COMP2700 实验室 5：Unix 安全

在本实验室中，我们将了解一些与安全相关的概念，以及它们是如何在 Linux 中实现的。这些概念包括：主体（进程）、访问控制（文件和目录权限）、受控调用（通过 SUID）和 *下层*（规避文件系统访问控制）。

下面的练习假定我们使用的是为课程提供的实验室虚拟机。具体来说，我们有两组用户：导师组和学生组。辅导员组由用户 alice、bob 和 charlie 组成。学生组由用户 dennis、eve 和 felix 组成。管理员用户名为 `admin2700`。要完成本实验，您需要熟悉实验 1 和实验 2 中讨论的 Linux 命令。

实验室设置

以 alice 登录，下载 lab5.tar.gz 文件并解压。

```
$ su -l alice
$ wget http://users.cecs.anu.edu.au/~tiu/comp2700/lab5.tar.gz
$ extract-lab lab5.tar.gz
```

这将在 alice 的主目录下创建 lab5 目录。下面的练习将引用 lab5 中的文件和目录。

文件和目录的用户、所有权和权限

我们在此回顾一些切换用户（成为其他用户或 root 用户）以及更改资源所有权和权限的基本命令。复习结束后，进行所要求的练习。

用户和用户组

以下命令可用于查询组和用户的身份：

- `whoami`：该命令显示当前登录用户的用户名。
- `id`：该命令显示与用户名相关联的用户 ID 以及该用户所属的组。例如

  ```
  $ id alice
  ```

将显示 alice 的 uid、主 gid 和 alice 所属的所有组。

- 组：该命令显示用户所属的所有组。

特权升级

以下命令允许用户在经过适当身份验证的情况下假定另一个用户的身份。

- `su` 命令。该命令可用于切换到 shell 中的另一个用户。语法如下

```
su [用户名]
```

- 其中 [username] 是目标用户名。

sudo 命令。该命令允许 sudoers（sudo 组中的用户，如用户实验室虚拟机中的 admin2700）成为根用户：

```
sudo [选项] [命令]
```

其中 [command] 是要在 root 权限下执行的命令。其中一个有用的命令是 bash shell 本身：

```
$ sudo /bin/bash
```

然后以根用户身份运行 bash shell。更简短的版本

```
$ sudo -s
```

改变所有权

语法

```
chown [所有者:组] [文件或目录]
```

chown（更改所有者）命令通常需要在 root 权限下运行（因此可能需要 sudo）。作为普通用户，你通常没有这样做的权限。不过，你总是可以将属于你的文件更改为你所在的另一个组。例如

```
$ chown alice:tutors 文档
```

将 Documents 目录的所有者用户和所有者组分别更改为 alice 和 tutors。

更改权限

chmod 命令用于更改文件或目录的权限。其语法大致如下

```
chmod [options] [file_or_directory] （文件或目录
```

命令的选项部分指定了文件的权限。可以使用权限位（通过八进制符号）来指定权限，这在有关 Unix 安全性的讲座中已经讨论过，也可以使用更方便的文本表示法来指定，例如下面的例子。

有关更改权限的各种选项的更多详情，请参阅 "man chmod"。一些例子

- chmod 644 test.txt：该命令赋予文件 test.txt 的所有者用户读写权限，组所有者的读取权限，以及其他人的读取权限。
- chmod 4755 程序：该命令赋予用户所有者读取、写入和执行的权限，赋予组所有者和其他人读取和执行的权限，并将 "program "变为 SUID 二进制文件。
- chmod u+rwx test.txt：此命令为文件 test.txt 的所有者添加读取、写入和执行权限
- chmod g-w test.txt：该命令将删除 test.txt 所有者组的写入权限。
- chmod g+x test.txt：该命令为 test.txt 的所有者组添加执行权限。
- chmod o-r test.txt：该命令将删除其他文件的读取权限。
- chmod u+s program：该命令设置文件 "program "的 SUID 位。

陈
2024-08-15 16:27:27
八进制记数法

检查权限和所有权

回顾一下讲座和之前的实验，要显示文件和目录的权限和所有权详情，可以使用以下命令：

```
$ ls -l [file_or_directory] (文件或目录)
```

例如

```
$ ls -l ~/lab5/
```

显示 `~/lab5` 中文件和目录的权限和所有权详情。

练习 1.在 `~/lab5/share/` 目录中有几个文件。这些文件目前的所有权限位都设置为 0，因此不允许访问其中的任何文件。更改这些文件的权限和/或组所有权，以实现以下访问控制。

.文件 `tutorsonly.txt`：给 alice 读写权限，给导师组成员读权限，其他用户无权限。

.文件 `courses.txt`：给 alice 读写权限，给辅导员组的成员读写权限，给其他用户读权限。

.文件 `feedbacks.txt`：给 alice 读写权限，给导师组成员读权限，其他用户无权限。

.文件 addfeedback：这是一个二进制程序，用于将用户输入的信息添加到 feedbacks.txt 文件中。更改该程序的权限和/或组所有权，以便除辅导员（alice 除外）以外的任何人都可以执行该程序，在 feedbacks.txt 中添加反馈信息（您需要先完成练习 1.3，以更改 feedbacks.txt 的权限，如上文练习所示）。

练习 2.要读取一个目录中的文件内容，用户对该目录的最小权限是多少？假设用户知道文件名，并拥有对文件的读写权限。要回答这个问题，请使用虚拟机测试您的假设，例如，创建一个新目录（如 testdir），并在 testdir 中创建一个文件（如 testfile）。然后试验 testdir 的不同权限组合，并执行 cat testdir/testfile。

流程

请注意，在 Unix/Linux 的访问控制机制中，进程与 "主体 "相对应。ps 命令可用于列出系统中运行的进程。默认情况下，不带任何参数的 ps 命令将显示当前终端中当前用户的活动进程。要查看系统中运行的所有进程（包括 root 拥有的进程），可以使用以下命令：

```
$ ps -e -F
```

陈
2024-08-15 16:44:44

指定输出

我们还可以使用选项 -o（除了 -e）选择要显示的列。例如，下面的命令会显示 "real user"（启动此进程的用户）、进程 ID 和正在执行的命令的信息。

```
$ ps -e -o ruser,pid,comm
```

您可能会发现，有些用户名（如 admin2700）被截断，显示为 admin27+。您可以用 :N（N 为所需宽度）指定宽度来设置列的宽度，例如，将 ruser 列的宽度设置为 10 个字符：

```
$ ps -e -o ruser:10,pid,comm
```

ps 命令有丰富的选项，可以显示与系统中运行的进程有关的各种信息。使用 man ps 查看可用选项。

信号

在 Linux 中，操作系统可以通过信号与进程交互。一个常见的信号是 SIGTERM 信号，其数值为 15，用于通

知进程优雅地终止。

要向进程发送信号，请使用 kill 命令：

```
kill -[信号值] [进程 ID]
```

例如

```
$ kill -15 1658
```

向进程 id 为 1658 的进程发送信号 15（该信号的含义见下文）。以下是 Linux man 页面（"man 7 signal"）的摘录，其中包含信号的相关细节。

```
信号处置                                    倾向，性格
    每个信号都有一个当前处置，它决定了进程在收到信号时的行为方式。

    下表 "操作 "栏中的条目指定了每个信号的默认处理方式，如下所示：

    学期      默认值    行动   是  至  终止进程。
    点火      默认值    行动   是  至  忽略信号。
    核心      默认值    行动   是  至  终止进程并                转储   核心   (见
              core(5))。
    停止默认 操作是停止进程。

    内容     默认       行动     是   如果进程当前已停止，则     继续执行。

标准信号
    Linux 支持下列标准信号。如 "值 "一栏所示，有几个信号值与体系结构有关。(在给出三个值
    的情况下，第一个通常适用于 alpha 和 sparc，中间一个适用于 x86、arm 和大多数其他架
    构，最后一个适用于 mips。(未显示 parisc 的值；有关该架构的信号编号，请参阅 Linux
    内核源代码）。破折号 (-) 表示相应架构上没有信号。

    首先是原始 POSIX.1-1990 标准中描述的信号。信号值 操作 注释
    ──────────────────────────────────────────────────────────────────
    SIGHUP        1          学期     控制终端检测到挂机
                                      或控制程序死亡
    SIGINT        2          学期     键盘中断
    退出          3          核心     从键盘退出
    西吉尔        4          核心     非法教学
```

SIGABRT	6	核心	来自 abort(3) 的终止信号
SIGFPE	8	核心	浮点运算异常
SIGKILL	9	学期	杀死信号

```
    SIGSEGV       11          核心      无效内存引用
    SIGPIPE       13          学期      管道断裂：写入管道时没有
                                          读者；见管道(7)
    SIGALRM       14          学期      警报器发出的计时器信号(2)
    SIGTERM       15          学期      终止信号
    SIGUSR1    30,10,16       学期      用户定义信号 1
    SIGUSR2    31,12,17       学期      用户自定义信号 2
    SIGCHLD    20,17,18       点火      儿童停学或终止学业
    SIGCONT    19,18,25       Cont      如果停止则继续
    SIGSTOP    17,19,23       停止      停止进程
    SIGTSTP    18,20,24       停止      终端停止键入
    SIGTTIN    21,21,26       停止      后台程序的终端输入
    SIGTTOU    22,22,27       停止      后台进程的终端输出

不能捕获、阻止或忽略 SIGKILL 和 SIGSTOP 信号。
```

练习 3.在 `~/lab5/signals/` 目录中有两个程序：whatsmyid 和 stubborn。(我们还提供了这些程序的相应 C 代码，但你暂时不需要理解它们）。

.打开一个终端，以用户 `alice` 的身份登录并运行 `whatsmyid`。然后打开另一个终端，使用相同的登录名（`alice`），并尝试找出第一个终端中 `whatsmyid` 进程的进程 ID。然后发送 SIGTERM 终止该进程。你使用了什么命令?

.与练习 3.1 相同，但这次运行 `stubborn`。您可能会发现这个进程更难终止，因为它会试图忽略发送给它的所有信号。使用什么命令来终止这个进程？如果允许进程忽略所有信号，会有什么安全隐患?

软链接和硬链接

软链接相当于 Windows 等其他操作系统中的快捷方式。软链接和硬链接（将在下文讨论）都是使用 ln 命令创建的。例如，要创建指向文件 `foo.txt` 的快捷方式，并将其称为 `bar.txt`，只需运行

```
$ ln -s foo.txt bar.txt
```

这将使 bar.txt 指向 foo.txt。

在 Linux 中，系统会为每个文件关联一个 inode，这是一个保存文件各种信息（权限、各种属性（如创建/修改时间）、磁盘上保存数据块的链接等）的数据结构。一个文件（如 `a.txt`）的硬链接是一个与

`a.txt` 共享相同 inode 的普通文件。换句话说，它们具有相同的内容。

要创建指向文件 `a.txt` 的硬链接并将其称为 `b.txt`，只需运行命令即可：

```
$ ln a.txt b.txt
```

`ls` 命令中的 `-i` 选项可用于显示文件的 inode。`find` 命令也有一个选项，用于查找具有相同 inode 的文件。

练习 4.在 Linux 中，你可以创建一个指向不存在文件的符号链接。如果文件的存在是一个秘密信息，那么这个功能是否可以用作推断目标目录中文件存在的旁路？请演示一个实验来确认你的答案，例如，在 /root 目录中创建一些文件（以 root 用户身份），然后以非 root 用户（如 bob）身份登录，尝试观察与现有文件的软链接和与不存在文件的软链接之间是否有任何区别。

练习 5.在 `~/lab5/links/` 目录中有几个文件和目录。找出哪些文件与哪些文件有硬链接。您可能需要使用查找命令。有关如何使用查找命令的更多信息，请访问 `man find`。

SUID 计划

从讲座中可以忆及，一个进程可以有两个与之相关的用户 ID：真实用户 ID（即启动进程的用户 ID）和有效用户 ID（用于确定进程权限的用户 ID）。对于普通程序而言，一旦启动，其进程将拥有相同的真实用户 ID 和有效用户 ID。但对于 SUID 程序，真实用户 ID 可以与有效用户 ID 不同。SUID 程序用于实现受控调用的概念，通常是为了确保普通用户访问敏感的系统资源。

您可以使用 `ps` 命令同时显示与进程相关的真实用户（`ruser`）和有效用户（`euser`），例如

```
$ ps -e -oruser,euser,pid,comm
```

您还可以根据真实用户（使用 `-U`）和有效用户（使用 `-u`）过滤 `ps` 的输出。例如

```
$ ps -u root -o ruser,euser,pid,comm
```

显示以根用户的有效用户 ID 运行的进程。

练习 6.Linux 中的 `/usr/bin/passwd` 程序是 SUID 程序的一个示例，它允许普通用户修改 `/etc/shadow` 文件以更改密码。运行 `passwd` 命令（但不要更改密码）并查询进程列表，确认它是以根用户的有效用户 ID 运行的。

shell 脚本上的 SUID

在 Linux 中，出于安全考虑，SUID 位对 shell 脚本没有影响。因此，即使 shell 脚本设置了 SUID 位，在调用该脚本时，有效用户 ID 也将与真实用户 ID 相同。为了让 shell 脚本在不同的有效用户 ID 下执行，可以通过普通二进制程序间接调用 shell 脚本，例如使用 system 函数或 C 语言中的 execve 函数。

练习 7.请看下面这个使用 shell 脚本进行控制调用的示例。Alice 希望与除 Charlie 之外的所有人共享一个名为 `not_for_charlie.txt` 的文件。由于仅使用权限位无法实现这一目标，她决定实施一个 SUID 程序来强制执行这一访问策略。但她并没有编写合适的 SUID 二进制程序，而是决定使用 C 语言程序来封装对 shell 脚本的调用，该脚本会在检查当前用户名后显示文件。代码列在这里，但你也可以在 `~/lab5/suid/` 中找到它。该代码已编译成名为 `filter` 的 SUID 二进制程序。

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char *argv[], char *envp[]) {
  execve("filter.sh",argv,envp);
  返回 0;
}
```

程序 filter 使用函数 `execve` 调用 shell 脚本 `filter.sh`。文件 `filter.sh` 包含以下脚本

```
#!/bin/bash -p
if [ "$USER" = "charlie" ]; then
   echo "Go away charlie";
   出口 1;
fi
/bin/cat not_for_charlie.txt
```

脚本使用 USER 环境变量拒绝查理访问文件 `not_for_charlie.txt`。Alice 拥有对该文件的读写权限，但其他人无法直接访问该文件。现在以查理的身份登录，并尝试绕过爱丽丝的访问控制机制来读取文件 `not_for_charlie.txt`。你（作为用户 charlie）将如何实现这一目标？你认为上述 SUID 程序有什么问题?

下面一层

在 Linux 中，块设备（如硬盘驱动器）在文件系统中表示为位于 /dev目录。例如，文件 `/dev/sda` 通常代表系统中的第一个硬盘。硬盘中的分区也可以单独表示，例如，`/dev/sda1` 表示硬盘 `/dev/sda` 中的第一个分区。

并非 `/dev` 中的所有文件都代表真实设备。例如，我们有以下虚拟设备：

- `/dev/null`：这是一个字符设备，会丢弃所有写入的数据。与其他命令一起使用时，它可以抑制输出

或错误信息。例如，运行

将运行 ls 命令，但会抑制所有输出，因此不会打印任何内容。

- `/dev/零`：这是一个字符设备，代表一个零流。例如，这对擦除零位的（真实）设备非常有用。
- `/dev/urandom`：这是一个字符设备，表示（加密安全的）随机字节流。它可用于生成随机位字符串，在需要强加密随机性的应用中使用。

块设备文件（如 `/dev/sda1`）可以像普通文件一样被读写，并受制于文件系统中管理普通文件的相同权限机制。例如，对 `/dev/sda` 的读取访问允许用户读取 `/dev/sda` 所代表硬盘中的任何位块。这种对 `/dev/sda` 的位块的直接访问绕过了存储在 `/dev/sda` 上的文件系统的访问控制机制。

例如，如果 `/dev/sda` 代表系统的主硬盘驱动器，那么具有以下读取权限的用户 `/dev/sda` 将能读取硬盘中的任何比特块，包括与密码哈希值等敏感数据相关的比特块。

一些查询系统设备的有用命令：

- `df`：`df` 命令（不带参数）显示系统中当前挂载的所有设备，包括可用空间、挂载点等信息。
- `debugfs`：该命令用于直接访问块设备。命令语法为

```
debugfs [选项] 块设备
```

其中 `block-device` 表示 `/dev` 目录中的任何块设备。默认情况下，如果没有指定任何选项，块设备将以只读模式打开。要以写模式打开，请使用选项 `-w`。

> 警告：除非您知道自己在做什么，否则切勿对计算机上的实际磁盘执行低级别磁盘写入操作。在本实验室中，只能在提供的实验室虚拟机中尝试，因此任何损坏都将仅限于虚拟机的虚拟磁盘。

运行 `debugfs` 会启动一个 shell，在 shell 中可以发出各种命令来访问存储在设备上的信息，例如文件的 inodes、文件的扇区等。使用 debugfs 的帮助信息尝试 debugfs 中的不同命令。其中一个有用的命令是 `cat`，它与 bash shell 中的 `cat` 程序功能相同，即显示文件（或 debugfs 中的 inode）内容。

练习 8a.(仅限 VirtualBox 实验室虚拟机)

注意：如果您使用的是 Azure Labs VM，请跳过此练习，改做练习 8b。

在 Unix 系统中，与文件和目录相关的访问控制是在文件系统级别实施的。如果可以直接访问文件系统所在的设备，就可以规避这种访问保护。在本练习中，我们将研究如何绕过实验室虚拟机文件系统的访问控制。

首先，我们创建一个只有 root 可读的文件。打开与虚拟机的新 ssh 连接并运行（以用户 `admin2700` 的身份）：

```
$ echo hello | sudo tee /mnt/test.txt
$ sudo chmod 600 /mnt/test.txt
```

然后切换回用户 alice。

.找出挂载到根目录 / 的块设备。

.除根用户和 admin2700 用户外，哪个用户可以直接访问 / 目录下的块设备？请说明理由。

.演示上面练习 8a.2 中提到的用户如何通过文件系统的访问限制来显示文件 /mnt/test.txt 的内容

。

练习 8b。(仅限 Azure Labs 虚拟机）本练习与练习 8a 几乎相同，因此如果您已完成练习 8a，则可以跳过本练习。本练习仅适用于 Azure Labs VM，其设置方式不同（作为基于云的 VM 默认配置的一部分），因此练习 8a 在此 VM 中不起作用。

首先，我们创建一个只有 root 可读的文件。打开与 Azure Labs 虚拟机的新 ssh 连接并运行（以用户 admin2700 的身份）：

```
$ echo hello | sudo tee /mnt/test.txt
$ sudo chmod 600 /mnt/test.txt
```

然后切换回用户 alice。

.查找挂载到 /mnt 目录的块设备。

.除根用户和 admin2700 用户外，哪个用户可以直接访问 /mnt 目录下的块设备？请说明理由。

.说明上面练习 8b.2 中提到的用户如何绕过 /mnt/ 的访问限制。

显示文件 /mnt/test.txt 的内容。

扩展练习（可选）

练习 9（仅限 VirtualBox 虚拟机）。在尝试下面的练习之前，请确保为要保留在虚拟机外的文件创建了备份。

您可以创建虚拟机快照，以便在出现问题时恢复虚拟机的状态。VirtualBox 网站

(https://www.virtualbox.org/manual/ch01.html#snapshots) 提供了一些有关如何创建快照的说明。辅导员也会指导你如何创建快照。

这是练习 8a 的后续步骤。以拥有原始磁盘访问权限的非 root 用户身份登录（见练习 8a）。演示如何更改 /etc/shadow 文件的权限，使所有用户都能读写该文件，而无需 root 权限。*提示：您需要直接修改影子文件的 inode。修改权限后，可能需要重启虚拟机。*

练习 10.由 root 拥有并设置了所有者 suid 位的程序通常是漏洞研究人员的攻击目标，因为此类程序如果包含可利用的漏洞，可能会允许本地攻击者将权限升级到 root。使用 `find` 命令识别实验室虚拟机中所有 root 所有程序。提示：你可能会发现 `find` 命令的 `-perm` 和 `-user` 选项很有用。详情请查阅 `find` 手册（`man find`）。