(12) STANDARD PATENT
(19) AUSTRALIAN PATENT OFFICE

(11) Application No. **AU 2017273371 B2**

(54) Title
Method and device for preventing server from being attacked

(51) International Patent Classification(s)
H04L 29/02 (2006.01)

(21) Application No: **2017273371** (22) Date of Filing: **2017.04.18**

(87) WIPO No: **WO17/206605**

(30) Priority Data

(31) Number	(32) Date	(33) Country
201610377847.0	**2016.05.31**	**CN**

(43) Publication Date: **2017.12.07**
(44) Accepted Journal Date: **2020.12.24**

(71) Applicant(s)
Advanced New Technologies Co., Ltd.

(72) Inventor(s)
Lu, Yaran

(74) Agent / Attorney
Spruson & Ferguson, Level 24 Tower 2 Darling Park 201 Sussex Street, Sydney, NSW, 2000, AU

(56) Related Art
US 20160119304 A1

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2017 年 12 月 7 日 (07.12.2017)

WIPO | PCT

(10) 国际公布号
WO 2017/206605 A1

(51) 国际专利分类号:
H04L 29/02 (2006.01)

(21) 国际申请号: PCT/CN2017/080862

(22) 国际申请日: 2017 年 4 月 18 日 (18.04.2017)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201610377847.0 2016年5月31日 (31.05.2016) CN

(71) 申请人: 阿里巴巴集团控股有限公司 (**ALIBABA GROUP HOLDING LIMITED**) [—/CN]; 开曼群岛大开曼资本大厦一座四层847号邮箱, Grand Cayman (KY)。

(72) 发明人; 及

(71) 申请人 (仅对*US*): 鲁亚然 (**LU, Yaran**) [CN/CN]; 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。

(74) 代理人: 北京国昊天诚知识产权代理有限公司 (**CO-HORIZON INTELLECTUAL PROPERTY INC.**); 中国北京市朝阳区小关北里甲2号渔阳置业大厦B座605, Beijing 100029 (CN)。

(81) 指定国 (除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY,

(54) **Title:** METHOD AND DEVICE FOR PREVENTING SERVER FROM BEING ATTACKED

(54) **发明名称**: 防止服务器被攻击的方法及装置

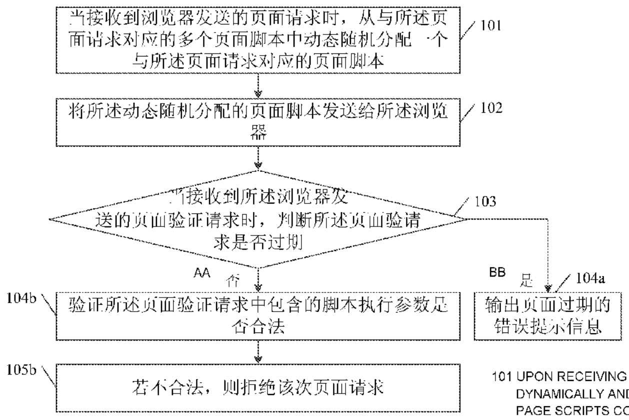
当接收到浏览器发送的页面请求时，从与所述页面请求对应的多个页面脚本中动态随机分配一个与所述页面请求对应的页面脚本
101
将所述动态随机分配的页面脚本发送给所述浏览器
102
当接收到所述浏览器发送的页面验证请求时，判断所述页面验证请求是否过期
103
AA
否
BB
是
104a
输出页面过期的错误提示信息
104b
验证所述页面验证请求中包含的脚本执行参数是否合法
105b
若不合法，则拒绝该次页面请求

图 1

101 UPON RECEIVING A PAGE REQUEST SENT BY A BROWSER, DYNAMICALLY AND RANDOMLY ALLOCATE, FROM MULTIPLE PAGE SCRIPTS CORRESPONDING TO THE PAGE REQUEST, A PAGE SCRIPT CORRESPONDING TO THE PAGE REQUEST
102 SEND THE DYNAMICALLY AND RANDOMLY ALLOCATED PAGE SCRIPT TO THE BROWSER
103 UPON RECEIVING A PAGE VERIFICATION REQUEST SENT BY THE BROWSER, DETERMINE WHETHER THE PAGE VERIFICATION REQUEST HAS EXPIRED
104A OUTPUT ERROR PROMPT INFORMATION INDICATING PAGE EXPIRATION
104B VERIFY WHETHER THE SCRIPT EXECUTION PARAMETER COMPRISED IN THE PAGE VERIFICATION REQUEST IS LEGAL
105B IF THE SCRIPT EXECUTION PARAMETER IS ILLEGAL, REJECT THE CURRENT PAGE REQUEST
AA NO
BB YES

(57) **Abstract:** The present invention relates to the technical field of network security, and discloses a method and device for preventing a server from being attacked, solving the problem of low server security. The main technical solution of the present invention comprises: upon receiving a page request sent by a browser, dynamically and randomly allocating, from multiple page scripts corresponding to the page request, a page script corresponding to the page request; sending the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter; upon receiving a page verification request

WO 2017/206605 A1

[见续页]

MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

(84) **指定国**(除另有指明，要求每一种可提供的地区保护)：ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布：

— 包括国际检索报告(条约第21条(3))。

sent by the browser, determining whether the page verification request is expired; if so, outputting error prompt information indicating page expiration; if not, verifying whether the script execution parameter comprised in the page verification request is legal; and if the script execution parameter is illegal, rejecting the current page request. The present invention mainly has application in preventing server attacks.

(57) 摘要：本发明公开了一种防止服务器被攻击的方法及装置，涉及网络安全技术领域，解决了服务器安全性低的问题。本发明的主要技术方案为：当接收到浏览器发送的页面请求时，从与所述页面请求对应的多个页面脚本中动态随机分配一个与所述页面请求对应的页面脚本；将所述动态随机分配的页面脚本发送给所述浏览器，以使得所述浏览器执行所述页面脚本获取脚本执行参数；当接收到所述浏览器发送的页面验证请求时，判断所述页面验证请求是否过期；若过期，则输出页面过期的错误提示信息；若没有过期，则验证所述页面验证请求中包含的脚本执行参数是否合法；若不合法，则拒绝该次页面请求。本发明主要用于防止服务器被攻击。

2017273371 13 Nov 2020

METHOD AND DEVICE FOR PREVENTING SERVER FROM BEING ATTACKED

TECHNICAL FIELD

[0001] The present disclosure relates to the field of network security technologies, and in particular, to a method and device for preventing a server from being attacked.

BACKGROUND

[0002] As Internet technologies rapidly develop, network security assurance attracts more attention. Generally, network security relates to how to prevent a server in the network from being attacked. To attack a server, an attacker uses a service request to occupy excessive service resources of the server, which leads to overload of the server. Moreover, the server cannot respond to other requests, and consequently the resources of the server can run out. As such, the attacker makes the server to refuse to provide services.

[0003] Currently, a browser first sends a service request to the server. The service request includes a token value in the cookie (data stored by the server on a local terminal device of a user) encrypted by using the message digest algorithm 5 (MD5). After receiving the service request, the server verifies the encrypted token value to determine whether the service request sent by the browser is valid to prevent the server from being attacked. However, the encrypted token value is obtained by executing static script code, and the static script code is exposed in a plaintext. Therefore, the attacker can directly obtain logic in the static script code and parse out the encryption method of the token value. As such, the attacker can skip a detection mechanism of the server by simulating the service request of the regular user and then attack the server. Therefore, security protection for the existing server is low.

SUMMARY

[0004] In view of the previous problem, the present disclosure is proposed to provide a method and device for preventing a server from being attacked, to overcome the previous problem or at least partially resolve the previous problem.

[0005] To achieve the previous objective, the present disclosure mainly provides the technical solutions below.

[0006] According to one aspect, an implementation of the present disclosure provides a method for preventing a server from being attacked, and the method includes the following: dynamically and randomly allocating a page script corresponding to a page request from a plurality of page scripts corresponding to the page request, when receiving the page request sent by a browser; sending the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter; determining whether the page verification request is expired, when receiving a page verification request sent by the browser; and if expired, outputting error prompt information indicating page expiration; or if unexpired, verifying whether the script execution parameter included in the page verification request is valid; and if invalid, rejecting the page request.

[0007] According to another aspect, an implementation of the present disclosure further provides a device for preventing a server from being attacked, and the device includes an allocation unit, configured to dynamically and randomly allocate a page script corresponding to a page request from a plurality of page scripts corresponding to the page request, when the page request sent by a browser is received; a sending unit, configured to send the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter; a determining unit, configured to determine whether a page verification request is expired, when the page verification request sent by the browser is received; an output unit, configured to output error prompt information indicating page expiration, if the page verification request is expired; a verification unit, configured to verify whether the script execution parameter included in the page verification request is valid, if the page verification request is not expired; and a rejection unit, configured to reject the page request if the script execution parameter included in the page verification request is invalid.

[0007a] According to another aspect, an implementation of the present disclosure further provides a method for preventing a server from being attacked, the method comprising: in response to receiving a page request sent by a browser, dynamically and randomly allocating a page script corresponding to the page request from a plurality of page scripts corresponding to the page request, comprising: obtaining a page URL in the page request; and randomly extracting a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL, wherein different script execution parameter acquisition logic is used in the plurality of page scripts; sending the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter; in response to receiving the page verification request sent by the browser, determining whether a page verification request is expired; in response to determining the page verification request is expired, outputting error prompt information indicating page expiration; or in response to determining that the page verification request is unexpired, verifying whether the script execution parameter comprised in the page verification request is valid; and if the script execution parameter is invalid, rejecting the page request.

[0007b] According to another aspect, an implementation of the present disclosure further provides a device for preventing a server from being attacked, the device comprising a plurality of modules configured to perform a method comprising: in response to receiving a page request sent by a browser, dynamically and randomly allocating a page script corresponding to the page request from a plurality of page scripts corresponding to the page request, comprising: obtaining a page URL in the page request; and randomly extracting a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL, wherein different script execution parameter acquisition logic is used in the plurality of page scripts; sending the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter; in response to receiving the page verification request sent by the browser, determining whether a page verification request is expired; in response to determining the page verification request is expired, outputting error prompt information indicating page expiration; or in response to determining that the page verification request is unexpired, verifying whether the script execution parameter comprised in the page verification request is valid; and if the script execution parameter is invalid, rejecting the page request.

[0008] Based on the above, the technical solutions provided in the implementations of the present disclosure have at least the advantages below.

According to the method and device for preventing a server from being attacked provided in the implementations of the present disclosure, when receiving the page verification request sent by the browser, the server first determines whether the page verification request is expired. If the page verification request is unexpired, the server verifies whether the script execution parameter included in the page verification request is valid. If the script execution parameter is invalid, the server rejects the page request to prevent the server from being attacked. Currently, the server verifies an encrypted token value to determine whether a service request sent by the browser is valid to prevent the server from being attacked. Compared with this, in the present disclosure, the server verifies the script execution parameter to prevent the server from being attacked. The script execution parameter is obtained based on the page script dynamically and randomly allocated from the plurality of page scripts corresponding to the page request, and different script execution parameter acquisition logic is used in the page scripts. Therefore, even if an attacker obtains script logic in dynamic code, the attacker cannot parse out the encryption method of the script execution parameter within a predetermined time period. In addition, when the request time of the page verification request exceeds the predetermined time period, the server rejects the page request. The browser needs to reload a page verification request to send the page verification request again, and a script execution parameter in the reloaded page verification request is obtained by executing a re-extracted page script. Therefore, the attacker cannot attack the server in the implementations of the present disclosure. As such, security of the server is improved in the implementations of the present disclosure.

BRIEF DESCRIPTION OF DRAWINGS

[0009] By reading the detailed descriptions of the following preferred implementations, various other advantages and benefits can be understood by a person of ordinary skill in the art. The accompanying drawings are only used for the purpose of illustrating the preferred implementations, and are not considered as a limitation on the present disclosure. In addition, the same reference symbols are used to represent the same components throughout the accompanying drawings. In the accompanying drawings:

[0010] FIG. 1 is a flowchart illustrating a method for preventing a server from being attacked, according to an implementation of the present disclosure;

[0011] FIG. 2 is a flowchart illustrating another method for preventing a server from being attacked, according to an implementation of the present disclosure;

[0013] FIG. 3 is a composition block diagram illustrating a device for preventing a server from being attacked, according to an implementation of the present disclosure;

[0014] FIG. 4 is a composition block diagram illustrating another device for preventing a server from being attacked, according to an implementation of the present disclosure; and

[0015] FIG. 5 is a block diagram illustrating a system for preventing a server from being attacked, according to an implementation of the present disclosure.

DESCRIPTION OF IMPLEMENTATIONS

[0016] The following describes the example implementations of the present disclosure in more detail with reference to the accompanying drawings. Although the accompanying drawings show the example implementations of the present disclosure, it should be understood that the present disclosure can be implemented in various forms, and shall not be limited by the implementations described here. Instead, these implementations are provided to provide a more thorough understanding of the present disclosure and to completely convey the scope of the present disclosure to a person of ordinary skill in the art.

[0017] To make the advantages of the technical solutions in the present disclosure clearer, the following describes the present disclosure in detail with reference to the accompanying drawings and the implementations.

[0018] An implementation of the present disclosure provides a method for preventing a server from being attacked. As shown in FIG. 1, the method includes the steps below.

[0019] 101. Dynamically and randomly allocate a page script corresponding to the page request from a plurality of page scripts corresponding to the page request, when receiving a page request sent by a browser.

[0020] The page request includes a page URL, and the page URL is a page that is corresponding to the page script requested by the browser. In this implementation of the present disclosure, one page URL is corresponding to a plurality of page scripts. Different script execution parameter acquisition logic is used in page scripts corresponding to the same page URL. A plurality of different script execution parameters are obtained when a plurality of page scripts corresponding to the same page URL are executed. Page execution results of the plurality of page scripts corresponding to the same page URL are the same, that is, pages generated after the browser loads and executes the page scripts are the same. It is worthwhile to note that in this implementation of the present disclosure, the page script corresponding to the page request can be dynamically and randomly allocated from the plurality of page scripts corresponding to the page request by using a random number generated based on a current time or a random number generated based on a time for sending the page request. No specific limitation is imposed in this implementation of the present disclosure.

[0021] For example, if a user enters a URL of AMAZON in an address bar of the browser and presses an enter key, the server receives a page request sent by the browser. A page URL in the page request is AMAZON. Then, the server dynamically and randomly allocates a page script corresponding to the page request from a plurality of page scripts corresponding to the page request. That is, the server dynamically and randomly allocates a page script corresponding to the page request from a plurality of page scripts corresponding to AMAZON.

[0022] 102. Send the dynamically and randomly allocated page script to the browser.

[0023] Further, the browser loads and executes the page script to obtain a script execution parameter. After loading and executing the page script, the browser displays a corresponding page and obtains the script execution parameter. Script execution parameters are some additional parameters on the page and do not affect the displayed page. In this implementation of the present disclosure, when page scripts corresponding to the same page URL are executed, different script execution parameters are generated in different page script execution results.

[0024] For example, a certain page URL is corresponding to three page scripts. Script execution parameter acquisition logic in a page script is to obtain an encrypted token value in cookie information. Script execution parameter acquisition logic in another page script is to obtain encrypted values of current mouse coordinates of the user. Script execution parameter acquisition logic in the remaining page script is to obtain an encrypted value of a current time. It is worthwhile to note that in this implementation of the present disclosure, logic for executing the page script and the encryption method of the script execution parameter are not limited, provided that different script execution parameter acquisition logic is used in the page scripts in the present disclosure to distinguish between different page scripts.

[0025] 103. Determine whether a page verification request is expired, when receiving the page verification request sent by the browser.

[0026] The page verification request can be sent by using the page obtained based by loading and executing in step 102. Specifically, the request can be sent by clicking a certain link on the page, selecting a certain command button, entering some keywords, etc. No specific limitation is imposed in this implementation of the present disclosure. The page verification request includes the script execution parameter, and the script execution parameter is an encrypted parameter.

[0027] In this implementation of the present disclosure, a specific process of determining whether the page verification request is expired can be as follows: First, the server obtains a request time of the page verification request and a time for the browser to execute the page script from the page verification request. Then, the server determines whether the request time of the page verification request is later than the sum of a predetermined time period and the time for the browser to execute the page script. The server determines that the page verification request is expired, if the request time of the page verification request is later than the sum of the predetermined time period and the time for the browser to execute the page script. The server determines that the page verification request is not expired, if the request time of the page verification request is not later than the sum of the predetermined time period and the time for the browser to execute the page script.

[0028] The predetermined time period can be set based on actual requirements, for example, 10 minutes, 20 minutes, or 40 minutes. No limitation is imposed in this implementation of the present disclosure. It is worthwhile to note that an attacker can attack the server by simulating behavior of the regular user, and it takes time for the attacker to simulate the behavior of the regular user. Therefore, in this implementation of the present disclosure, under the condition that the user can send a request to the server normally, the shorter the predetermined time period, the less likely the attacker is to attack the server.

[0029] For example, if the request time of the page verification request obtained from the page verification request is 13:15, the time for the browser to execute the page script is 12:34, and the predetermined time period is one hour, the server determines that the request time of the page verification request is earlier than the sum of the predetermined time period and the time for the browser to execute the page script, that is, 13:15 is earlier than 13:34. Hence the page verification request is not expired. If the request time of the page verification request is 14:12, the server can use the previous determining method to conclude that the page verification request has expired.

[0030] 104a. If expired, output error prompt information indicating page expiration.

[0031] In this implementation of the present disclosure, the error prompt information indicating page expiration is output if the page verification request has expired, to prompt the user with a fact that the current page has expired. If the user still wants to perform an operation on the page, the user needs to refresh the page. Refreshing the page is equivalent to sending a page request to the server. After receiving the request, the server dynamically and randomly allocates a page script corresponding to the page request from a plurality of page scripts corresponding to the page request. That is, after step 104a, if receiving a refresh instruction from the user, the server jumps to step 101 to perform step 101 again.

[0032] 104b. If not expired, verify whether a script execution parameter included in the page verification request is valid.

[0033] Step 104b is parallel to step 104a. The server verifies whether the script execution parameter included in the page verification request is valid, if the page verification request is not expired. In this implementation of the present disclosure, the server can verify, based on the page script executed to obtain the script execution parameter, whether the script execution parameter is valid. A specific process of step 104b can be as follows: First, the server obtains a local script parameter based on the page script, and then determines whether the local script parameter is the same as the script execution parameter included in the page verification request; if yes, it indicates that the script execution parameter is valid; if no, it indicates that the script execution parameter is invalid.

[0034] For example, after the page verification request sent by the browser is received, if script execution parameter acquisition logic in the page script dynamically and randomly extracted from the plurality of page scripts corresponding to the page request is to obtain a token value in cookie information that is encrypted by using a message digest algorithm 5 (MD5), and the page verification request is in a validity period, the server determines, based on the script execution parameter acquisition logic in the page script, whether verification on the script parameter can succeed. That is, the server determines, based on the MD5 encrypted value of the token in the corresponding cookie information stored in the server, whether verification on the script parameter sent by the browser can succeed.

[0035] 105b. If invalid, reject the page request.

[0036] In this implementation of the present disclosure, the page request is rejected if the script execution parameter is invalid after verification. Afterwards, if the user still wants to perform an operation on the page, the user needs to refresh the page. Refreshing the page is equivalent to sending a page request to the server. After receiving the request, the server dynamically and randomly allocates a page script corresponding to the page request from a plurality of page scripts corresponding to the page request. That is, after step 105b, if receiving a refresh instruction from the user, the server jumps to step 101 to perform step 101 again.

[0037] It is worthwhile to note that in the present disclosure, the page request can be rejected by using user identity information in the page verification request, and the user identity information is cookie information generated by the server. After receiving the cookie information sent by the server, the browser stores a key/value in the cookie information into a text file in a certain directory. The browser sends the cookie information to the server when requesting a webpage next time. If the page verification request sent by the browser has expired, or the script execution parameter is invalid after verification, the server can set a service corresponding to the cookie information in the page verification request to "reject" so as to reject the page request.

[0038] In the method for preventing a server from being attacked provided in this implementation of the present disclosure, when receiving the page verification request sent by the browser, the server first determines whether the page verification request is expired. If the page verification request is not expired, the server verifies whether the script execution parameter included in the page verification request is valid. If the script execution parameter is invalid, the server rejects the page request to prevent the server from being attacked. Currently, the server verifies an encrypted token value to determine whether a service request sent by the browser is valid to prevent the server from being attacked. Compared with this, in the present disclosure, the server verifies the script execution parameter to prevent the server from being attacked. The script execution parameter is obtained based on the page script dynamically and randomly allocated from the plurality of page scripts corresponding to the page request, and different script execution parameter acquisition logic is used in the page scripts. Therefore, even if an attacker obtains script logic in dynamic code, the attacker cannot parse out the encryption method of the script execution parameter within the predetermined time period. In addition, when the request time of the page verification request exceeds the predetermined time period, the server rejects the page request. The browser needs to reload a page verification request to send the page verification request again, and a script execution parameter in the reloaded page verification request is obtained by executing a re-extracted page script. Therefore, the attacker cannot attack the server in this implementation of the present disclosure. As such, security of the server is improved in this implementation of the present disclosure.

[0039] An implementation of the present disclosure provides another method for preventing a server from being attacked. As shown in FIG. 2, the method includes the steps below.

[0040] 201. Obtain a page URL in a page request when receiving the page request sent by a browser.

[0041] 202. Randomly extract a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL.

[0042] Different script execution parameter acquisition logic is used in the plurality of page scripts. In this implementation of the present disclosure, before step 202, the method further includes configuring page scripts corresponding to each page URL in the predetermined script library. The predetermined script library stores a plurality of page scripts respectively corresponding to different page URLs. Different script parameter acquisition logic is used in page scripts corresponding to each page URL. In this implementation of the present disclosure, the page script is used to execute and load a page that the browser requests, and obtain a script execution parameter from an execution result. When page scripts corresponding to the same page URL are executed, different script execution parameters are generated from executing different page scripts.

[0043] It is worthwhile to note that, because the only difference between the page scripts is that different script execution parameter acquisition logic is used, pages generated by loading and executing the page scripts are the same. The only difference from execution of the page scripts is that the script execution parameters generated are different.

[0044] 203. Send the dynamically and randomly allocated page script to the browser.

[0045] Further, the browser executes the page script to obtain a script execution parameter. After loading and executing the page script, the browser displays a corresponding page and obtains the script execution parameter. Script execution parameters are some additional parameters on the page and do not affect the displayed page.

[0046] 204. Determine whether a page verification request is expired, when receiving the page verification request sent by the browser.

[0047] In this implementation of the present disclosure, step 204 includes determining whether a request time of the page verification request is later than the sum of a predetermined time period and a time for the browser to execute the page script; and if yes, determining that the page verification request is expired; or if no, determining that the page verification request is not expired. The predetermined time period is used to limit a time when the browser can send the page request to the server, and the predetermined time period can be set based on actual requirements.

[0048] It is worthwhile to note that an attacker can attack the server by simulating behavior of the regular user, and it takes time for the attacker to simulate the behavior of the regular user. Therefore, in this implementation of the present disclosure, under the condition that the user can send a request to the server normally, the shorter the predetermined time period, the less likely the attacker is to attack the server.

[0049] 205a. If expired, output error prompt information indicating page expiration.

[0050] 205b. If not expired, verify whether a script execution parameter included in the page verification request is valid.

[0051] Step 205b is parallel to step 205a. If the page verification request is not expired, the server verifies whether the script execution parameter included in the page verification request is valid. In this implementation of the present disclosure, the page request further includes identifier information of the page script executed by the browser. Step 205b includes the following: searching the predetermined script library for a page script corresponding to the identifier information, where the predetermined script library further stores identifier information corresponding to each page script; and verifying, based on the page script corresponding to the identifier information, whether the script execution parameter included in the page verification request is valid. The page script executed by the browser is the page script sent by the server to the browser in step 203. When sending the page script to the browser, the server also sends the identifier information corresponding to the page script to the browser. When sending the page verification request to the server, the browser also sends the identifier information of the page script to the server. Then, the server obtains a corresponding page script from the predetermined script library based on the identifier information of the page script, and verifies, based on the obtained page script, whether the script execution parameter is correct.

[0052] In this implementation of the present disclosure, the verifying, based on the page script corresponding to the identifier information, whether the script execution parameter included in the page verification request is valid includes obtaining a local script parameter based on the page script corresponding to the identifier information; determining whether the local script parameter is the same as the script execution parameter included in the page verification request; and if yes, determining that the script execution parameter is valid; or if no, determining that the script execution parameter is invalid.

[0053] 206b. If invalid, reject the page request.

[0054] In this implementation of the present disclosure, if the script execution parameter is invalid after verification, the page request is rejected. Afterwards, if the user still wants to perform an operation on the page, the user needs to refresh the page. Refreshing the page is equivalent to sending a page request to the server. After receiving the request, the server dynamically and randomly allocates a page script corresponding to the page request from a plurality of page scripts corresponding to the page request. That is, after rejecting the page request, if receiving a refresh instruction from the user, the server jumps to step 201 to perform step 201 again, and randomly re-extracts a page script after performing step 201 again. Therefore, the attacker cannot attack the server in this implementation of the present disclosure. As such, security of the server is improved in this implementation of the present disclosure.

[0055] In this implementation of the present disclosure, a scenario shown in FIG. 5 can be applied, but is not limited to this. The scenario includes the following: In step 1 in FIG. 5, the browser first sends the page request to the server. The page request includes the page URL. After receiving the page request, the server randomly extracts the page script corresponding to the page URL from the predetermined script library, and then sends the page script to the browser. That is, the server sends the randomly extracted page script to the browser by using step 2 in FIG. 5. After receiving the page script sent by the server, the browser executes the page script and obtains the script execution parameter obtained by executing the page script. Then, the browser sends the page verification request to the server. The page verification request includes the script execution parameter. That is, the browser sends the page verification request to the server by using step 3 in FIG. 5. After receiving the page verification request, the server first determines whether the time of the page verification request is later than the sum of the predetermined time period and the time for the browser to execute the page script. The server verifies whether the script execution parameter is valid, if the time of the page verification request is earlier than the sum of the predetermined time period and the time for the browser to execute the page script. If invalid, the server rejects the page request. The script execution parameter in the present disclosure is obtained by executing the page script that is corresponding to the page URL and is randomly extracted from the predetermined script library, and different script execution parameter acquisition logic is used in the page scripts. Therefore, even if an attacker obtains script logic in dynamic code, the attacker cannot parse out the encryption method of the script execution parameter within the predetermined time period. In addition, when the page verification request is expired, the browser needs to reload a page verification request, and a script execution parameter in the reloaded page verification request is obtained by executing a re-extracted page script corresponding to the page URL. Therefore, the attacker cannot attack the server in this implementation of the present disclosure. As such, security of the server is improved in this implementation of the present disclosure.

[0056] In another method for preventing a server from being attacked provided in this implementation of the present disclosure, when receiving the page verification request sent by the browser, the server first determines whether the page verification request is expired. If the page verification request is not expired, the server verifies whether the script execution parameter included in the page verification request is valid. If the script execution parameter is invalid, the server rejects the page request to prevent the server from being attacked. Currently, the server verifies an encrypted token value to determine whether a service request sent by the browser is valid to prevent the server from being attacked. Compared with this, in the present disclosure, the server verifies the script execution parameter to prevent the server from being attacked. The script execution parameter is obtained based on the page script dynamically and randomly allocated from the plurality of page scripts corresponding to the page request, and different script execution parameter acquisition logic is used in the page scripts. Therefore, even if an attacker obtains script logic in dynamic code, the attacker cannot parse out the encryption method of the script execution parameter within the predetermined time period. In addition, when the request time of the page verification request exceeds the predetermined time period, the server rejects the page request. The browser needs to reload a page verification request to send the page verification request again, and a script execution parameter in the reloaded page verification request is obtained by executing a re-extracted page script. Therefore, the attacker cannot attack the server in this implementation of the present disclosure. As such, security of the server is improved in this implementation of the present disclosure.

[0057] Further, an implementation of the present disclosure provides a device for preventing a server from being attacked. As shown in FIG. 3, the device includes an allocation unit 31, a sending unit 32, a determining unit 33, an output unit 34, a verification unit 35, and a rejection unit 36.

[0058] The allocation unit 31 is configured to dynamically and randomly allocate a page script corresponding to a page request from a plurality of page scripts corresponding to the page request, when the page request sent by a browser is received.

[0059] The sending unit 32 is configured to send the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter.

[0060] The determining unit 33 is configured to determine whether a page verification request is expired, when the page verification request sent by the browser is received.

[0061] The output unit 34 is configured to output error prompt information indicating page expiration if expired.

[0062] The verification unit 35 is configured to verify whether the script execution parameter included in the page verification request is valid, if the page verification request is not expired.

[0063] The rejection unit 36 is configured to reject the page request if the script execution parameter included in the page verification request is invalid.

[0064] It is worthwhile to note that, for other corresponding descriptions of the function units in the device for preventing a server from being attacked in this implementation of the present disclosure, reference can be made to corresponding descriptions of the method shown in FIG. 1. Details are omitted here for simplicity. However, it should be clear that the device in this implementation can correspondingly implement all the content in the method implementation.

[0065] According to the method and device for preventing a server from being attacked provided in the implementations of the present disclosure, when receiving the page verification request sent by the browser, the server first determines whether the page verification request is expired. If the page verification request is not expired, the server verifies whether the script execution parameter included in the page verification request is valid. If the script execution parameter is invalid, the server rejects the page request to prevent the server from being attacked. Currently, the server verifies an encrypted token value to determine whether a service request sent by the browser is valid to prevent the server from being attacked. Compared with this, in the present disclosure, the server verifies the script execution parameter to prevent the server from being attacked. The script execution parameter is obtained based on the page script dynamically and randomly allocated from the plurality of page scripts corresponding to the page request, and different script execution parameter acquisition logic is used in the page scripts. Therefore, even if an attacker obtains script logic in dynamic code, the attacker cannot parse out the encryption method of the script execution parameter within a predetermined time period. In addition, when a request time of the page verification request exceeds the predetermined time period, the server rejects the page request. The browser needs to reload a page verification request to send the page verification request again, and a script execution parameter in the reloaded page verification request is obtained by executing a re-extracted page script. Therefore, the attacker cannot attack the server in this implementation of the present disclosure. As such, security of the server is improved in this implementation of the present disclosure.

[0066] Further, an implementation of the present disclosure provides another device for preventing a server from being attacked. As shown in FIG. 4, the device includes an allocation unit 41, a sending unit 42, a determining unit 43, an output unit 44, a verification unit 45, and a rejection unit 46.

[0067] The allocation unit 41 is configured to dynamically and randomly allocate a page script corresponding to a page request from a plurality of page scripts corresponding to the page request, when the page request sent by a browser is received.

[0068] The sending unit 42 is configured to send the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter.

[0069] The determining unit 43 is configured to determine whether a page verification request is expired, when the page verification request sent by the browser is received.

[0070] The output unit 44 is configured to output error prompt information indicating page expiration, if the page verification request is expired.

[0071] The verification unit 45 is configured to verify whether the script execution parameter included in the page verification request is valid, if the page verification request is not expired.

[0072] The rejection unit 46 is configured to reject the page request if the script execution parameter included in the page verification request is invalid.

[0073] Further, the allocation unit 41 includes an obtaining module 411, configured to obtain a page URL in the page request; and an extraction module 412, configured to randomly extract a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL, where different script execution parameter acquisition logic is used in the plurality of page scripts.

[0074] Further, the determining unit 43 includes a determining module 431, configured to determine whether a request time of the page verification request is later than the sum of a predetermined time period and a time for the browser to execute the page script; and a determining module 432, configured to determine that the page verification request is expired, if the request time of the page verification request is later than the sum of the predetermined time period and the time for the browser to execute the page script; or a determining module 432, configured to determine that the page verification request is not expired, if the request time of the page verification request is not later than the sum of the predetermined time period and the time for the browser to execute the page script.

[0075] In this implementation of the present disclosure, the page request further includes identifier information of the page script executed by the browser, and the verification unit 45 includes a searching module 451, configured to search the predetermined script library for a page script corresponding to the identifier information, where the predetermined script library further stores identifier information corresponding to each page script; and a verification module 452, configured to verify, based on the page script corresponding to the identifier information, whether the script execution parameter included in the page verification request is valid.

[0076] In this implementation of the present disclosure, the verification module 452 is configured to obtain a local script parameter based on the page script corresponding to the identifier information.

[0077] The verification module 452 is configured to determine whether the local script parameter is the same as the script execution parameter included in the page verification request.

[0078] The verification module 452 is configured to determine that the script execution parameter is valid, if the local script parameter is the same as the script execution parameter included in the page verification request.

[0079] The verification module 452 is configured to determine that the script execution parameter is invalid, if the local script parameter is different from the script execution parameter included in the page verification request.

[0080] The device further includes a configuration unit 47, configured to configure page scripts corresponding to each page URL in the predetermined script library.

[0081] It is worthwhile to note that, for other corresponding descriptions of the function units in another device for preventing a server from being attacked in this implementation of the present disclosure, reference can be made to corresponding descriptions of the method shown in FIG. 2. Details are omitted here for simplicity. However, it should be clear that the device in this implementation can correspondingly implement all the content in the method implementation.

[0082] According to the method and device for preventing a server from being attacked provided in the implementations of the present disclosure, when receiving the page verification request sent by the browser, the server first determines whether the page verification request is expired. If the page verification request is not expired, the server verifies whether the script execution parameter included in the page verification request is valid. If the script execution parameter is invalid, the server rejects the page request to prevent the server from being attacked. Currently, the server verifies an encrypted token value to determine whether a service request sent by the browser is valid to prevent the server from being attacked. Compared with this, in the present disclosure, the server verifies the script execution parameter to prevent the server from being attacked. The script execution parameter is obtained based on the page script dynamically and randomly allocated from the plurality of page scripts corresponding to the page request, and different script execution parameter acquisition logic is used in the page scripts. Therefore, even if an attacker obtains script logic in dynamic code, the attacker cannot parse out the encryption method of the script execution parameter within the predetermined time period. In addition, when the request time of the page verification request exceeds the predetermined time period, the server rejects the page request. The browser needs to reload a page verification request to send the page verification request again, and a script execution parameter in the reloaded page verification request is obtained by executing a re-extracted page script. Therefore, the attacker cannot attack the server in this implementation of the present disclosure. As such, security of the server is improved in this implementation of the present disclosure.

[0083] The device for preventing a server from being attacked includes a processor and a memory. The allocation unit, the sending unit, the determining unit, the output unit, the verification unit, the rejection unit, and the configuration unit are stored in the memory as program units. The processor executes the program units stored in the memory to implement corresponding functions.

[0084] The processor includes kernel, and the kernel invokes a corresponding program unit from the memory. There can be one or more kernels to improve security of the server by adjusting a kernel parameter.

[0085] The memory can include a non-persistent storage, a random access memory (RAM), and/or a nonvolatile memory in a computer readable medium, for example, a read-only memory (ROM) or a flash memory (flash RAM). The memory includes at least one storage chip.

[0086] The present disclosure further provides a computer program product, and when being executed on a data processing apparatus, the product is applicable to initialize the program code that includes the following steps: when receiving a page request sent by a browser, dynamically and randomly allocating a page script corresponding to the page request from a plurality of page scripts corresponding to the page request; sending the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter; when receiving a page verification request sent by the browser, determining whether the page verification request is expired; and if expired, outputting error prompt information indicating page expiration; or if not expired, verifying whether the script execution parameter included in the page verification request is valid; and if invalid, rejecting the page request.

[0087] A person of ordinary skill in the art should understand that the implementations of the present disclosure can be provided as a method, a system, or a computer program product. Therefore, the present disclosure can use a form of hardware only implementations, software only implementations, or implementations with a combination of software and hardware. In addition, the present disclosure can use a form of a computer program product that is implemented on one or more computer-usable storage media (including but not limited to a magnetic disk storage, a CD-ROM, an optical memory, etc.) that include computer-usable program code.

[0088] The present disclosure is described with reference to the flowcharts and/or block diagrams of the method and device for preventing a server from being attacked and the computer program product, according to the implementations of the present disclosure. It should be understood that computer program instructions can be used to implement each process and/or each block in the flowcharts and/or the block diagrams and a combination of a process and/or a block in the flowcharts and/or the block diagrams. These computer program instructions can be provided for a general-purpose computer, a dedicated computer, an embedded processor, or a processor of another programmable data processing device to generate a machine, so that the instructions executed by the computer or the processor of the another programmable data processing device generate an apparatus for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0089] These computer program instructions can be stored in a computer readable memory that can instruct the computer or another programmable data processing device to work in a specific way, so that the instructions stored in the computer readable memory generate an artifact that includes an instruction apparatus. The instruction apparatus implements a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0090] These computer program instructions can be loaded onto the computer or another programmable data processing device, so that a series of operations and steps are performed on the computer or the another programmable device, thereby generating computer-implemented processing. Therefore, the instructions executed on the computer or another programmable device provide steps for implementing a specific function in one or more processes in the flowcharts and/or in one or more blocks in the block diagrams.

[0091] In a typical configuration, a computing device includes one or more processors (CPU), an input/output interface, a network interface, and a memory.

[0092] The memory can include a non-persistent storage, a random access memory (RAM), and/or a nonvolatile memory in a computer readable medium, for example, a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of the computer readable medium.

[0093] The computer readable medium includes persistent, non-persistent, movable, and unmovable media that can store information by using any method or technology. The information can be a computer readable instruction, a data structure, a program module, or other data. A computer storage medium includes but is not limited to a phase-change random access memory (PRAM), a static random access memory

(SRAM), a dynamic random access memory (DRAM), another type of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or another optical storage, a magnetic cassette, a magnetic tape, a magnetic disk storage, another magnetic storage device, or any other non-transitory medium. The computer storage medium can be used to store information accessible by the computing device. Based on the definition in the present specification, the computer readable medium does not include transitory computer-readable media (transitory media), for example, a modulated data signal and carrier.

[0094] The previous descriptions are only implementations of the present disclosure, and are not intended to limit the present disclosure. A person of ordinary skill in the art can make various modifications and variations to the present disclosure. Any modifications, equivalent substitutions, or improvements made without departing from the spirit and principle of the present disclosure shall fall in the scope of the claims in the present disclosure.

2017273371 13 Nov 2020

CLAIMS

1. A method for preventing a server from being attacked, the method comprising:

in response to receiving a page request sent by a browser, dynamically and randomly allocating a page script corresponding to the page request from a plurality of page scripts corresponding to the page request, comprising:

obtaining a page URL in the page request; and

randomly extracting a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL, wherein different script execution parameter acquisition logic is used in the plurality of page scripts;

sending the dynamically and randomly allocated page script to the browser, so that the browser executes the page script to obtain a script execution parameter;

in response to receiving the page verification request sent by the browser, determining whether a page verification request is expired;

in response to determining the page verification request is expired, outputting error prompt information indicating page expiration; or

in response to determining that the page verification request is unexpired, verifying whether the script execution parameter comprised in the page verification request is valid; and

if the script execution parameter is invalid, rejecting the page request.

2. The method according to claim 1, wherein the script execution parameter acquisition logic comprises obtaining a token value in encrypted cookie information.

3. The method according to claim 1, wherein before randomly extracting a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL, the method further comprises:

configuring page scripts corresponding to each page URL in the predetermined script library and identifier information corresponding to the page scripts.

4. The method according to claim 1, wherein determining whether the page verification request is expired comprises:

determining whether a request time of the page verification request is later than the sum of a predetermined time period and a time for the browser to execute the page script; and in response to determining that the request time of the page verification request is later than the sum of a predetermined time period and a time for the browser to execute the page script, determining that the page verification request is expired; or

in response to determining that the request time of the page verification request is not later than the sum of a predetermined time period and time for the browser to execute the page script, determining that the page verification request is not expired.

5. The method according to any one of claims 1 to 4, wherein the page request further comprises identifier information of the page script executed by the browser, and wherein verifying whether the script execution parameter comprised in the page verification request is valid comprises:

searching the predetermined script library for a page script corresponding to the identifier information, wherein the predetermined script library further stores identifier information corresponding to each page script; and

verifying, based on the page script corresponding to the identifier information, whether the script execution parameter comprised in the page verification request is valid.

6. The method according to claim 5, wherein verifying, based on the page script corresponding to the identifier information, whether the script execution parameter comprised in the page verification request is valid comprises:

obtaining a local script parameter based on the page script corresponding to the identifier information;

determining whether the local script parameter is the same as the script execution parameter comprised in the page verification request; and

in response to determining that the local script parameter is the same as the script execution parameter comprised in the page verification request, determining that the script execution parameter is valid; or

in response to determining that the local script parameter is no the same as the script execution parameter comprised in the page verification request, determining that the script execution parameter is invalid.

7. The method according to any one of claims 1 to 6, wherein rejecting the page request can be based on a user identity information included in the page verification request.

8. The method according to claim 7, wherein the user identity information comprises a cookie information generated by the server.

9. A device for preventing a server from being attacked, the device comprising a plurality of modules configured to perform the method of any one of claims 1 to 8.

Advanced New Technologies Co., Ltd.
Patent Attorneys for the Applicant/Nominated Person
SPRUSON & FERGUSON

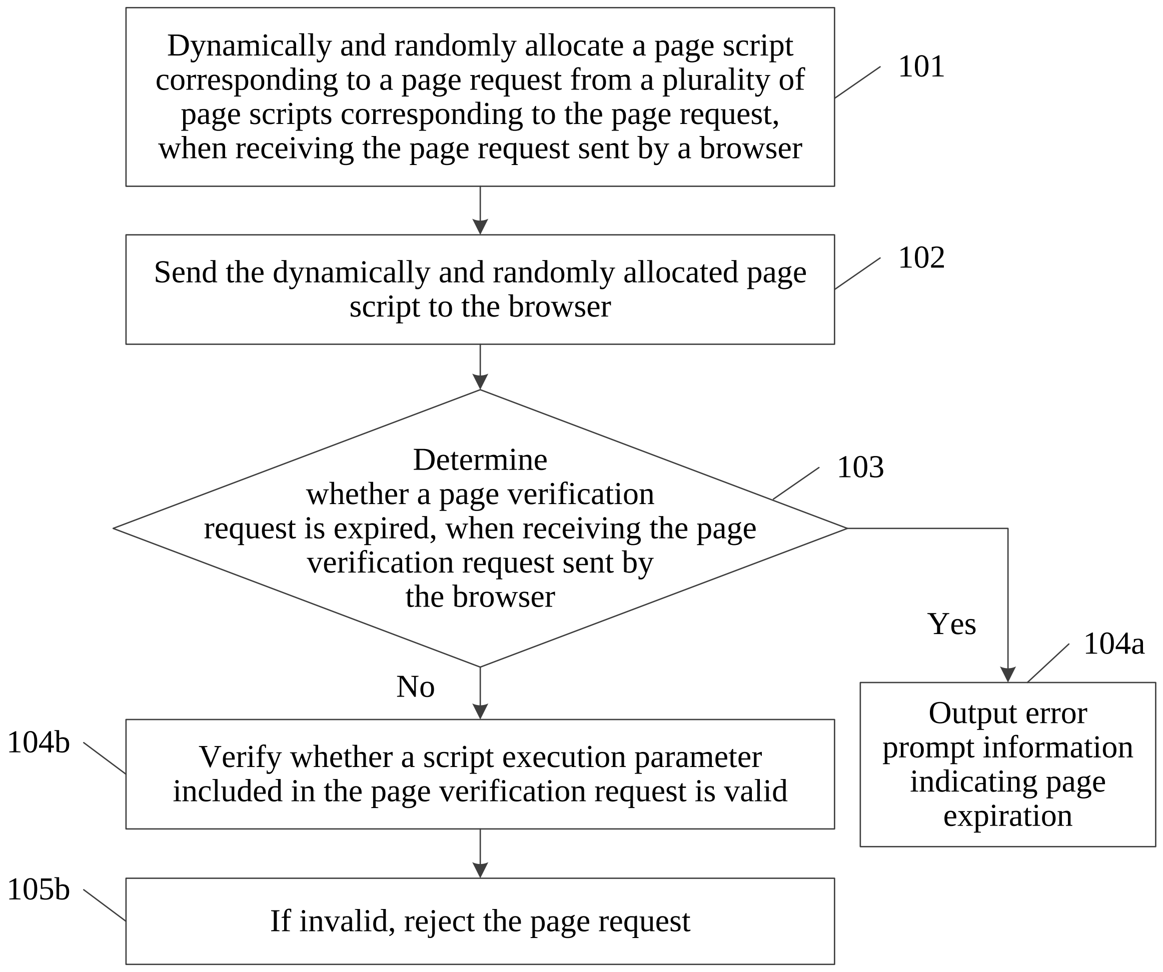
Dynamically and randomly allocate a page script corresponding to a page request from a plurality of page scripts corresponding to the page request, when receiving the page request sent by a browser
101
Send the dynamically and randomly allocated page script to the browser
102
Determine whether a page verification request is expired, when receiving the page verification request sent by the browser
103
Yes
104a
Output error prompt information indicating page expiration
No
104b
Verify whether a script execution parameter included in the page verification request is valid
105b
If invalid, reject the page request

FIG. 1

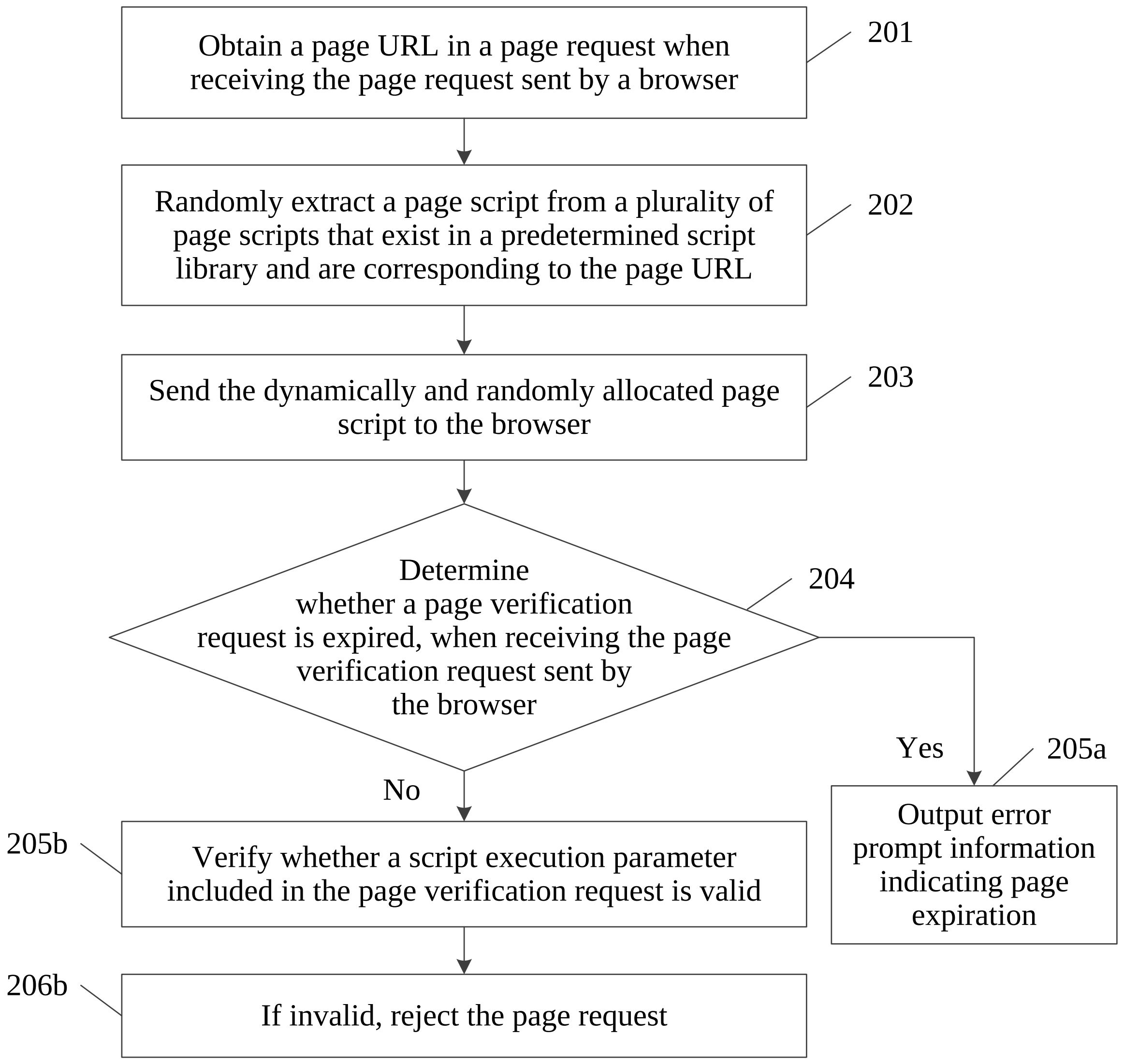
Obtain a page URL in a page request when receiving the page request sent by a browser
201
Randomly extract a page script from a plurality of page scripts that exist in a predetermined script library and are corresponding to the page URL
202
Send the dynamically and randomly allocated page script to the browser
203
Determine whether a page verification request is expired, when receiving the page verification request sent by the browser
204
Yes
205a
Output error prompt information indicating page expiration
No
205b
Verify whether a script execution parameter included in the page verification request is valid
206b
If invalid, reject the page request

FIG. 2

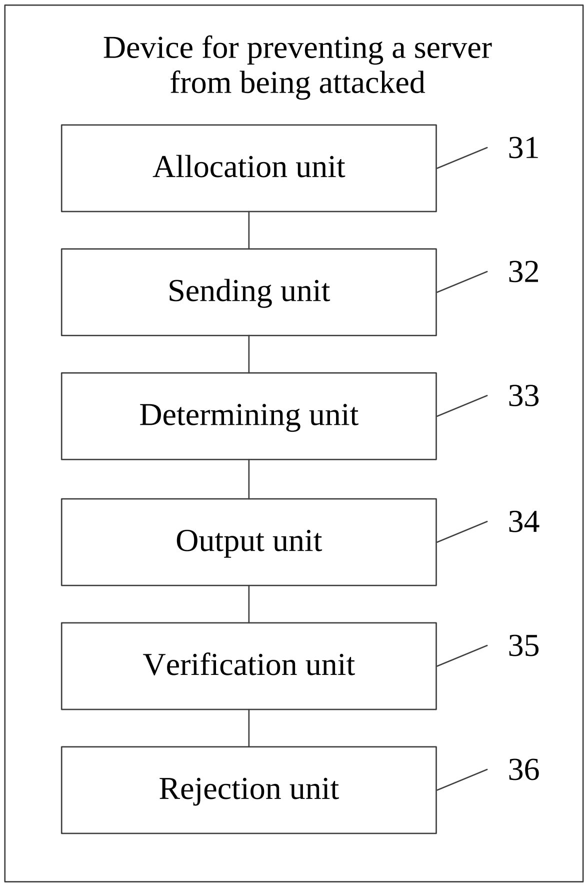
Device for preventing a server from being attacked
Allocation unit
31
Sending unit
32
Determining unit
33
Output unit
34
Verification unit
35
Rejection unit
36

FIG. 3

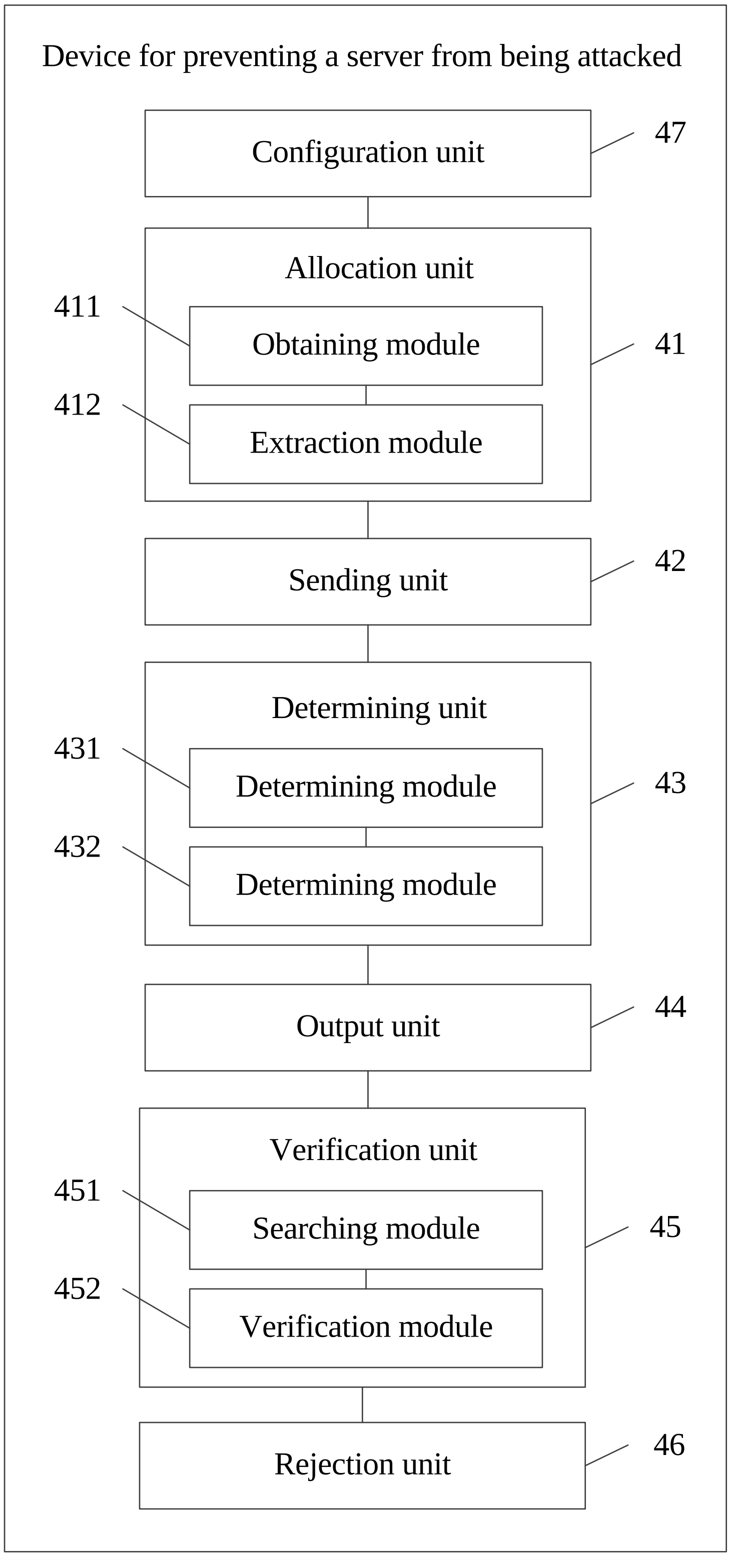
Device for preventing a server from being attacked
Configuration unit
47
Allocation unit
411
Obtaining module
41
412
Extraction module
Sending unit
42
Determining unit
431
Determining module
43
432
Determining module
Output unit
44
Verification unit
451
Searching module
45
452
Verification module
Rejection unit
46

FIG. 4

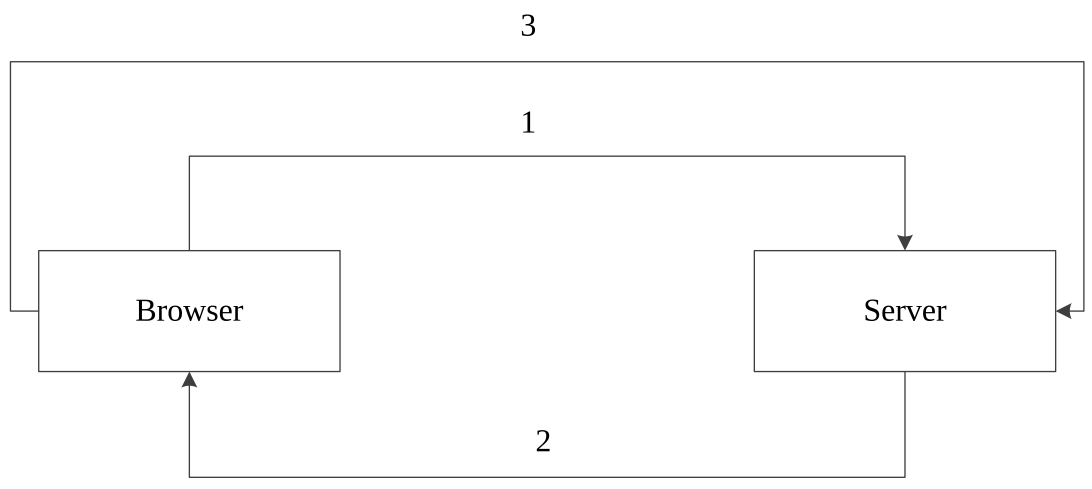

FIG. 5