizing a client terminal, thereby improving the security of communications associated with an invocation of a local service component by a browser.

Referring to FIG. 2, an apparatus for implementing invocation of a local service component by a browser according to an embodiment of the present application is a client terminal. In correspondence with the foregoing method of implementing an invocation of a local service component by a browser, the client terminal includes a connection request reception module 21, a validity verification module 22, and a connection control module 23.

The connection request reception module 21 is configured to receive a connection request based on a cross-domain communication protocol initiated by a browser terminal. When the browser terminal needs to invoke a local service component, the browser terminal initiates a connection request based on a WebSocket protocol to the client terminal, and the connection request reception module 21 receives the request

correspondingly. The WebSocket protocol is a network technology of full duplex communications between a browser and a server. In the WebSocket protocol, communication parties only need a handshake action to form a communication channel quickly between the communication parties, and mutual data transmission may be directly conducted between the communication parties. Current mainstream browsers (such as Chrome16+, IE10+, Firefox11+, Safari6.0+, Opera12.10+, 360 Security Browser, 360 Chrome, Taobao™ Browser, Sogou Browser, UC Browser, Cheetah Browser, and Maxthon Browser) all support the WebSocket protocol.

It should be noted that a cross-domain communication protocol needs to be used between a client terminal and a browser in the embodiments of the present application, to enable most mainstream browsers to invoke a local service component. The cross-domain refers to crossing system security domains. The WebSocket protocol is preferred in the embodiments of the present application. However, the embodiments of the present application may also use other cross-domain communication protocols (such as a JSONP protocol and a CORS protocol, etc.) as an alternative. However, the JSONP protocol has a limit on the size of a data packet that is sent, and if the length of data that is sent exceeds the limit, the rest of the data may be truncated. When the CORS protocol is used, a special field needs to be added to a Response header that is returned by a client terminal to a server. The client terminal, however, cannot send a message to the server proactively, and the server needs to query the client terminal ceaselessly about whether there is data to be sent to the server. The WebSocket protocol does not have the limits of the above two protocols, and can establish a bi-directional communication channel for communication parties quickly. Moreover, when this type of communication approach of the WebSocket protocol is used, even the client terminal breaks down, a website cannot perceive this,

because the client terminal can restart at the back end and resume an associated service. Therefore, the experience of the website is improved.

The validity verification module 22 is configured to perform validity verification of the connection request. The purpose of the validity verification is to prevent a malicious website from utilizing the client terminal, thereby improving the security of communications. In the validity verification module 22, the validity verification preferably performs verification by determining whether a website corresponding to the browser terminal initiating the connection request is a website included in a website white list. This is because a relative few number of websites need to invoke a local service component, generally a few tens. Thus, verification using a website white list may accelerate the processing speed of validity verification of a client terminal. Apparently, the validity verification may also use other approaches, such as token verification, in this operation. The token verification refers to encryption using a non-symmetric encryption algorithm, and only a legal website can generate such token. A client terminal can identify modifications made to the token, and thus refuse a connection.

The connection control module 23 is configured to establish a communication connection with the browser terminal based on the WebSocket protocol to allow the browser terminal to invoke a local service component in response to the connection request passing the validity verification. Otherwise, the connection with the browser terminal is disconnected to prevent a malicious website from utilizing the client terminal. After the communication connection is established, a process for the browser terminal to invoke a local service component may substantially include the following:

1) the browser terminal sending a command to the client terminal;

2) the client terminal parsing the command and delivering the command to a

corresponding local service component;

3) the local service component processing the command and returning a processing result to the client terminal; and

4) the client terminal forwarding the processing result to the browser terminal.

In the embodiments of the present application, the client terminal may further include:

a port binding module 24 configured to select a port from a port list for binding according to a set port binding priority order when the connection request reception module 21 receives the connection request based on the cross-domain communication protocol initiated by the browser terminal, where the port binding priority order is consistent with a port selection priority order of the browser terminal. For example, the port list includes four selectable ports, namely, A, B, C and D, and the port binding priority order of the client terminal is A>B>C>D. Similarly, the port selection priority order of the browser terminal is also A>B>C>D. In this way, when the most preferred port A is occupied by another software, the port binding module 24 and the browser terminal may both attempt to select the port B first, thereby being conducive to improving the success rate of establishing a connection, and shortening the time for establishing the connection.

In the embodiments of the present application, the client terminal may further include:

a breakdown report processing module 25 configured to send a breakdown report of the client terminal to the server upon determining that the client terminal breaks down, for performing corresponding improvements and upgrades on the client terminal according to the breakdown report at a later time. The breakdown report may include information such as an error code, an error location, and a current state, etc. The breakdown report processing module 25 may obtain the breakdown report by invoking a fault diagnostic program that is built in an operating system via an Application Programming Interface of the

operating system.

Moreover, in the embodiments of the present application, the client terminal may further include:

an update module 26 configured to update the website white list at the client terminal by querying the server periodically.

In the embodiments of the present application, since the client terminal supports the WebSocket protocol and most existing mainstream browsers also support the WebSocket protocol, the client terminal is able to establish a communication channel with an existing mainstream browser. As such, under the circumstance that NPAPI is to be eliminated at present but other browser plugins are not universal, a communication method that enables a majority of mainstream browsers to invoke a local service component is provided. Furthermore, in the embodiments of the present application, prior to establishing a communication connection with the browser, the client terminal performs validity verification of the connection request of the browser, and the communication connection will be established therewith only when the request passes the validity verification. Therefore, the embodiments of the present application also possess a function of preventing a malicious website from utilizing a client terminal, thereby improving the security of communications associated with an invocation of a local service component by a browser.

Referring to FIG. 3, a system for implementing invocation of a local service component by a browser according to an embodiment of the present application includes a browser terminal 31 and a client terminal 32.

The browser terminal 31 is configured to initiate a connection request based on a cross-domain communication protocol to the client terminal 32.

The client terminal 32 is configured to perform validity verification of the connection

request sent by the browser terminal 31, and establish a communication connection between the client terminal 32 and the browser terminal 31 based on the WebSocket protocol if the connection request passes the validity verification, to allow the browser terminal to invoke a local service component. Details of the client terminal 32 may be obtained by referencing to the foregoing embodiments of the present application, and are not repeatedly described herein.

In the embodiments of the present application, the system further includes:

a server 33 configured to receive and store a breakdown report sent by the client terminal 31.

In the embodiments of the present application, since the client terminal supports the WebSocket protocol and most existing mainstream browsers also support the WebSocket protocol, a communication channel may be established between the client terminal and an existing mainstream browser. As such, under the circumstance that NPAPI is to be eliminated at present and other browser plugins are not universal, enabling a majority of mainstream browsers to invoke a local service component is achieved. Furthermore, in the embodiments of the present application, prior to establishing a communication connection with a browser, the client terminal performs validity verification of the connection request of the browser, and the communication connection will be established therewith only when the request passes the validity verification. Therefore, the embodiments of the present application also possess a function of preventing a malicious website from utilizing a client terminal, thereby improving the security of communications.

One skilled in the art may further understand that each illustrative logical block, unit and operation listed in the embodiments of the present application may be implemented by hardware, software or a combination thereof. Whether an implementation is performed

using hardware or software depends on a specific application and a design requirement of the entire system. One skilled in the art may use a variety of methods to implement the function(s) for each specific application, but such implementation should not be understood as falling outside the scope of protection of the embodiments of the present application.

Function(s) of each illustrative logical block or unit described in the embodiments of the present application may be implemented or operated via a general-purpose processor, a digital signal processor, an Application Specific Integrated Circuit (ASIC), a field-programmable gate array or another programmable logic device, a discrete gate or transistor logic, a discrete hardware component, or any combination thereof. The general-purpose processor may be a microprocessor, and optionally, the general-purpose processor may also be any conventional processor, controller, micro-controller or state machine. The processor may also be implemented using a combination of computing devices, such as a digital signal processor and a microprocessor, multiple microprocessors, one or more microprocessors associated with a digital signal processor core, or any other similar configurations.

The operations of the methods or algorithms described in the embodiments of the present application may be directly embedded in hardware, software module(s) executed by processor(s), or a combination thereof. Software module(s) may be stored in a RAM memory, a flash memory, a ROM memory, an EPROM memory, an EEPROM memory, a register, a hard disk, a removable disk, a CD-ROM, or a storage media in any other form in the field. As an illustration, a storage media may be coupled to a processor, such that the processor can read information from the storage media, and can store and write information into the storage media. Optionally, a storage media may also be integrated into a processor. The processor and the storage media may be disposed in an ASIC, and the ASIC may be

disposed in a user terminal. Optionally, the processor and the storage media may also be disposed in different components of the user terminal.

In one or more exemplary designs, the foregoing functions described in the embodiments of the present application may be implemented in hardware, software, firmware, or any combination thereof. If being implemented in software, these functions may be stored in a computer readable media, or transmitted to a computer readable media in a form of one or more instructions or codes. The computer readable media includes a computer storage media and a communication media that facilitates a transfer of a computer program from one place to another. The storage media may be any usable media accessible to any universal or specialized computer. For example, the computer readable media may include, but not limited to, a RAM, a ROM, an EEPROM, a CD-ROM or other optical disk storage, a magnetic disc storage or other magnetic storage devices, or any other media that can be used to carry or store program codes in an instruction or data structure or other forms that are readable by a universal or special computer or a universal or special processor. Moreover, any connection may be properly defined as a computer readable media. For example, if software is transmitted from a website, a server or another remote resource through a co-axial cable, an optical cable, a twisted pair, a digital subscriber line (DSL) or via a wireless manner such as infrared, wireless, and microwave, etc., this is also included as the defined computer readable media. The disk and the disc include a compact disc, a laser disk, an optical disk, a DVD, a floppy disc, and a blue-ray optical disk. The disc generally duplicates data magnetically, while the disk generally duplicates data optically using laser. A combination of the above may also be included as the computer readable media.

The exemplary embodiments described above describe the objectives, technical

solutions and beneficial effects of the embodiments of the present application in further detail. It should be understood that the above descriptions are merely exemplary embodiments of the embodiments of the present application, and are not intended to limit the scope of protection of the embodiments of the present application. Any modifications, equivalent replacements, improvements, etc., that are made without departing from the spirit and principles of the embodiments of the present application shall fall within the scope of protection of the embodiments of the present application.

Claims

1. A method for implementing an invocation of a local service component by a browser, the method comprising:

a client terminal receiving a connection request based on a cross-domain communication protocol initiated by a browser terminal;

the client terminal selecting a port from a port list for binding according to a set port binding priority order upon receiving the connection request;

the client terminal performing validity verification of the connection request; and

the client terminal establishing a communication connection with the browser terminal based on the cross-domain communication protocol to allow the browser terminal to invoke a local service component, if the connection request passes the validity verification.

2. The method of claim 1, wherein the port binding priority order being consistent with a port selection priority order of the browser terminal.

3. The method of claim 1, wherein the client terminal sends a breakdown report thereof to a server when the client terminal breaks down.

4. The method of claim 3, wherein the breakdown report comprises an error code, an error location, and a current state.

5. The method of claim 1, wherein the cross-domain communication protocol comprises a WebSocket protocol, a JSONP protocol, or a CORS protocol.

6. The method of claim 1, wherein the validity verification comprises determining, by the client terminal, whether a website corresponding to the browser terminal initiating the connection request is a website included in a website white list.

7. The method of claim 6, wherein the client terminal updates the website white list by querying a server periodically.

8. The method of claim 3, wherein the client terminal obtains the breakdown report by invoking a fault diagnostic program built in an operating system via an application programming interface of the operating system.

9. The method of claim 1, wherein a process of invoking the local service component by the browser after the communication connection is established comprises:

1) the client terminal receiving a command sent by the browser terminal;

2) the client terminal parsing the command and delivering the command to a corresponding local service component;

3) the client terminal receiving a processing result returned by the local service component after processing the command; and

4) the client terminal forwarding the processing result to the browser terminal.

10. An apparatus for implementing invocation of a local service component by a browser, wherein the apparatus is a client terminal, and comprises:

a connection request reception module configured to receive a connection request based on a cross-domain communication protocol initiated by a browser terminal;

a port binding module configured to select a port from a port list for binding according to a set port binding priority order when the connection request reception module receives the connection request;

a validity verification module configured to perform validity verification of the connection request; and

a connection control module configured to establish a communication connection with the browser terminal based on the cross-domain communication protocol to enable the browser terminal to invoke a local service component if the connection request passes the validity verification.

11. The apparatus of claim 10, wherein the port binding priority order being consistent with a port selection priority order of the browser terminal.

12. The apparatus of claim 10, wherein the client terminal further comprises:

a breakdown report processing module configured to send a breakdown report of the client terminal to a server upon determining that the client terminal breaks down.

13. The apparatus of claim 12, wherein the breakdown report comprises an error code, an error location, and a current state.

14. The apparatus of claim 10, wherein the cross-domain communication protocol comprises a WebSocket protocol, a JSONP protocol, or a CORS protocol.

15. The apparatus of claim 10, wherein the validity verification comprises:

determining, by the validity verification module, whether a website corresponding to the browser terminal initiating the connection request is a website included in a website white list.

16. The apparatus of claim 15, wherein the client terminal further comprises:

an update module configured to update the website white list by querying a server periodically.

17. The apparatus of claim 12, wherein the breakdown report processing module acquires the breakdown report by invoking a fault diagnostic program built in an operating system via an application programming interface of the operating system.

18. A system for implementing invocation of a local service component by a browser, comprising:

a browser terminal configured to initiate a connection request based on a cross-domain communication protocol to a client terminal; and

the client terminal configured to select a port from a port list for binding according to a

set port binding priority order upon receiving the connection request, perform validity verification of the connection request, and establish a communication connection with the browser terminal based on the cross-domain communication protocol to enable the browser terminal to invoke a local service component, if the connection request passes the validity verification.

19. The system of claim 18, further comprising:

a server configured to receive and store a breakdown report sent by the client terminal.