文件权限

登录到您的 Debian VM 进行以下练习。

创建用户和组

创建一个新用户 `sudo adduser NAME` - 我将 `brian` 在这些注释中使用它作为示例名称。当它要求输入密码时，你可以使用 `brian` 或其他东西；它会抱怨密码太短，但无论如何它都会创建用户。您可以跳过要求提供全名和电话号码的 GECOS 信息——这只是为了帮助管理员在需要时与您联系。

`tail /etc/passwd` 使用和检查用户和组文件 `tail /etc/group` 以检查新用户是否已创建 - `tail` 默认显示文件的最后 10 行； `tail -n N FILE` 将显示最后 N 行。您的新用户 `brian` （或您对他们的任何称呼）应该出现在这两个文件中。另请检查 `ls -l /home` Brian 的主目录是否存在以及是否设置为正确的用户和组。

是时候更改用户： `su brian` 并输入密码。请注意，提示已更改为 `brian@debian12:/home/vagrant$` （至少如果您是在该文件夹中开始的）。因此用户已更改，并且由于 `/home/vagrant` 不再是当前用户的主目录，因此它会被完整写出。跑 `cd` 回家，然后 `pwd` 检查您现在是否处于 `/home/brian` 或您所谓的新用户中。

接下来，创建一个用户 `nigel` （或其他名称），使用命令将两个新用户（但不是 `vagrant` ）添加到组 `users` （已存在）中 `sudo usermod -aG GROUPNAME USERNAME` ，其中组和用户名会相应更改。注意： `brian` 无法使用 sudo，因此您必须退出终端才能返回到以 vagrant 身份运行的终端。

探索文件权限

作为用户 `brian` （或者您所说的第一个新用户），使用您在视频中学到的内容设置您的主目录，以便

- 你可以做一切（rwx）。
- 该 `users` 组的成员可以列出文件并更改到您的主目录，但不能添加/删除文件。 `users` 为此，您需要使用命令将主目录的组更改为 `chgrp -R GROUPNAME DIRECTORY` 。
- 其他人无法对您的主目录执行任何操作。

在您的主目录中创建一个文件，例如， `nano readme.txt` 然后添加一些内容。

通过 `su USERNAME` 以不同用户身份登录来检查：

- `nigel` 可以查看 Brian 的主目录，但不能在那里创建文件；
- `nigel` 可以查看但不能编辑 Brian 的自述文件；
- `vagrant` 根本无法列出或进入 Brian 的主目录中的文件。当你尝试时会发生什么？

当然，vagrant 可以使用 sudo 来绕过所有这些限制。权限并不能保护您免受任何可以成为 root 的人的侵害。

另外 `brian` ，在您的主文件夹中创建一个 `private` 子目录，除了您之外没有人可以访问（读取、写入或执行）。以 Brian 的主目录中的用户身份 `secret.txt` 在其中创建一个文件，并在其中放入一些内容。不要更改其自身的任何权限。 `nano private/secret.txt brian secret.txt`

检查 Nigel 是否可以看到文件夹本身，但不能 cd 进入其中，也不能列出文件。检查即使知道文件名 (`cat /home/brian/private/secret.txt`) 作为 Nigel 也不起作用。

在and中使用 `ls -l` Brian 的身份，比较文件和文件夹的条目。为什么两个文件的组不同？
`~ ~/private ~/readme.txt ~/private/secret.txt ~/private`

请注意，即使默认情况下秘密文件具有每个人的读取权限，Nigel 也无法读取它。规则是您需要对 `/` 文件的整个路径的权限才能访问它。

这是另一个提醒，如果您想在实验室计算机上存储私人文件，请将其放在只有您可以访问的文件夹中。默认情况下，其他学生可以读取您的主目录，并且他们可以查看您的作业。这在过去导致了抄袭问题，但好消息是：我们保留日志并且通常可以弄清楚发生了什么！ :-）。

或者，您可以删除其他人对您的主目录的权限，但这会阻止您共享您确实想要与其他学生共享的特定文件夹中的文件。

塞图伊德

我们将创建一个文件，让 Nigel（以及用户组中的其他人）向 Brian 发送消息，这些消息将保存在他的主目录中的文件中。

作为 Brian，在您的主目录中创建一个文件 `message-brian.c` 并添加以下行：

```c
#include <stdio.h>
#include <stdlib.h>

const char *filename ="/home/brian/messages.txt";

int main(int argc, char **argv) {
  if (argc != 2) {
    puts("Usage: message-brian MESSAGE");
    return 1;
  }
  FILE *file = fopen(filename, "a");
  if (file == NULL) {
    puts("Error opening file");
    return 2;
  }
  int r = fputs(argv[1], file);
  if (r == EOF) {
    puts("Error writing message");
    return 2;
  }
  r = fputc('\n', file);
  if (r == EOF) {
    puts("Error writing newline");
    return 2;
  }
  fclose(file);
  return 0;
}
```

编译它 `gcc -Wall message-brian.c -o message-brian` （你不应该收到任何警告）并检查 `ls -l`，你会看到像这样的行

```
-rwxr-xr-x    1 brian     brian         19984 Oct 28 13:26 message-brian
```

这些是新创建的可执行文件的默认权限；请注意，gcc 已 `+x` 为您设置了三位。还是像 Brian 一样， `chmod u+s message-brian` 再次运行并检查文件：您现在应该看到 `-rwsr-xr-x` 文件权限。这 `s` 是 setuid 位。

作为 Nigel（ `su nigel`），进入 Brian 的主目录并运行 `./message-brian "Hi from Nigel!"`。这里需要引号，因为程序只接受一个参数。

现在运行 `ls -l` 并注意 a `messages.txt` 已与所有者和组一起出现 `brian` 。用 检查内容 `cat messages.txt` 。 `messages.txt` 尽管 Nigel 无法亲自在 Brian 的主目录中创建和编辑文件（例如，他无法编辑，但他可以读取文件），但程序 `message-brian` 以 Brian 的身份运行，这让它创建文件。Nigel 可以发送另一条类似这样的消息（ `./message-brian "Hi again!"`），该消息会附加到文件中：尝试一下。

这显示了如何使用 setuid 程序来允许其他用户在不同的用户帐户下有选择地执行特定任务。

警告：如果您不了解安全编码和黑客 C 程序的基础知识，那么编写自己的 setuid 程序是极其危险的，因为此类程序中的错误可能会让某人接管您的用户帐户。您至少应该了解我们的安全单元的内

容，包括第四年。

安全分析师的一般任务可能是查找系统上设置了 setuid 位的所有文件。您可以自己尝试一下，但首先返回到 vagrant shell，以便可以使用 sudo：

```
sudo find / -perm /4000
```

您可能会收到一些与文件相关的错误 `/proc` ，您可以忽略这些错误：这些是查找用于查看单个文件的子进程。

除了 之外 `message-brian` ，默认情况下您还会发现一些文件： `sudo` 、 `mount` 和。第一个你已经知道了；查看接下来的两个做什么并思考为什么他们是 setuid。具体来说，根据手册页，非 root 用户可以执行哪些类型的（卸载）挂载？ `umount su`

`passwd` 在手册页中查找该程序。为什么该程序需要 setuid？

须藤

确保您的终端正在运行 `brian` 并尝试 `sudo ls` . 您将看到一条一般消息，系统会要求您输入密码，然后您将收到错误消息 `brian is not in the sudoers file. This incident will be reported.` （这意味着条目已登录 `/var/log/messages` 。）

所以， `brian` 目前还不能使用sudo。切换回 `vagrant` 并运行命令 `sudo cat /etc/sudoers` 。 `root ALL=(ALL) ALL` 除了最后一行 `#includedir /etc/sudoers.d` （这不是注释！）之外的所有内容都被注释掉，它包含一个 `vagrant` 带有该行的文件 `vagrant ALL=(ALL) NOPASSWD: ALL` ，这就是 vagrant 可以首先使用 sudo 的原因。

但是，请注意注释行，例如

```
# %wheel ALL=(ALL) NOPASSWD: ALL
# %sudo ALL=(ALL) ALL
```

如果未注释，第一个将允许组中的每个人使用 sudo 运行命令（这是其他一些 Linux 发行版上的默认设置），而第二个将允许组中的每个人 `sudo` 执行此操作，但会事先提示输入自己的密码。

让我们允许用户组中的人员重新启动计算机。使用 as vagrant 打开 root shell `sudo su` ；这样，如果我们破坏了 sudo，我们就不会被锁定。

以 root 身份编辑 sudoers 文件 `visudo` ，并添加以下行：

```
%users ALL=(ALL) /sbin/reboot
```

并保存 sudoers 文件。

警告： `/etc/sudoers` ：切勿直接编辑，而*应始终*使用 `visudo` 。如果您犯了一个错误并向文件添加了语法错误，那么 `sudo` 将拒绝工作。如果您的根帐户没有密码（有些人不喜欢这样做作为安全预防措施），那么您将不得不花费接下来的半小时来弄清楚如何闯入您自己的计算机并夺回控制权。在替换当前配置文件之前，几乎总是有一个命令来检查配置文件：相同的建议也适用于 ssh 配置文件。如果你破坏了它们，你可能不得不带着键盘和显示器前往服务器所在的任何地方。

您现在可以切换回 `brian` （检查提示以确保您是 Brian）并执行 `sudo reboot` 。在询问 Brian 的密码后，虚拟机现在将重新启动，您会注意到这一点，因为您被踢出了 ssh 连接。几秒钟后另一个 `vagrant ssh` 将使您再次返回。

Advanced note

重新启动后，您的 `/vagrant` 共享文件夹可能无法使用。在这种情况下，请注销并在主机上重复执行此操作 `vagrant halt` 。 `vagrant up vagrant ssh`

当 vagrant 启动你的虚拟机时，它会自动设置共享文件夹，但如果你自己重新启动虚拟机，这并不总是有效。