NCSTISC Linux Kernel PWN450

转载

tangsilian 于 2018-11-25 23:45:45 发布 212 收藏

分类专栏： # 3 CTF

3 CTF 专栏收录该内容

2 篇文章 0 订阅

订阅专栏

0x00 参考：

通过真题理解通过UAF漏洞修改tty_strcut绕过semp并进行提权。

http://whereisk0shl.top/NCSTISC Linux Kernel pwn450 writeup.html

https://www.anquanke.com/post/id/85281

https://www.anquanke.com/post/id/86490

0x01 UAF漏洞学习

Use After free漏洞简单的理解示例如下：

https://blog.csdn.net/qq_20307987/article/details/51511230

```
#include <stdio.h>
#include <stdlib.h>
typedef void (*func_ptr)(char *);
void evil_fuc(char command[])
{
system(command);
}
void echo(char content[])
{
printf("%s",content);
}
int main()
{
func_ptr *p1=(int*)malloc(4*sizeof(int));
printf("malloc addr: %pn",p1);
p1[3]=echo;
p1[3]("hello worldn");
free(p1); //在这里free了p1,但并未将p1置空,导致后续可以再使用p1指针
p1[3]("hello againn"); //p1指针未被置空,虽然free了,但仍可使用.
func_ptr *p2=(int*)malloc(4*sizeof(int));//malloc在free一块内存后,再次申请同样大小的指针会把刚刚释放的内存分配出来.
printf("malloc addr: %pn",p2);
 printf("malloc addr: %pn",p1);//p2与p1指针指向的内存为同一地址
p2[3]=evil_fuc; //在这里将p1指针里面保存的echo函数指针覆盖成为了evil_func指针.
p1[3]("whoami");
return 0;
}
```

1、申请一段空间，并将其释放，释放后并不将指针置为空，因此这个指针仍然可以使用，把这个指针简称为p1。

2、申请空间p2，由于malloc分配的过程使得p2指向的空间为刚刚释放的p1指针的空间，构造恶意的数据将这段内存空间布局好，即覆盖了p1中的数据。（为什么会完美覆盖p1，这是个问题）

3、利用p1，一般多有一个函数指针，由于之前已使用p2将p1中的数据给覆盖了，所以此时的数据既是我们可控制的，即可能存在劫持函数流的情况。

0x02 真实的题目

一：gdb调试内核

二：绕过semp

三：修改结构体进行提权