在Linux系统下面如何截获系统调用 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022__E5_9C_A8Linux_E7_B3_c103_145289.htm 使用Linux Kernel Module的一般目的就是扩展系统的功能，或者给某些特殊的设备提供驱动等等。其实利用Linux内核模块我们还可以做一些比较“ 黑客 ”的事情，例如用来拦截系统调用，然后自己处理。嘿嘿，有意思的说。 下面给出一个简单的例子，说明了其基本的工作过程。 来源：www.examda.com #define MODULE #define __KERNEL__ #include #include #include #include #include #include #include #include #include #include extern void* sys_call_table[]. /*sys_call_table is exported, so we can access it*/ int (*orig_mkdir)(const char *path). /*the original systemcall*/ int hacked_mkdir(const char *path) { return 0. /*everything is ok, but he new systemcall does nothing*/ } int init_module(void) /*module setup*/ { orig_mkdir=sys_call_table[SYS_mkdir]. sys_call_table[SYS_mkdir]=hacked_mkdir. return 0. } void cleanup_module(void) /*module shutdown*/ { sys_call_table[SYS_mkdir]=orig_mkdir. /*set mkdir syscall to the origal one*/ } 大家看到前面的代码了，非常简单，我们就是替换了内核的系统调用数组中我们关心的指针的值，系统调用在内核中实际就是一个数组列表指针对应的函数列表。我们通过替换我们想“ 黑 ”的函数的指针，就可以达到我们特定的目的。这个例子中我们替换了“ mkdir ”这个函数。这样，用户的应用程序如果调用mkdir后，当内核响应的时候，实际上是调用我们“ 黑 ”了的函数，而我们实现的函数里面是什

么都没有干，所以这里会导致用户运行“ mkdir ”得不到结果。这个例子很简单，但是我们可以看出，如果我们想截获一个系统调用，那么我们只需要做以下的事情： 1. 查找出感兴趣的系统调用在系统内核数组中的入口位置。可以参看include/sys/ syscall.h文件。 2. 将内核中原来的调用函数对应的指针sys_call_table[X]保留下来。 3. 将我们新的伪造的系统函数指针给sys_call_table[X]。 不妨试试看。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com