# 项目#5：Rootkit

# ECE 650 – 2023 年春季

**截止日期：2023 年 5 月 4 日，星期四，晚上 11:59**

## 一般说明

1. 您将单独完成此任务。

2. 应使用 Linux Virtual 开发和测试此作业的代码
机器基于**Ubuntu 20.04**您可以在以下链接创建：
https://vcm.duke.edu/
选择下图：Ubuntu 20.04 VM
内核版本：Linux 5.4.0-104-generic（您可以使用“uname -r”查看您的内核版本）您可以使用“apt-cache search linux-image”搜索图像并使用“apt-get upgrade [image name]”更新到特定的内核版本

请确保您使用的是正确的环境。否则，您的代码在我们的测试环境中可能无法正常工作。

3. 您必须仔细遵循此作业指南，并上交作业中的所有内容
描述的格式。否则，您将失去成绩。

4. 此作业涉及编写和测试内核模块。如果你的程序中有错误
模块，**完全有可能破坏你的系统**. 确保使用**快照功能**在 Duke VM 上恢复到可用状态。最好在不影响任何其他工作的系统上工作。

5. 需要注意的是，该项目不会进行预测试。

## 概述

在此作业中，您将实现 Rootkit 的一部分功能以获得： 1. 内核编程的实践练习

2、详细了解内核内部系统调用的运行
3.练习用fork/exec启动子进程
4. 了解攻击者可能尝试对系统进行的恶意活动类型（特别是针对特权系统程序）。

我们的假设是，通过成功利用漏洞，您获得了在系统中执行特权代码的能力。您的“攻击”代码将由您将编写的一个小程序表示，该程序将（除其他一些内容外，如下所述）加载一个内核模块，该模块将隐藏您的攻击程序及其一些恶意活动的存在. 接下来描述攻击程序和内核模块所需的特定功能（以及有关实现此功能的有用提示）。

## 使用虚拟机的技巧

当您第一次创建虚拟机并登录时，您会注意到可能安装了几个程序（例如没有 gcc、emacs、vim 等）。您可以使用以下命令轻松下载您选择的软件：sudo apt-get install <package name>。例如：

sudo apt install build-essential emacs

## 详细的提交说明

您提交的内容将包括 3 个（并且只有 3 个）文件：1.
**偷偷摸摸的_mod.c**–具有如下所述功能的偷偷摸摸的模块的源代码。

2. **鬼鬼祟祟的进程.c**–具有如下所述功能的偷偷摸摸（攻击）程序的源代码。

3. **生成文件**–一个将“sneaky_process.c”编译成“sneaky_process”，并将“sneaky_mod.c”编译成“sneaky_mod.ko”的makefile。在大多数情况下，这只是与框架模块示例代码一起提供的示例 Makefile。

您将提交一个名为“**proj5_netid.zip**”到 gradescope，例如：
zip proj5_netid.zip sneaky_mod.c sneaky_process.c Makefile

## 攻击程序

您的攻击程序（名为 sneaky_process.c）将让您练习通过从用户程序调用相关 API（用于进程创建、文件 I/O 和从标准输入接收键盘输入）来执行系统调用。如果你运行，你的程序应该按以下步骤运行 sudo ./sneaky_process:

1. 你的程序应该将它自己的进程 ID 打印到屏幕上，消息如下（代码中的打印命令可能不同，但打印的文本应该匹配）： printf(“sneaky_process pid = %d\n”, getpid());

2. 你的程序将执行 1 次恶意行为。它将 /etc/passwd 文件（用于用户身份验证）复制到一个新文件：/tmp/passwd。然后它将打开 /etc/passwd 文件并在文件末尾打印一个新行，其中包含一个用户名和密码，可以允许所需用户对系统进行身份验证。请注意，这实际上不会让您以“偷偷摸摸的用户”身份向系统进行身份验证，但此步骤说明了攻击者可能利用的一种破坏行为。添加到密码文件的这一行应该完全如下： 鬼鬼祟祟的用户：abc123:2000:2000:鬼鬼祟祟的用户：/root:bash

3. 您的程序将使用“insmod”命令加载偷偷摸摸的模块（sneaky_mod.ko）。请注意，在加载模块时，您的偷偷摸摸的程序也会将其进程 ID 传递到模块中。您可以参考以下页面以了解如何在加载内核模块时将参数传递给它：

http://www.tldp.org/LDP/lkmpg/2.6/html/x323.html
4. 然后您的程序将进入一个循环，从键盘输入中一次读取一个字符，直到它接收到字符“q”（表示退出）。然后程序将退出这个等待循环。请注意，这里的这一步是为了让您有机会与系统交互，同时：1) 您的偷偷摸摸的进程正在运行，以及 2) 偷偷摸摸的内核模块已加载。**这是恶意行为将在进程运行时在同一系统上的另一个终端中进行测试的时间点。**

5. 您的程序将使用“rmmod”命令卸载偷偷摸摸的内核模块。
6. 您的程序将通过将 /tmp/passwd 复制到 /etc/passwd 来恢复 /etc/passwd 文件（并删除添加的“sneakyuser”身份验证信息）。

回想一下，进程可以通过以下方式执行新程序：1) 使用 fork() 创建子进程，以及 2) 子进程可以使用 exec*() 系统调用的某种风格来执行新程序。您将希望您的父攻击进程在子进程的每个 fork() 之后等待新的子进程（例如使用 waitpid(...) 调用）。记住**系统（）**是 fork() 和 execve() 的包装器。

## Sneaky Kernel Module（Linux 内核模块 – LKM）

您偷偷摸摸的内核模块将执行以下颠覆性操作： 1. 它将从“ls”和“find”UNIX 命令中隐藏“sneaky_process”可执行文件。例如，如果名为“sneaky_process”的可执行文件位于 /home/userid/hw5：

A。“ls /home/用户名/hw5”应该显示该目录中除“sneaky_process”之外的所有文件。
b. “cd /home/用户名/hw5；ls”应该显示该目录中除“sneaky_process”之外的所有文件

C。“找到 /home/userid -name sneaky_process”不应返回任何结果
2. 在UNIX 环境中，每个正在执行的进程在/proc 下都有一个目录，该目录以其进程ID 命名（例如/proc/1480）。该目录包含有关该过程的许多详细信息。您偷偷摸摸的内核模块将隐藏 /proc/<sneaky_process_id> 目录（注意隐藏具有特定名称的目录等同于隐藏文件！）。例如，如果您的 sneaky_process 被分配了 500 的进程 ID，那么：

A。“ls /proc”不应显示名为“500”的子目录
b. “ps -a -u <您的用户 ID>”不应显示名为“sneaky_process”的进程 500 的条目（因为“ps”命令查看 /proc 目录以检查所有正在执行的进程）。

3. 它将隐藏 sneaky_process 对 /etc/passwd 文件所做的修改。当看到打开“/etc/passwd”的请求时，它将通过打开保存的“/tmp/passwd”来做到这一点。例如：

A。“cat /etc/passwd”应该返回原始密码文件的内容，而不修改偷偷摸摸的过程对 /etc/passwd.conf 的修改。
4. 它会隐藏 sneaky_module 本身是已安装的内核模块这一事实。活动内核模块列表存储在 /proc/modules 文件中。因此，当读取该文件的内容时，sneaky_module 将从返回的读取数据缓冲区中删除“sneaky_mod”行的内容。例如：

A。“lsmod”应该返回除“sneaky_mod”之外的所有模块的列表

您的整体提交将通过编译您的内核模块和偷偷摸摸的进程、运行偷偷摸摸的进程，然后执行如上所述的命令来测试，以确保您的模块正在执行预期的颠覆性操作。

## 实施 sneaky_mod.c 的有用提示和技巧

● 此任务不应需要大量代码。例如，在我的
示例解决方案，sneaky_mod.c 文件大约有 200 行，sneaky_process.c 更短。

● 您可以使用“strace”UNIX 检查命令发出的系统调用
命令，例如“strace ls”。
● 如果您对“pt_regs”结构感到困惑，您可能会发现此链接很有帮助。 https://elixir.bootlin.com/linux/v5.4/source/arch/x86/include/uapi/asm/ptrace.h#L44

● 您需要从“pt_regs”结构中获取参数。系统调用的参数传递约定是：param#0到param#5分别传入**DI**,**国际单位制**,**DX**,**R10**,**R8**和**R9**寄存器。您可以在此处找到 pt_regs 的定义：

https://elixir.bootlin.com/linux/v5.4/source/arch/x86/include/asm/ptrace.h#L56

● 对于偷偷摸摸的内核模块中的这些颠覆性行为，您将需要劫持（和
可能修改系统调用返回的内容。

● 对于#1 和#2，我强烈建议阅读“man getdents64”页面（包括
代码示例）。它将填充一个“struct linux_dirent64”对象数组，一个对应目录中的每个文件或目录。

● 对于 #2，您可以使用 module_param(...)
此处描述的技术：http://www.tldp.org/LDP/lkmpg/2.6/html/x323.html

● 对于 #3，您应该检查“openat”系统调用（如在骨架 kermel 模块中
贴）。请注意，例如，如果您想将一个新的字符串文件名传递给 open 系统调用函数，则该字符串必须位于“用户空间”中，而不是在您的内核空间中定义的内容
模块。您可以使用“copy_to_user(…)”函数来实现： copy_to_user(void __user *to, const void *from, unsigned long nbytes)

提示，对于用户缓冲区，您可以使用传递给 openat(...) 调用的字符缓冲区吗？

● 对于#4，您可能需要查看“read”系统调用。

● 您可能还需要了解 enable_page_rw & disable_page_rw 和
程序中使用的sys_call_table。

● 您可能会发现诸如 memmove 之类的函数对于从数据中“删除”信息很有用
structure，即将要删除的条目之后的所有内容复制到要删除的数据开始的位置。

● 如果您有兴趣了解一些骨架模块代码
更深入，请在ED上提问！我们很乐意提供详细说明。

● 如果您想在内核空间中分配和释放内存，请查看 kvzalloc ̀kvfreè 。
和

- 如果您想将某些内容从用户空间复制到内核空间，请查看 copy_from_user。

- 享受这个任务吧！尝试其他偷偷摸摸的动作，如果你愿意，一旦你得到它的诀窍。