Fine-grained Service Side Access Control Model for Web Application

Zhijun Liang, Hua Zhang, Zhonghua Zhao

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, 100029, China

Keywords: access control, extension, privilege management

Abstract. In the paper of SCUTA, to the more and more complicated network environment and the w3c's Same-Origin Policy's vulnerability, the author designs a new web access control model. It eases the difficulty of web policy's configuration and reduces the potential vulnerabilities. However, because of the inflexible model configuration design, it has low system flexibility. What's more, it also has many vulnerability. On the basis of it, we design a new model in detail, which was implemented and checked with Apache, PHP and Mysql. Relative to SCUTA model, this access control model can make the developer implements policy configuration easier, more efficient and more flexible avoiding the original vulnerabilities, which provides that this model can be more secure, more effective and faster.

Introduction

To this day, with the Internet technology development, our daily life can't leave the Internet totally. What followed by this are the many different types of attacks. The attacks are becoming hard to be taken because of its myriads of changes. Especially, there are many terrible vulnerabilities such as the Fish privilege permit and access control vulnerability [1], Xiao mi router privacy disk function bypassing access control vulnerability [2] and Adobe Reader privilege permit and access control vulnerability [3], which proves that under the seemingly simple daily Internet suffering, the security of the Internet makes its environment hid dangerous anytime anywhere and risk there and here. All of these make malicious damage. Now, the browser base on the Same-Origin Policy to control the method of how a origin load another origin's text, script or other resource. But it will grant no different privileges to all requests in the same session; the existed RBAC model can only manage the privileges of resource in a application and can't manage different origins' resources interactive.

To manage the requests among different origins and deal with the worse XSS or CSRF attacks, we design a fine-grained service side access control model for web application basing on the privilege level. This model, using the PHP extensions and added privileges judgement function accurate to function level in PHP core source code, is deployed on the server side. It solves the problem that how to manage the interactive privileges among different origins and provides powerful protection for the management of cross origin invoking.

This paper uses 3 parts to describe the design and implement of the model. The first part introduces the design of the fine-grained service side access control model. Secondly, the next part presents the implement of the fine-grained service side access control model, evaluates and compares the performance of the model with other models. In the finally part the paper makes a conclusion.

Model design

This paper is implemented in the PHP core. Because the PHP source code is opened so that in this paper we add extension in the PHP core to implement access control function. In Escudo [4] implemented a browser side access control model for user. But it can only supply two choices that it will permit the request from the client side or deny this request. Excluding this defect, Escudo's client side implemented access control model has good performance. This paper's client side use Escudo model's design to support the paper's server side implement. In SCUTA [5], it implement a server side access control model, but it must takes control in every PHP page and if you have decided the access level and which function you can invoke or database you can access, you can't change it anymore, only if you revise the source code. What's more, in SCUTA, the "gate" would give the privileges which can access to one higher function to all the tags in the same ring level. This is a more fatal vulnerability. In this paper, we design a new access control model which not only makes up the SCUTA's fatal vulnerability that it can't deal with the privileges by level but also fixes up the bug and vulnerability that we can't get a flexible server side access control.

In this paper, we integrate PHP core in Apache as a module of Apache. Apache works under Prefork MPM mode, which means that when Apache starts up it will fork multiple processes and waiting for the user's request. When the user's request get to Apache server, Apache will check the request. If the result is a PHP script, it will invoke PHP model and pass the invoke privileges to the model. The PHP model is working in the SAPI mode here. After the Apache start up, PHP would initialize models. When a user's request arrive, it will initialize request and entry a corresponding processing environment.

In this paper, the module design as figure 1:

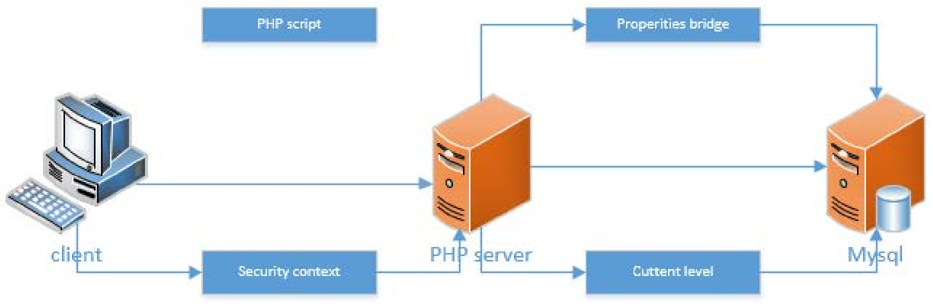

Fig.1 Model design

As this figure shows, this model implements some functions. When a user suffering on line and select some hyperlink, if he uses the ESCUDO browser, the browser would write the selected module's div tag's section in the cookie and send the request to the server side automatically. This request include this client side saved cookie and this cookie's section is corresponding to some block of the configuration file, which identifies one block's privileges ring level. In the pages, we generate a pair of random numbers at start and end of the div tag and check it at the server side to prevent user from tampering his right by forging a tag at client side and avoid crossing tags attacks.

In this paper's access control privileges ring level is decreasing starting from 0 to infinity. Which means that 0 is the biggest right and infinity is the smallest right. What's more, higher level functions can invoke lower level's functions but the lower level's functions can't invoke the higher level functions. Such as if one function in 0 level, it could invoke the functions in the level from 0 to infinity, but not the opposite. All the right configuration of functions and database's user configuration follow these rules in this paper.

At the server side, when the server get a request from client, the revised session module in the PHP core would intercept the cookie, extract the section name in the cookie and write it in the security context file. Then, if the selected block need invoke a PHP function, the engine would read current cookie's section name from security context file to decide which module is selected. After that, the revised PHP core engine would decide the module's corresponded ring level by the var-ring configuration field reading from the properties bridge configuration file and write the current running level into current level configuration file. At the last, basing on the right in the function-ring configuration field from the properties bridge file to decide whether the request could invoke the function to finish the last judgment. If the needed function have higher level, the server side would response with an error message. Otherwise, the server would invoke the function and pass the right to the invoked function.

For example, in this paper, we configure the server side's privileges as follows:

```
[var-ring]
sectionA=ring_0
sectionB=ring_1
sectionC=ring_2
[function-ring]
ring_0=writeMyName0,writeMyName01,writeMyName02
ring_1=writeMyName1,writeMyName11,writeMyName12,writeMyName13
ring_2=writeMyName2
[mysql-user-ring]
ring_0=dbuser_0
ring_1=dbuser_1
ring_2=dbuser_2
```

In this configuration file, we show that sectionA at 0 ring level and this module can invoke all the functions in the function-ring field. What's more, sectionB at 1 ring level and this module can only invoke the functions which in lower or equal to the ring level 1 (that is in the ring level 1 or 2). Finally, sectionC at 2 ring level and this module can only invoke the function whose name is writeMyName2. User named as dbuser_0 belong to 0 ring level so that when a function at 0 ring level accesses to the database, the logging user would replace the user whose name is dbuser_0, log in the database and get the right corresponding to the dbuser_0.

When we want to set any ring level to any Web page's module, we only need giving this module a section variable and configuring a corresponding ring level in the properties bridge file. This design increases the flexibility of configuration for the Web page privileges, what's more, makes it easy to manage privileges centralized, improves the efficiency of management and decreases the occurrence rate of vulnerabilities.

When you want to upgrade any function's privilege, you needn't granting a higher ring level to all the functions in the same ring level and just moving the function name you needed in the corresponding ring level. For example, if you want to upgrade the function whose name is writeMyName11 to 0 ring level, what you need to do is just move this function name to the 0 ring level and needn't grant 0 ring level to all the functions in the 1 ring level. When you want to change section's ring level, you can just do the same things and make corresponding change. All of these designs can prevent all the functions in the same ring level from upgrading their ring level when the SCUTA open a GAT to a function in the ring level and avoid the configuration files becoming fragmenting and complicating.

For those none security policy configured functions, Web pages and database tables, this model would process the requests as usual PHP processes Web page. So that, this model keeps a good backward compatibility.

Implementation and performance evaluation

Fine-grained service side access control model for web application can be implemented on many kinds of servers such as PHP, Java servlet and ASP.NET. In this paper, we select PHP as server whose source code is opened mainly because, in PHP, we can integrate extensions designed by us and revise the PHP's Zend Engine to implement some special function. We use Linux system and deployed PHP server, Apache server and Mysql database. Making Apache parse Web pages, we integrate, load and run PHP as a module of Aapache. In the background, we use Mysql to storage data, audit user's privileges and manage the logging user's access control.

PHP's source code divide into two parts as Zend Engine and Extension. The Zend Engine, as the core of PHP, implements the PHP's scripts parsing function. The Extension is PHP's plugin and extends the PHP's function. In the PHP Engine, we can revise some source code, rewrite some modules and integrate some kernel hooks to implements the access control auditing function.

Fine-grained service side access control model for web application has been divided into three parts. We modify three modules of PHP kernel that is session module, function invoke module and logging Mysql database's module.

In the session module, we revised the file named as session.c in the ext/session directory in the source code of PHP. We added the read and write cookies operation in the PHPAPI void php_session_start(TSRMLS_D) function. When client side sends a request, the Engine plugin interrupt cookie array first. And then, write all the cookies content in the file whose name is security context for later use. The module sketch shows as figure 2:

Fig. 2 Model design of session

We revised the static int ZEND_FASTCALL zend_do_fcall_common_helper_ SPEC(ZEND_ OPCODE_HANDLER_ARGS) function in the source code file whose name is Zend/zend_vm_ execute.h to implement the operation that read the cookie user's access field. Basing on the read access block to check the correspoonded ring level. If the request needs invoke PHP function, this plugin would find the functions which could be invoked basing on the corresponded ring level and makes a judgment for the function which will be invoked wheather grant the invoking privilege or prevernt the user from invoking it. The module sketch shows as figure 3:

If the request needs accessing the Mysql database in this process, the request would be passed to the PHP's Mysql extension. The static void PHP_Mysql_do_connect (INTERNAL_FUNCTION _PARAMETERS, int persistent) function in the file named as php_mysql.c in the ext/Mysql module of PHP source code implements the operation that read file whose name is current level, basing on the configuration file whose name is properties bridge, finds the corresponded database user name. Then it will replace the current database user with the corresponded user has corresponded privileges. After the user log in the database, the user's access control security would be guaranteed by the database. Parts of the code sketch shows as figure4:

We implements the database user's privileges granting function in the Mysql extension and at the last the configuration of privilegs show as follows:

dbUser_0: tableA, tableB, tableC
dbUser_1: tableB, tableC
dbUser_2: tableC

At last, we implements the fine-grained service side access control model for

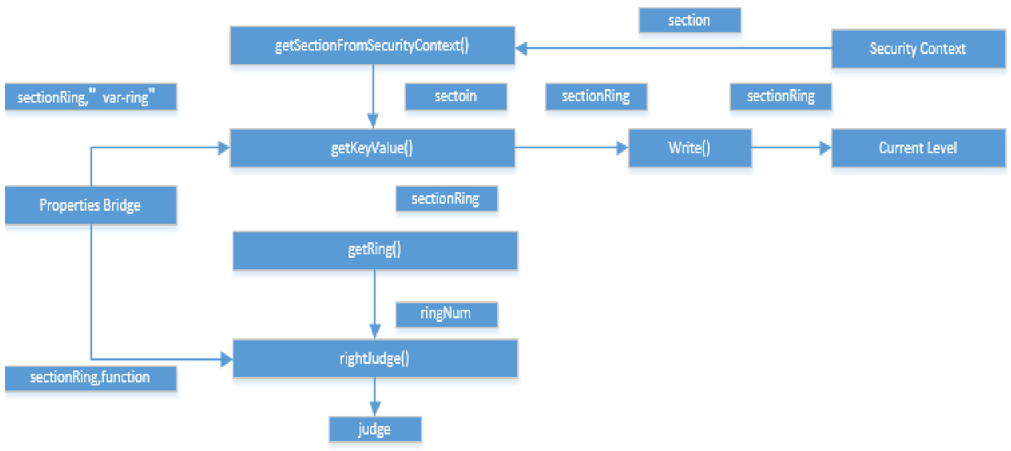

Fig. 3 Model design of function calling judge

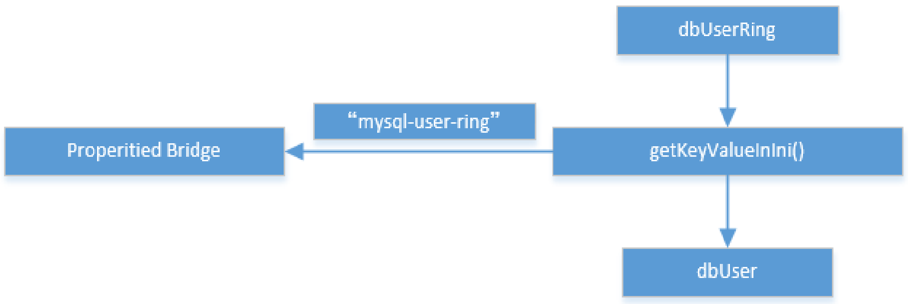

Fig. 4 Database calling judge model design

web application.

When a user configurates a PHP site with this access control model, its Web page could be configurated as follows:

```
<div sectionA nonce=n1 callwriteMyName0()"…>
</div nonce=n1>
<div sectionB nonce=n2 callwriteMyName1()"….>
        <div sectionC nonce=n3 callwriteMyName2()"…>
        </div nonce=n3>
</div nonce=n2>
```

In this page, sectionA, section and sectionC identify the names of the block of corresponded div tag. The attribute of nonce prevents users from forging tags and guaranteed the security of the Web page. When you click the sectionA hyperlink, the browser would send a request to server to access a.php and a.php would invoke the function whose name is writeMyName0() and take off the name from the database. Some parts of the a.php show as follows:

```
   writeMyName0(){
                mysql _connect("dbuser_0");
      mysql_query("select * from PHP_paper.tableA");
        mysql _connect("dbuser_1");
mysql_query("select * from PHP_paper.tableB");
        }
   writeMyName0();
```

We used Apache's benchmarking to evaluate the performance of every modules of fine-grained service side access control model for web application and get the result as figure5 shows:

As figure 5 shows, after revising, each module's processing time increased almost three present and it is very little. In all the three modules, Session module interrupt cookies and write those cookies into security context's time increased 3.7%. PHP Zend Engine module processes the judgment of invoked functions' privileges and the handling time increased 2.5%. Database's corresponded ring level user login time increased 3.5%. Thus, it can be seen that the increased cost time is little.

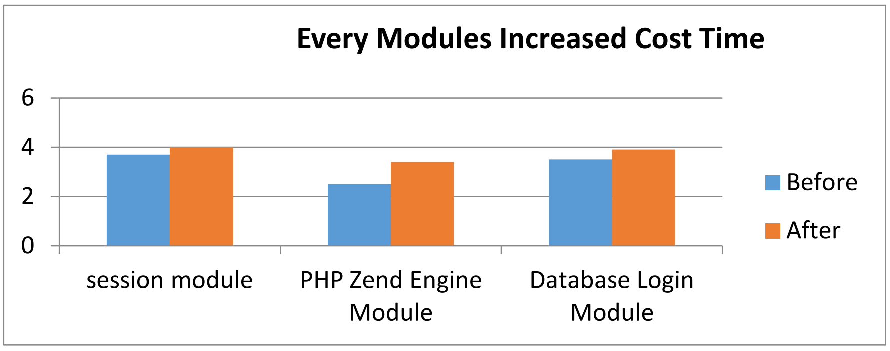

Fig. 5 Compare result

Summary

For a long time, the access control has many methods such as role based access control [6], language structure based access control [7], and many other control methods [8]. In this paper, we introduce a new access control method which amend the fatal vulnerabilities and the complexity control method in the old model. Basing on this model, we can supply with multiple ring level privileges control, distinguish different request and upgrade Web application security environment. In this paper, basing on the PHP environment, we used Linux system to configure and PHP extension technology design and implement a model. The practice has proved that this model could be used in Web application’s security guard and meet the needs of the users.

Acknowledgements

This work is supported by NSFC (Grant Nos. 61300181, 61272057, 61202434, 61170270, 61100203, 61121061), the Fundamental Research Funds for the Central Universities (Grant No. 2012RC0612, 2011YB01), 242 projet No.2014Z025.

References

[1] Information on http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-2905

[2] Information on http://www.wooyun.org/bugs/wooyun-2014-059139

[3] Information on http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0512

[4] Jayaraman, K, Wenliang Du, Rajagopalan, B., ChAPIn, S.J., ESCUDO: A fine-grained protection model for Web browsers, Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, 6(30), pp.231-240, 2010.

[5] Xi Tan, Wenliang Du, Tongbo Luo, et al, SCUTA: A Server-Side Access Control System for Web Applications, SACMAT '12 Proceedings of the 17th ACM symposium on Access Control Models and Technologies, 06(17), pp.71-82, 2012.

[6] Bo Song, Shengbo Chen, Roles-based Access Control Modeling and Testing for Web Applications, Software Engineering (WCSE), 2012 Third World Congress on, 11(3), pp.57-62, 2012.

[7] Stephen Chong, K.Vikram, Andrew C.Myers, SIF Enforcing Confidentiality and Integrity in Web Applications, SS'07 Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, 08(16), pp.1-16, 2007.

[8] Ezra Cooper, Sam Lindley, Philip Wadler, Links: Web programming without tiers, the 5th international conference on Formal methods for components and objects,11(5), pp.266-296, 2006.