CLASS 1

吴瑞欣-E41614059

1、linux 下用 open 函数打开文件时，要是采用 O_WRONLY 模式为何容易产生竞争条件漏洞？换成 O_WRONLY | O_CREAT | O_EXCL 模式后情况会如何？

open 函数用来打开一个设备，他返回的是一个整型变量，如果这个值等于-1, 说明打开文件出现错误, 如果为大于 0 的值，参考格式>int open(const char *pathname, int oflag, …/*, mode_t mode * /) ;

打开的操作类型有如下几种:

1) O_RDONLY 只读打开

2) O_WRONLY 只写打开

3) O_RDWR 读、写打开

Reference: https://blog.csdn.net/wdzxl198/article/details/6876879

2、（不需要完全搞懂全部流程）阅读一篇文章“从一个漏洞谈 ptrace 的漏洞发现及利用方法”，地址为：http://www.nsfocus.net/index.php?act=magazine&do=view&mid=1795。描述其中的竞争条件漏洞出现的原因。

解：代码是在内核线程 exec_modprobe（）的上下文运行的，程序的 current 指向这个内核线程的 task_struct 结构，而与创建这个线程时的 current 不同，那时候的 current 指向当时的当前进程，即

exec_modprobe（）的父进程。内核线程 exec_modprobe（）从其父进程继承了绝大部分资源和特性，包括它的 fs_struct 的内容和打开的所有文件，以及它的进程号、组号，还有所有的特权。但是这些特性在这个函数里大多被拚弃了(见源码的 19 行到 42 行，这里设置了该内核线程的信号、euid、egid 等，使之变成超级用户)，不过在拚弃这些特性之前之前，我们的父进程，或同组进程是应该可以调试该内核线程的。漏洞也就在这里。

3、（选做）上网搜索 CVE-2016-5195 漏洞的相关资料。描述其中的整数溢出和竞争条件漏洞出现的原因。

解:整数溢出： 存储的值超过类型系统最大的表示范围叫 interger overflow,小于其最小表示范围叫 interger underflow,整数溢出一般不会直接导致任意代码执行,但是会导致栈或者堆溢出间接引发任意代码执行

竞争条件漏洞: 竞争条件漏洞就是多个进程访问同一资源时产生的时间或者序列的冲突，并利用这个冲突来对系统进行攻击。

4 、 echo "crack:"$(openssl passwd -1 -salt a3g1 123456)":0:0:,,,:/root:/bin/bash"的输出结果是什么？root 用户把输出的这一行加入/etc/passwd 末尾会产生什么效果？（提前备份该文件，之后恢复）

输出如下:

```
[09/15/2018 17:12] seed@ubuntu:~$ echo "crack:"$(openssl passwd -1 -salt a3g1 12
3456)":0:0:,,,:/root:/bin/bash"
crack:$1$a3g1$sjnd1nkAwCfjT4/r0sTA20:0:0:,,,:/root:/bin/bash
[09/15/2018 17:12] seed@ubuntu:~$
```

我原以为会增添一个用户，crack，利用此用户可以提权成功，但是发现并没有成功。不知道是操作失误还是，结果即是如此：

```
[09/15/2018 17:37] seed@ubuntu:/bin$ su crack
Unknown id: crack
[09/15/2018 17:37] seed@ubuntu:/bin$
```

这题对结果不是很放心，心中存疑，待本节课结束后再下结论。