## 5

### CONTROLLING FILE AND DIRECTORY PERMISSIONS



Not every user of a single operating system should have the same level of access to files and directories. Like any professional or enterprise­level operating system, Linux has methods for securing file and directory access. This security system allows the system administrator—the *root* user—or the file owner to protect their files from unwanted access or tampering by granting select users *permissions* to read, write, or execute files. For each file and directory, we can specify the permission status for the file’s owner, for particular groups of users, and for all other users. This is a necessity in a multiuser, enterprise­level operating system. The alternative would be quite chaotic.

In this chapter, I’ll show you how to check for and change permissions on files and directories for select users, how to set default file and directory permissions, and how to set special permissions. Finally, you will see how a hacker’s understanding of permissions might help them exploit a system.

并非单个操作系统的每个用户都应具有相同级别的文件和目录访问权限。与任何专业或企业级操作系统一样，Linux具有保护文件和目录访问的方法。此安全系统允许系统管理员（root用户或文件所有者）通过授予选择用户读取，写入或执行文件的权限来保护其文件免受不必要的访问或篡改。对于每个文件和目录，我们可以为文件所有者，特定用户组以及所有其他用户指定权限状态。这在多用户企业级操作系统中是必需的。替代方案将非常混乱。

在本章中，我将向您展示如何检查和更改选定用户的文件和目录的权限，如何设置默认文件和目录权限以及如何设置特殊权限。最后，您将看到黑客对权限的理解如何帮助他们利用系统。

##### DIFFERENT TYPES OF USERS

As you know, in Linux, the root user is all­powerful. The root user can do basically *anything* on the system. Other users on the system have more limited capabilities and permissions and almost never have the access that the root user has.

These other users are usually collected into *groups* that generally share a similar

function. In a commercial entity, these groups might be finance, engineering, sales, and so on. In an IT environment, these groups might include developers, network administrators, and database administrators. The idea is to put people with similar needs into a group that is granted relevant permissions; then each member of the group inherits the group permissions. This is primarily for the ease of administering permissions and, thus, security.

The root user is part of the root group by default. Each new user on the system must be added to a group in order to inherit the permissions of that group.

如您所知，在Linux中，root用户非常强大。 root用户基本上可以在系统上执行任何操作。 系统上的其他用户具有更多有限的功能和权限，并且几乎从未拥有root用户具有的访问权限。

这些其他用户通常被收集到通常共享相似的组中

功能。 在商业实体中，这些组可能是财务，工程，销售等。 在IT环境中，这些组可能包括开发人员，网络管理员和数据库管理员。 我们的想法是将具有相似需求的人员放入被授予相关权限的组中;然后该组的每个成员都会继承组权限。 这主要是为了便于管理权限，从而确保安全性。

默认情况下，root用户是根组的一部分。 必须将系统上的每个新用户添加到组中才能继承该组的权限。

##### GRANTING PERMISSIONS

Each and every file and directory must be allocated a particular level of permission for the different identities using it. The three levels of permission are as follows:

**r** Permission to read. This grants permission only to open and view a file.

**w** Permission to write. This allows users to view and edit a file.

**x** Permission to execute. This allows users to execute a file (but not necessarily view or edit it).

In this way, the root user can grant users a level of permission depending on what they need the files for. When a file is created, typically the user who created it is the owner of the file, and the owning group is the user’s current group. The owner of the file can grant various access privileges to it. Let’s look at how to change permissions to pass ownership to individual users and to groups.

通过这种方式，root用户可以根据用户需要的权限向用户授予一定级别的权限。 创建文件时，通常创建文件的用户是文件的所有者，拥有组是用户的当前组。 该文件的所有者可以授予它各种访问权限。 让我们看看如何更改权限以将所有权传递给单个用户和组。

Granting Ownership to an Individual User

To move ownership of a file to a different user so that they have the ability to control permissions, we can use the chown (or change owner) command:

kali >**chown** ➊**bob** ➋**/tmp/bobsfile**

Here, we give the command, the name of the user we are giving ownership to, and then the location and name of the relevant file. This command grants the user account for Bob ➊ ownership of *bobsfile* ➋.

Granting Ownership to a Group

To transfer ownership of a file from one group to another, we can use the chgrp (or change group) command.

Hackers are often more likely to work alone than in groups, but it’s not unheard of for several hackers or pentesters work together on a project, and in that case, using groups is necessary. For instance, you might have a group of pentesters and a group of security team members working on the same project. The pentesters in this example are the root group, meaning they have all permissions and access. The root group needs access to the hacking tools, whereas the security folk only need access to defensive tools such as an intrusion detection system (IDS). Let’s say the root group downloads and installs a program named newIDS; the root group will need to change the ownership to the security group so the security group can use it at will. To do so, the root group would simply enter the following command:

要将文件的所有权从一个组转移到另一个组，我们可以使用chgrp（或更改组）命令。

黑客通常更有可能单独工作而不是团体工作，但对于一些黑客或测试者在项目上一起工作并不是闻所未闻，在这种情况下，使用组是必要的。 例如，您可能有一组测试人员和一组安全团队成员在同一个项目上工作。 此示例中的测试者是根组，这意味着他们具有所有权限和访问权限。 根组需要访问黑客工具，而安全人员只需要访问防御工具，如入侵检测系统（IDS）。 假设根组下载并安装名为newIDS的程序;根组需要将所有权更改为安全组，以便安全组可以随意使用它。 为此，根组只需输入以下命令：

kali >**chgrp** ➊**security** ➋**newIDS**

This command passes the security group ➊ ownership of newIDS ➋.

Now you need to know how to check whether these allocations have worked. You’ll do that by checking a file’s permissions.

##### CHECKING PERMISSIONS

When you want to find out what permissions are granted to what users for a file or directory, use the ls command with the –l (long) switch to display the contents of a

directory in long format—this list will contain the permissions. In isting 5­1, I use the

ls –l command on the file */usr/share/hashcat* (one of my favorite password­cracking tools) in order to see what we can learn about the files there.

如果要查找为文件或目录的哪些用户授予的权限，请使用带有-l（长）开关的ls命令来显示a的内容。

长格式的目录 - 此列表将包含权限。 在51号，我用的是

ls -l命令在文件/ usr / share / hashcat（我最喜欢的密码破解工具之一）上，以便查看我们可以在那里了解的文件。

kali >**ls –l /usr/share/hashcat**

total 32952

➊ ➋ ➌ ➍ ➎ ➏ ➐

drwxr­xr­x 5 root root 4096 Dec 5 10:47 charsets

­rw­r­­r­­ 1 root root 33685504 June 28 2018 hashcat.hcstat

­rw­r­­r­­ 1 root root 33685504 June 28 2018 hashcat.hctune

drwxr ­xr­x 2 root root 4096 Dec 5 10:47 masks

drwxr ­xr­x 2 root root 4096 Dec 5 10:47 OpenCL

drwxr ­xr­x 3 root root 4096 Dec 5 10:47 rules

*Listing 5­1: Checking a file’s permissions with the long listing command* On each line, we get information about:

➊ The file type

➋ The permissions on the file for owner, groups, and users, respectively

➌ The number of links (This topic is beyond the scope of the book.)

➍ The owner of the file

➎ The size of the file in bytes

➏ When the file was created or last modified

➐ The name of the file

For now, let’s focus on the seemingly incomprehensible strings of letters and dashes on the left edge of each line. They tell us whether an item is a file or directory and what permissions, if any, are on it.

The first character tells you the file type, where d stands for a directory and a dash (–) indicates a file. These are the two most common file types.

The next section defines the permissions on the file. There are three sets of three characters, made of some combination of read (r), write (w), and execute (x), in that order. The first set represents the permissions of the owner; the second, those of the

group; and the last, those of all other users.

Regardless of which set of three letters you’re looking at, if you see an r first, that user or group of users has permission to open and read that file or directory. A w as the middle letter means they can write to (modify) the file or directory, and an x at the end means they can execute (or run) the file or directory. If any r, w, or x is replaced with a dash (-), then the respective permission hasn’t been given. Note that users can have permission to execute only either binaries or scripts.

Let’s use the third line of output in isting 5­1 as an example:

现在，让我们关注每条线左边看似难以理解的字母和短划线。它们告诉我们一个项目是文件还是目录，以及它上面有哪些权限（如果有的话）。

第一个字符告诉您文件类型，其中d代表目录，短划线（ - ）表示文件。这是两种最常见的文件类型。

下一节定义文件的权限。有三组三个字符，由read（r），write（w）和execute（x）的某种组合按顺序组成。第一组代表所有者的权限;第二组代表所有者的权限

group;以及最后一个，所有其他用户的那些。

无论您正在查看哪一组三个字母，如果您首先看到r，该用户或用户组都有权打开和读取该文件或目录。作为中间字母的w意味着它们可以写入（修改）文件或目录，并且最后的x意味着它们可以执行（或运行）文件或目录。如果用短划线（ - ）替换任何r，w或x，则未给出相应的权限。请注意，用户只能执行二进制文件或脚本的权限。

让我们使用isting 51中的第三行输出作为示例：

­rw­r­­r­­ 1 root root 33685504 June 28 2018 hashcat.hcstat

The file is called, as we know from the right end of the line, *hashcat.hcstat*. After the

initial – (which indicates it’s a file), the permissions rw- tell us that the owner has read and write permissions but not execute permission.

The next set of permissions (r--) represents those of the group and shows that the group has read permission but not write or execute permissions. And, finally, we see that the rest of the users also have only read permission (r--).

These permissions aren’t set in stone. As a root user or file owner, you can change them. Next, we’ll do just that.

正如我们从行的右端hashcat.hcstat所知，调用该文件。 之后

initial - （表示它是一个文件），权限rw-告诉我们所有者具有读写权限但没有执行权限。

下一组权限（r--）表示该组的权限，并显示该组具有读取权限但不具有写入或执行权限。 最后，我们看到其他用户也只有读取权限（r--）。

这些权限不是一成不变的。 作为root用户或文件所有者，您可以更改它们。 接下来，我们会做到这一点。

##### CHANGING PERMISSIONS

We can use the Linux command chmod (or change mode) to change the permissions. Only a root user or the file’s owner can change permissions.

In this section, we use chmod to change permissions on *hashcat.hcstat* using two

different methods. First we use a numerical representation of permissions, and then we use a symbolic representation.

我们可以使用Linux命令chmod（或更改模式）来更改权限。 只有root用户或文件所有者才能更改权限。

在本节中，我们使用chmod使用两个来更改hashcat.hcstat的权限

不同的方法。 首先，我们使用权限的数字表示，然后我们使用符号表示。

Changing Permissions with Decimal Notation

We can use a shortcut to refer to permissions by using a single number to represent one rwx set of permissions. Like everything underlying the operating system, permissions are represented in binary, so ON and OFF switches are represented by 1 and 0, respectively. You can think of the rwx permissions as three ON/OFF switches, so when all permissions are granted, this equates to 111 in binary.

A binary set like this is then easily represented as one digit by converting it into *octal*, an eight­digit number system that starts with 0 and ends with 7. An octal digit represents a set of three binary digits, meaning we can represent an entire rwx set with

one digit. able 5­1 contains all possible permission combinations and their octal and

binary representatives.

我们可以使用快捷方式通过使用单个数字来表示一个rwx权限集来引用权限。 与操作系统下的所有内容一样，权限以二进制表示，因此ON和OFF开关分别由1和0表示。 您可以将rwx权限视为三个ON / OFF开关，因此当授予所有权限时，这相当于111的二进制。

这样的二进制集很容易通过将其转换为八进制来表示为一位数，八位数字系统以0开头并以7结尾。八进制数字表示一组三位二进制数字，这意味着我们可以表示整个rwx集同

一位数。 51包含所有可能的权限组合及其八进制和

二进制代表。

**Table 5­1:** Octal and Binary Representations of Permissions



BinaryOctalrwx

000 0

---

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 001 | 1 | --x |  |  |
|  | 010 | 2 | -w- |  |
|  | 011 | 3 | -wx |  |
|  | 100 | 4 | r-- |  |
|  | 101 | 5 | r-x |  |
|  | 110 | 6 | rw- |  |
|  | 111 | 7 | rwx |  |
|  |  |  |  |  |  |
|  |  | | |  |  |

Using this information, let’s go through some examples. First, if we want to set only the read permission, we could consult able 5­1 and locate the value for read:



r w x 4 ­ ­

Next, if we want to set the permission to wx, we could use the same methodology and look for what sets the w and what sets the x:

r w x

­ 2 1

Notice in able 5­1 that the octal representation for -wx is 3, which not so coincidently

happens to be the same value we get when we add the two values for setting w and x

individually: 2 + 1 = 3.

Finally, when all three permissions are on, it looks like this:

r w x 4 2 1

And 4 + 2 + 1 = 7. Here, we see that in Linux, when all the permission switches are on, they are represented by the octal equivalent of 7.

So, if we wanted to represent all permissions for the owner, group, and all users, we could write it as follows:

7 7 7

Here’s where the shortcut comes in. By passing chmod three octal digits (one for each rwx

set), followed by a filename, we can change permissions on that file for each type of user. Enter the following into your command line:

kali >**chmod 774 hashcat.hcstat**

Looking at able 5­1, we can see that this statement gives the owner all permissions, the

group all permissions, and everyone else (other) only the read permission.

Now we can see whether those permissions have changed by running **ls -l** on the directory and looking at the *hashcat.hcstat* line. Navigate to the directory and run that command now:

kali >**ls -l**

total 32952

drwxr­xr­x 5 root root 4096 Dec 5 10:47 charsets

➊ ­rwxrwxr­­ 1 root root 33685504 June 28 2018 hashcat.hcstat

­rw­r­­r­­ 1 root root 33685504 June 28 2018 hashcat.hctune

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| drwxr ­xr­x 2 | root | root | 4096 | Dec 5 10:47 | masks |
| drwxr ­xr­x 2 | root | root | 4096 | Dec 5 10:47 | OpenCL |
| drwxr ­xr­x 3 | root | root | 4096 | Dec 5 10:47 | rules |

You should see -rwxrwxr-- on the left side of the *hashcat.hcstat* line ➊. This confirms that the chmod call successfully changed permissions on the file to give both the owner and the group the ability to execute the file.

Changing Permissions with UGO

Although the numeric method is probably the most common method for changing permissions in Linux, some people find chmod’s symbolic method more intuitive—both methods work equally well, so just find the one that suits you. The symbolic method is

often referred to as the *UGO* syntax, which stands for *user* (or owner), *group*, and *others*.

UGO syntax is very simple. Enter the chmod command and then the users you want to change permissions for, providing u for user, g for group, or o for others, followed by one of three operators:

虽然数值方法可能是在Linux中更改权限的最常用方法，但有些人发现chmod的符号方法更直观 - 两种方法同样有效，所以只需找到适合你的方法。 符号方法是

通常称为UGO语法，代表用户（或所有者），组和其他人。

UGO语法非常简单。 输入chmod命令，然后输入要更改权限的用户，为用户提供u，为组提供g，为其他用户提供o，或者输入三个运算符之一：

**-** Removes a permission

**+** Adds a permission

**=** Sets a permission

After the operator, include the permission you want to add or remove (rwx) and, finally, the name of the file to apply it to.

So, if you want to remove the write permission from the user that the file *hashcat.hcstat* belongs to, you could enter the following:

在运算符之后，包括要添加或删除的权限（rwx），最后包含要应用它的文件的名称。

因此，如果要删除用户对hashcat.hcstat所属文件的写入权限，可以输入以下内容：

kali >**chmod u-w hashcat.hcstat**

This command says to remove (-) the write (w) permission from *hashcat.hcstat* for the user (u).

Now when you check the permissions with ls –l again, you should see that the *hashcat.hcstat* file no longer has write permission for the user:

kali >**ls -l**

total 32952

drwxr­xr­x 5 root root 4096 Dec 5 10:47 charsets

­r­xr­xr­­ 1 root root 33685504 June 28 2018 hashcat.hcstat

­rw­r­­r­­ 1 root root 33685504 June 28 2018 hashcat.hctune

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| drwxr ­xr­x 2 | root | root | 4096 | Dec 5 10:47 masks |
| drwxr ­xr­x 2 | root | root | 4096 | Dec 5 10:47 OpenCL |
| drwxr ­xr­x 3 | root | root | 4096 | Dec 5 10:47 rules |

You can also change multiple permissions with just one command. If you want to give both the user and other users (not including the group) execute permission, you could enter the following:

您还可以使用一个命令更改多个权限。 如果要同时为用户和其他用户（不包括组）授予执行权限，可以输入以下内容：

chmod u+x, o+x hashcat.hcstat

This command tells Linux to add the execute permission for the user as well as the execute permission for others for the *hashcat.hcstat* file.

Giving Root Execute Permission on a New Tool

As a hacker, you’ll often need to download new hacking tools, but Linux automatically assigns all files and directories default permissions of 666 and 777, respectively. This means that, by default, you won’t be able to execute a file immediately after

downloading it. If you try, you’ll usually get a message that says something like “Permission denied.” For these cases, you’ll need to give yourself root and execute

permissions using chmod in order to execute the file.

For example, say we download a new hacker tool called newhackertool and place it into the root user’s directory (*/*).

作为黑客，您经常需要下载新的黑客工具，但Linux会自动分配所有文件和目录的默认权限666和777。 这意味着，默认情况下，您将无法立即执行文件

下载它。 如果你尝试，你通常会得到一条消息，上面写着“Permission denied。”。对于这些情况，你需要给自己一个root并执行

使用chmod执行文件的权限。

例如，假设我们下载了一个名为newhackertool的新黑客工具，并将其放入root用户的目录（/）中。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| kali >**ls -l** |  | | | | | | |
| total 80 |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Desktop |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Documents |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Downloads |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Music |
| ­rw­r­­r­­ 1 root root 1072 Dec 5 11.17 newhackertool➊ | | | | | | | |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Pictures |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Public |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Templates |
| drwxr­xr­x | 7 | root | root | 4096 | Dec 5 | 11.17 | Videos |

We can see *newhackertool* at ➊, along with the rest of the contents of the root directory. We can see that our *newhackertool* doesn’t have execute permission for anyone. This makes it impossible to use. It might seem strange that by default, Linux won’t let you execute a file you downloaded, but overall this setting makes your system more secure.

我们可以在➊看到newhackertool，以及根目录的其他内容。 我们可以看到我们的newhackertool没有任何人的执行权限。 这使得无法使用。 可能看起来很奇怪，默认情况下，Linux不允许您执行下载的文件，但总体而言，此设置使您的系统更安全。

We can give ourselves permission to execute *newhackertool* by entering the following:

kali >**chmod 766 newhackertool**

Now, when we perform a long listing on the directory, we can see that our *newhackertool* has execute permission for the owner:

kali >**chmod 766 newhackertool**

kali >**ls -l**

total 80

­­*snip*­­

drwxr­xr­x 7 root root 4096 Dec 5 11.17 Music

­rwxrw­rw­ 1 root root 1072 Dec 5 11.17 newhackertool

drwxr­xr­x 7 root root 4096 Dec 5 11.17 Pictures

­­*snip*­­

As you now understand, this grants us (as the owner) all permissions, including execute, and grants the group and everyone else only read and write permissions (4 + 2

= 6).

正如您现在所理解的，这将授予我们（作为所有者）所有权限，包括执行权限，并授予该组以及其他所有人只有读写权限（4 + 2）

= 6）。

##### SETTING MORE SECURE DEFAULT PERMISSIONS WITH MASKS

As you have seen, Linux automatically assigns base permissions—usually 666 for files and 777 for directories. You can change the default permissions allocated to files and directories created by each user with the umask (or unmask) method. The umask method

represents the permissions you want to *remove* from the base permissions on a file or directory to make them more secure.

The umask is a three­digit decimal number corresponding to the three permissions digits, but the umask number is *subtracted* from the permissions number to give the new permissions status. This means that when a new file or directory is created, its

permissions are set to the default value minus the value in umask, as shown in igure 5­1.

如您所见，Linux自动分配基本权限 - 通常为文件666和目录777。 您可以使用umask（或unmask）方法更改分配给每个用户创建的文件和目录的默认权限。 umask方法

表示要从文件或目录的基本权限中删除的权限，以使其更安全。

umask是对应于三个权限数字的三位十进制数字，但是从权限编号中减去umask编号以提供新的权限状态。 这意味着当创建新文件或目录时，它

权限设置为默认值减去umask中的值，如图51所示。



*Figure 5­1: How a umask value of 022 affects the permissions on new files and directories*

For example, if the umask is set to 022, a new file with the original default permissions of 666 will now have the permissions 644, meaning the owner has both read and write permissions, and the group and all other users have only read permission.

In Kali, as with most Debian systems, the umask is preconfigured to 022, meaning the Kali default is 644 for files and 755 for directories.

The umask value is not universal to all users on the system. Each user can set a personal default umask value for the files and directories in their personal *.profile* file. To see the current value when logged on as the user, simply enter the command umask and note what is returned. To change the umask value for a user, edit the file

*/home/username/.profile* and, for example, add umask 007 to set it so only the user and members of the user’s group have permissions.

例如，如果umask设置为022，则具有原始默认权限666的新文件现在将具有权限644，这意味着所有者具有读取和写入权限，并且该组和所有其他用户仅具有读取权限。

在Kali中，与大多数Debian系统一样，umask预先配置为022，这意味着Kali默认值为644，文件为755，目录为755。

umask值对于系统上的所有用户都不是通用的。 每个用户都可以为其个人.profile文件中的文件和目录设置个人默认umask值。 要以用户身份登录时查看当前值，只需输入命令umask并记下返回的内容。 要更改用户的umask值，请编辑该文件

/home/username/.profile，例如，添加umask 007进行设置，以便只有用户和用户组的成员才具有权限。

##### SPECIAL PERMISSIONS

In addition to the three general­purpose permissions, rwx, Linux has three special permissions that are slightly more complicated. These special permissions are set user ID (or SUID), set group ID (or SGID), and sticky bit. I’ll discuss each in turn in the next three sections.

除了三个通用权限，rwx，Linux还有三个特殊权限，稍微复杂一些。 这些特殊权限是设置用户ID（或SUID），设置组ID（或SGID）和粘滞位。 我将在接下来的三个部分中依次讨论每个部分。

Granting Temporary Root Permissions with SUID

As you should know by now, a user can execute a file only if they have permission to execute that particular file. If the user only has read and/or write permissions, they cannot execute. This may seem straightforward, but there are exceptions to this rule.

You may have encountered a case in which a file requires the permissions of the root user during execution for all users, even those who are not root. For example, a file that allows users to change their password would need access to the */etc/shadow* file—the file that holds the users’ passwords in Linux—which requires root user privileges in order to execute. In such a case, you can temporarily grant the owner’s privileges to execute the file by setting the SUID bit on the program.

Basically, the SUID bit says that any user can execute the file with the permissions of the owner but those permissions don’t extend beyond the use of that file.

To set the SUID bit, enter a 4 before the regular permissions, so a file with a new resulting permission of 644 is represented as 4644 when the SUID bit is set.

正如您现在应该知道的那样，用户只有在有权执行该特定文件时才能执行该文件。如果用户只具有读取和/或写入权限，则无法执行。这可能看起来很简单，但这条规则有例外。

您可能遇到过这样一种情况：在执行期间，文件需要root用户的权限才能为所有用户，即使是非root用户。例如，允许用户更改其密码的文件需要访问/ etc / shadow文件 - 在Linux中保存用户密码的文件 - 需要root用户权限才能执行。在这种情况下，您可以通过在程序上设置SUID位来临时授予所有者的权限以执行该文件。

基本上，SUID位表示任何用户都可以使用所有者的权限执行该文件，但这些权限不会超出该文件的使用范围。

要设置SUID位，请在常规权限之前输入4，因此当设置SUID位时，具有644的新结果权限的文件表示为4644。

Setting the SUID on a file is not something a typical user would do, but if you want to do so, you’ll use the chmod command, as in chmod 4644 *filename*.

Granting the Root User’s Group Permissions SGID

SGID also grants temporary elevated permissions, but it grants the permissions of the file owner’s group, rather than of the file’s owner. This means that, with an SGID bit set,

someone without execute permission can execute a file if the owner belongs to the group that has permission to execute that file.

The SGID bit works slightly differently when applied to a directory: when the bit is set on

a directory, ownership of new files created in that directory goes to the directory creator’s group, rather than the file creator’s group. This is very useful when a directory is shared by multiple users. All users in that group can execute the file(s), not just a single user.

The SGID bit is represented as 2 before the regular permissions, so a new file with the resulting permissions 644 would be represented as 2644 when the SGID bit is set. Again, you would use the chmod command for this—for example, chmod 2644 *filename*.

SGID还授予临时提升权限，但它授予文件所有者组的权限，而不是文件所有者的权限。 这意味着，设置SGID位，

没有执行权限的人可以执行文件，如果所有者属于有权执行该文件的组。

应用于目录时，SGID位的工作方式略有不同：当该位置位时

在目录中，在该目录中创建的新文件的所有权将转到目录创建者的组，而不是文件创建者的组。 当多个用户共享目录时，这非常有用。 该组中的所有用户都可以执行文件，而不仅仅是单个用户。

SGID位在常规权限之前表示为2，因此当SGID位置位时，具有结果权限644的新文件将表示为2644。 同样，您可以使用chmod命令 - 例如，chmod 2644 filename。

The Outmoded Sticky Bit

The *sticky bit* is a permission bit that you can set on a directory to allow a user to delete or rename files within that directory. However, the sticky bit is a legacy of older Unix systems, and modern systems (like Linux) ignore it. As such, I will not discuss it further here, but you should be familiar with the term because you might hear it in the Linux world.

粘滞位是您可以在目录上设置的权限位，以允许用户删除或重命名该目录中的文件。 然而，粘性位是旧Unix系统的遗留物，现代系统（如Linux）忽略了它。 因此，我不会在这里进一步讨论，但你应该熟悉这个术语，因为你可能会在Linux世界中听到它。

Special Permissions, Privilege Escalation, and the Hacker

As a hacker, these special permissions can be used to exploit Linux systems through *privilege escalation*, whereby a regular user gains root or sysadmin privileges and the associated permissions. With root privileges, you can do anything on the system.

One way to do this is to exploit the SUID bit. A system administrator or software developer might set the SUID bit on a program to allow that program access to files with root privileges. For instance, scripts that need to change passwords often have the SUID

bit set. You, the hacker, can use that permission to gain temporary root privileges and do something malicious, such as get access to the passwords at */etc/shadow*.

Let’s look for files with the SUID bit set on our Kali system to try this out. Back in Chapter 1

, I introduced you to the find command. We’ll use its power to find files with the SUID bit set.

As you’ll remember, the find command is powerful, but the syntax is bit more complicated than some of the other location commands, such as locate and which. Take a

moment to review the find syntax in hapter 1, if you need to.

In this case, we want to find files anywhere on the filesystem, for the root user or other sysadmin, with the permissions 4000. To do this, we can use the following find command:

作为黑客，这些特殊权限可用于通过权限提升来利用Linux系统，从而使普通用户获得root或sysadmin权限以及相关权限。使用root权限，您可以在系统上执行任何操作。

一种方法是利用SUID位。系统管理员或软件开发人员可以在程序上设置SUID位，以允许该程序访问具有root权限的文件。例如，需要更改密码的脚本通常具有SUID

位设置。作为黑客，您可以使用该权限获取临时root权限并执行恶意操作，例如访问/ etc / shadow中的密码。

让我们在我们的Kali系统上查找设置了SUID位的文件来试试这个。回到第1章

，我向你介绍了find命令。我们将使用它的功能来查找SUID位设置的文件。

你会记得，find命令功能强大，但语法比其他一些定位命令要复杂一些，比如locate和which。拿一个

如果需要，请查看第1章中的查找语法。

在这种情况下，我们希望在文件系统的任何位置查找文件，对于root用户或其他sysadmin，具有权限4000.为此，我们可以使用以下find命令：

kali >**find / -user root -perm -4000**

With this command, we ask Kali to start looking at the top of the filesystem with the / syntax. It then looks everywhere below */* for files that are owned by root, specified with user root, and that have the SUID permission bit set (-perm -4000).

When we run this command, we get the output shown in isting 5­2.

使用此命令，我们要求Kali开始使用/ syntax查看文件系统的顶部。 然后，它会在/下面找到由root拥有的文件，用户root指定的文件以及设置了SUID权限位的文件（-perm -4000）。

当我们运行此命令时，我们得到输出显示在isting 52。

/usr/bin/chsh

/usr/bin/gpasswd

/usr/bin/pkexec

/usr/bin/sudo

/usr/bin/passwd

/usr/bin/kismet\_capture

­­*snip*­­

*Listing 5­2: Finding files with the SUID bit set*

The output reveals numerous files that have the SUID bit set. Let’s navigate to the

*/usr/bin* directory, where many of these files reside, and then run a long listing on that

directory and scroll down to the *sudo* file, as shown in isting 5­3.

输出显示了许多具有SUID位设置的文件。 让我们导航到

/ usr / bin目录，其中包含许多这些文件，然后在其上运行一个长列表

目录并向下滚动到sudo文件，如isting 53所示。

kali >**cd /usr/bin**

kali >**ls -l**

­­*snip*­­

­rwxr­xr­x 1 root root 176272 Jul 18 2018 stunnel4

­rwxr­xr­x 1 root root 26696 Mar 17 2018 sucrack

➊ ­rwsr­xr­x 1 root root 140944 Jul 5 2018 sudo

­­*snip*­­

*Listing 5­3: Identifying files with the SUID bit set*

Note that at ➊, the first set of permissions—for the owner—has an s in place of the x. This is how Linux represents that the SUID bit is set. This means that anyone who runs

the *sudo* file has the privileges of the root user, which can be a security concern for the sysadmin and a potential attack vector for the hacker. For instance, some applications need to access the */etc/shadow* file to successfully complete their tasks. If the attacker can gain control of that application, they can use that application’s access to the passwords on a Linux system.

Linux has a well­developed system of security that protects files and directories from unauthorized access. The aspiring hacker needs to have a basic understanding of this system not only to protect their files but also to execute new tools and files. In some cases, hackers can exploit the SUID and SGID permissions to escalate privileges from a

regular user to a root user.

请注意，在➊，所有者的第一组权限 - 具有s代替x。 这就是Linux表示SUID位已设置的方式。 这意味着任何人都跑

sudo文件具有root用户的权限，这可能是系统管理员的安全问题，也可能是黑客的潜在攻击媒介。 例如，某些应用程序需要访问/ etc / shadow文件才能成功完成其任务。 如果攻击者可以控制该应用程序，他们可以使用该应用程序访问Linux系统上的密码。

Linux具有良好的安全系统，可保护文件和目录免受未经授权的访问。 有抱负的黑客需要对该系统有基本的了解，不仅要保护他们的文件，还要执行新的工具和文件。 在某些情况下，黑客可以利用SUID和SGID权限来升级来自的权限

普通用户到root用户。

##### SUMMARY

Linux’s use of permissions to protect a user’s or group’s files and directories from other users in the system can be used for offensive and defensive purposes. You should now know how to manage these permissions and how to exploit weak points in this security system—in particular, SUID and SGID bits.

Linux使用权限来保护用户或组的文件和目录免受系统中其他用户的攻击，可用于攻击性和防御性目的。 您现在应该知道如何管理这些权限以及如何利用此安全系统中的弱点 - 特别是SUID和SGID位。

**EXERCISES**

Before you move on to hapter 6, put the knowledge you learned from this chapter to the test by completing the following exercises:

1. Select a directory and run a long listing on it. Note the permissions

on the files and directories.

1. Select a file you don’t have permission to execute and give yourself execute permissions using the chmod command. Try using both the numeral method (777) and the UGO method.
2. Choose another file and change its ownership using chown.
3. Use the find command to find all files with the SGID bit set.

## 6

### PROCESS MANAGEMENT



At any given time, a Linux system typically has hundreds, or sometimes even thousands, of processes running simultaneously. A *process* is simply a program that’s running and using resources. It includes a terminal, web server, any running commands, any databases, the GUI interface, and much more. Any good Linux administrator—and particularly a hacker—needs to understand how to manage their processes to optimize their systems. For example, once a hacker takes control of a target system, they might want to find and stop a certain process, like an antivirus application or firewall. To do so, the hacker would first need to know how to find the process. The hacker might also want to set a scanning script to run periodically to find vulnerable systems, so we’ll also look at how to schedule such a script.

In this chapter, you’ll learn to manage those processes. First, you’ll learn to view and find processes and how to discover which processes are using the most resources. Then, you’ll learn to manage processes by running them in the background, prioritizing them, and killing them if necessary (no blood involved). Finally, you’ll learn to schedule processes to run on specified days and dates and at specific times.

在任何给定时间，Linux系统通常同时运行数百个，有时甚至数千个进程。进程只是一个正在运行和使用资源的程序。它包括终端，Web服务器，任何正在运行的命令，任何数据库，GUI界面等等。任何优秀的Linux管理员 - 特别是黑客 - 都需要了解如何管理他们的流程以优化他们的系统。例如，一旦黑客控制了目标系统，他们可能想要找到并停止某个进程，如防病毒应用程序或防火墙。为此，黑客首先需要知道如何找到该过程。黑客可能还想设置一个扫描脚本来定期运行以查找易受攻击的系统，因此我们还将研究如何安排这样的脚本。

在本章中，您将学习如何管理这些过程。首先，您将学习查看和查找流程以及如何发现哪些流程使用的资源最多。然后，您将学习如何通过在后台运行流程来管理流程，确定流程优先级，并在必要时将其删除（不涉及血液）。最后，您将学习如何安排在指定日期和日期以及特定时间运行的流程。

##### VIEWING PROCESSES

In most cases, the first step in managing processes is to view what processes are running on your system. The primary tool for viewing processes—and one of the Linux

administrator’s best friends—is the ps command. Run it in your command line to see what processes are active:

在大多数情况下，管理流程的第一步是查看系统上正在运行的进程。 用于查看进程的主要工具 - 以及Linux之一

管理员最好的朋友 - 是ps命令。 在命令行中运行它以查看哪些进程处于活动状态：

kali >**ps**

PID TTY TIME CMD

39659 pts/0 00:00:01 bash

39665 pts/0 00:00:00 ps

The Linux *kernel*, the inner core of the operating system that controls nearly everything, assigns a unique *process ID (PID)* to each process sequentially, as the processes are created. When working with these processes in Linux, you often need to specify their PIDs, so it is far more important to note the PID of the process than the name of the process.

Alone, the ps command doesn’t really provide you with much information. Running the

ps command without any options lists the processes started (said to be *invoked*) by the

currently logged­in user (in our case, root) and what processes are running on that terminal. Here, it simply says that the bash shell is open and running and that *we* ran

the ps command. We want and need far more information than that, particularly on

those processes run by other users and by the system in the background. Without this information, we know very little of what is actually taking place on our system.

Running the ps command with the options aux will show *all* processes running on the

system for *all* users, as shown in isting 6­1. Note that you don’t prefix these options

with a dash (-) and that everything is in lowercase; because Linux is case­sensitive, using uppercase options woud give you significantly different results.

Linux内核是控制几乎所有内容的操作系统的内核，在创建进程时按顺序为每个进程分配一个唯一的进程ID（PID）。在Linux中使用这些进程时，通常需要指定它们的PID，因此注意进程的PID比进程的名称更重要。

单独，ps命令并没有真正为您提供太多信息。跑步了

没有任何选项的ps命令列出了由...启动（称为调用）的进程

当前登录的用户（在我们的例子中是root）以及该终端上正在运行的进程。在这里，它只是说bash shell是打开并运行的，我们跑了

ps命令。我们想要并且需要更多的信息，尤其是关于

这些进程由其他用户和后台系统运行。如果没有这些信息，我们对系统实际发生的情况知之甚少。

使用选项aux运行ps命令将显示在其上运行的所有进程

系统适用于所有用户，如图61所示。请注意，您不要在这些选项前添加前缀

使用短划线（ - ）并且所有内容都是小写的;因为Linux是大小写敏感的，所以使用大写选项会给你带来截然不同的结果。

kali >**ps aux**

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

Root 1 0.0 0.4 202540 6396 ? Ss Apr24 0:46 /sbin/init

Root 2 0.0 0.0 0 0 ? S Apr24 0:00 [kthreadd]

Root 3 0.0 0.0 0 0 ? S Apr24 0:26 [ksoftirqd/0]

­­*snip*­­

root 39706 0.0 0.2 36096 3204 pts/0 R+ 15:05 0:00 ps aux

*Listing 6­1: Using the aux options to see processes for all users*

As you can see, this command now lists so many processes, they likely run off the bottom of your screen. The first process is init, listed in the final column, and the last

process is the command we ran to display, ps aux. Many of the details (PID, %CPU, TIME, COMMAND, and so on) may be different on your system but should have the same format. For our purposes, here are the most important columns in this output:

**USER** The user who invoked the process

**PID** The process ID

**%CPU** The percent of CPU this process is using

**%MEM** The percent of memory this process is using

**COMMAND** The name of the command that started the process

In general, to perform any action on a process, we must specify its PID. Let’s see how to use this identifier to our advantage.

Filtering by Process Name

When we inquire about or perform an action on processes, we usually don’t want all of the processes displayed on the screen. It’s simply a problem of too much information. Most often, we want to find information on a *single* process. To do so, we can use the

filtering command grep, which I introduced in hapter 1.

To demonstrate, we’ll use the Metasploit exploitation framework, the most widely used exploitation framework and nearly every hacker’s good friend. This comes installed on your Kali system, so start Metasploit with the following:

kali >**msfconsole**

Once the exploitation framework has been started, let’s see whether we can find it in the list of processes. To do so, use the ps aux command and then pipe it (|) to grep

looking for the string msfconsole, as in isting 6­2.

kali >**ps aux | grep msfconsole**

root 39756 0.0 0.0 4304 716 pts/2 Ss+ 15:13 0:00 sh ­c service postgresql start && msfdb init & msfconsole

root 39759 35.1 15.2 4304 227888 pts/2 Sl+ 15:13 1:36 ruby /usr/bin/ msfconsole

root 39892 0.0 0.0 4304 940 pts/2 S+ 15:18 0:00 grep msfconsole

*Listing 6­2: Filtering a ps search to find a particular process*

From the filtered output in this listing, you should see all the processes that match the term msfconsole. The PostgreSQL database, which is the database Metasploit uses, is

shown first, then the msfconsole program itself from */usr/bin/msfconsole*. Finally, you

should see the grep command you used to look for msfconsole. Notice that the output did not include the column header list from ps. Since the keyword, msfconsole, is not in the header, it is not displayed. Even so, the results are displayed in the same format.

From this, you can learn some important information. If, for example, you need to know how many resources Metasploit is using, you can consult the third column (the CPU column), to see that it’s using 35.1 percent of your CPU, and consult the fourth column to see that it’s using 15.2 percent of your system memory. That’s quite a bit. It’s a demanding beast!

Finding the Greediest Processes with top

When you enter the ps command, the processes are displayed in the order they were

started, and since the kernel assigns PIDs in the order they have started, what you see are processes ordered by PID number.

In many cases, we want to know which processes are using the *most* resources. This is where the top command comes in handy because it displays the processes ordered by

resources used, starting with the largest. Unlike the ps command, which gives us a one­

time snapshot of the processes, top refreshes the list dynamically—by default, every 10 seconds. You can watch and monitor those resource­hungry processes, as shown in

isting 6­3.

kali >**top**

top ­ 15:31:17 up 2 days, ^;50, 4 users, load average: 0.00, 0.04, 0.09

Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie

%Cpu(s): 1.3 us, 0.7 sy, ).) ni, 97.4 id, 0.0 wa, 0.0 hi 0.0 si 0.0

KiB Mem : 1491220 total, 64848 free, 488272 used, 938100 buff/cache

KiB Swap : 1046524 total, 1044356 free, 2168 used. 784476 avail MEM

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

|  |  |  |
| --- | --- | --- |
| 39759 root 20 | 0 | 893180 247232 11488 S 0.7 16.6 1:47.88 ruby |
| 39859 root 20 | 0 | 27308 16796 14272 S 0.3 1.2 1:47.88 postgres |
| 39933 root 20 | 0 | 293936 61500 29108 S 0.7 4.1 1:47.88 Xorg |
| ­­*snip*­­ |  |  |

*Listing 6­3: Finding the greediest processes with top*

System administrators often keep top running in a terminal to monitor use of process resources. As a hacker, you may want to do the same, especially if you have multiple tasks running on your system. While you have top running, pressing the H or ? key will

bring up a list of interactive commands, and pressing Q will quit top. You’ll use top again

soon to manage your processes in “

hanging Process Priority with nice” on

age 65 and

“ illing Processes” on age 66.

##### MANAGING PROCESSES

Hackers often need to multiprocess, and an operating system like Kali is ideal for this. The hacker may have a port scanner running while running a vulnerability scanner and an exploit simultaneously. This requires that the hacker manage these processes efficiently to best use system resources and complete the task. In this section, I'll show you how to manage multiple processes.

Changing Process Priority with nice

You don’t often hear the word *nice* used in the context of hackers, but here you will. The

nice command is used to influence the priority of a process to the kernel. As you saw when we ran the ps command, numerous processes run on the system at once, and all of

them are contending for the available resources. The kernel will have final say over the priority of a process, but you can use nice to *suggest* that a process should be elevated in priority.

The idea behind the use of the term *nice* is that, when you use it, you’re determining how “nice” you’ll be to *other* users: if your process is using most of the system resources, you aren’t being very nice.

The values for nice range from –20 to +19, with zero being the default value (see igure

­1). A high nice value translates to a low priority, and a low nice value translates to a high priority (when you’re not being so nice to other users and processes). When a process is started, it inherits the nice value of its parent process. The owner of the process can lower the priority of the process but cannot increase its priority. Of course, the superuser or root user can arbitrarily set the nice value to whatever they please.



*Figure 6­1: Niceness priority values*

When you start a process, you can set the priority level with the nice command and then alter the priority after the process has started running with the renice command. The syntax for these two commands is slightly different and can be confusing. The nice command requires that you *increment* the nice value, whereas the renice command wants an *absolute value* for niceness. Let’s look at an example to demonstrate this.

**Setting the Priority When Starting a Process**

For demonstration purposes, let’s assume we have a process named slowprocess that’s located at */bin/slowprocess*. If we wanted it to speed up its completion, we could start the process with the nice command:

kali >**nice -n -10 /bin/slowprocess**

This command would increment the nice value by -10, increasing its priority and allocating it more resources.

On the other hand, if we want to be nice to our fellow users and processes and give

slowprocess a lower priority, we could increment its nice value positively by 10: kali >**nice -n 10 /bin/slowprocess**

Give this a try on a process you have currently running and then run ps to see how it changes, if at all.

**Changing the Priority of a Running Process with renice**

The renice command takes absolute values between –20 and 19 and sets the priority to that particular level, rather than increasing or decreasing from the level at which it started. In addition, renice requires the PID of the process you are targeting rather than

the name. So, if slowprocess is using an inordinate amount of resources on your system

and you want to give it a lower priority, thus allowing other processes a higher priority and more resources, you could renice the slowprocess (which has a PID of 6996) and give

it a much higher nice value, like so:

kali >**renice 20 6996**

As with nice, only the root user can renice a process to a negative value to give it higher priority, but any user can be nice and reduce priority with renice.

You can also use the top utility to change the nice value. With the top utility running,

simply press the R key and then supply the PID and the nice value. isting 6­4 shows

the top utility running. When I press the R key and supply the PID and nice value, I get the following output:



*Listing 6­4: Changing a nice value when top is in use*

When I press the R key, I’m asked for the PID ➊ with the text renice PID [value] to value. The output should then change to reflect the new priorities.

Killing Processes

At times, a process will consume way too many system resources, exhibit unusual behavior, or—at worst—freeze. A process that exhibits this type of behavior is often referred to as a *zombie process*. For you, probably the most problematic symptom will be wasted resources used by the zombie that could be better allocated to useful processes.

When you identify a problematic process, you may want to stop it with the kill

command. There are many different ways to kill a program, and each has its own kill number.

The kill command has 64 different kill signals, and each does something slightly different. Here, we focus on a few you will likely find most useful. The syntax for the

kill command is kill-*signal PID*, where the signal switch is optional. If you don’t provide

a signal flag, it defaults to SIGTERM. able 6­1 lists the common kill signals

**Table 6­1:** Commonly Used Kill Signals



|  |  |  |  |
| --- | --- | --- | --- |
| Signal NumberDescription name for  option  SIGHUP 1 This is known as the Hangup (HUP) signal. It stops the designated process and restarts it with the same PID.  SIGINT 2 This is the Interrupt (INT) signal. It is a weak kill signal that isn’t guaranteed to work, but it works in most cases.  SIGQUIT 3 This is known as the core dump. It terminates the process and saves the process information in memory, and then it saves this information in the current working directory to a file named core. (The reasons for doing this are beyond the scope of this book.)  SIGTERM 15 This is the Termination (TERM) signal. It is the kill command’s default kill signal.  SIGKILL 9 This is the absolute kill signal. It forces the process to stop by sending the process’s resources to a special device, /dev/null. | | |  |
|  |
|  |
|  |  |  |  |

Using the top command, you can identify which processes are using too many resources;

often, those processes will be legitimate, but there may be malicious processes taking resources that you’ll want to kill.

If you just want to restart a process with the HUP signal, enter the -1 option with kill, like so:

kali >**kill -1 6996**

In the case of a zombie or a malicious process, you likely want to send the kill -9 signal,

the absolute kill signal, to the process. This makes certain that the process is terminated.

kali >**kill -9 6996**

If you don’t know a process’s PID, you can use the killall command to kill the process. This command takes the name of the process, instead of the PID, as an argument.

For example, you could terminate a hypothetical zombieprocess like this: kali >**killall -9 zombieprocess**

Finally, you can also terminate a process in the top command. Simply press the K key and then enter the PID of the offending process.

Running Processes in the Background

In Linux, whether you’re working from the command line or the GUI, you’re working within a shell. All commands that run are executed from within that shell, even if they run from the graphical interface. When you execute a command, the shell waits until the command is completed before offering another command prompt.

At times, you may want a process to run in the background, rather than having to wait for it to complete in that terminal. For instance, say we want to work on a script in a text editor and so have called our text editor (leafpad) by entering the following:

kali >**leafpad newscript**

In this case, the bash shell will open the leafpad text editor to create *newscript*. While we work in the text editor, the terminal is occupied with running the text editor. If we return to the terminal, we should see that it is running our text editor and that we have no new prompt to allow us to enter more commands.

We could, of course, open another terminal to run more commands, but a better option to save resources and screen real estate is to start the text editor running in the background. Running a process in the background simply means that it will continue to run without needing the terminal. In this way, the terminal is freed up for other duties.

To start the text editor in the background, just append an ampersand (&) to the end of the command, like so:

kali >**leafpad newscript &**

Now, when the text editor opens, the terminal returns a new command prompt so we can enter other commands on our system while also editing our *newscript*. This is effective for any process that may run for a significant length of time when you want use the terminal. As a hacker, you’ll find this useful for running multiple terminals with multiple tasks, to save resources and screen space.

Moving a Process to the Foreground

If you want to move a process running in the background to the foreground, you can use the fg (foreground) command. The fg command requires the PID of the process you want to return to the foreground, as shown next.

kali >**fg 1234**

If you don’t know the PID, you can use the ps command to find it.

##### SCHEDULING PROCESSES

Both Linux system administrators and hackers often need to schedule processes to run at a particular time of day. A system administrator might want to schedule a system backup to run every Saturday night at 2 AM, for example. A hacker might want to set a script to run to perform reconnaissance on a regular basis, finding open ports or vulnerabilities. In Linux, you can accomplish this in at least two ways: with at and crond.

The at command is a *daemon*—a background process—useful for scheduling a job to run once at some point in the future. The crond is more suited for scheduling tasks to

occur every day, week, or month, and we’ll cover this in detail in hapter 16.

We use the at daemon to schedule the execution of a command or set of commands in the future. The syntax is simply the at command followed by the time to execute the

Process. The time argument can be provided in various formats. most common at time formats.

**Table 6­2:** Time Formats Accepted by the at Command



able 6­2 contains the

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  |  |  |
| Time format  at 7:20pm | | Meaning  Scheduled to run at 7:20 PM on the current day |  |  |
| at 7:20pm June 25 | | Scheduled to run at 7:20 PM on June 25 |  |
| at noon | | Scheduled to run at noon on the current day |  |
| at noon June 25 | | Scheduled to run at noon on June 25 |  |
| at tomorrow | | Scheduled to run tomorrow |  |
| at now + 20 minutes | | Scheduled to run in 20 minutes from the current time |  |
| at now + 10 hours | | Scheduled to run in 10 hours from the current time |  |
| at now + 5 days | | Scheduled to run in five days from the current date |  |
| at now + 3 weeks | | Scheduled to run in three weeks from the current date |  |
| at 7:20pm 06/25/2019 | | Scheduled to run at 7:20 PM on June 25, 2019 |  |
|  |
|  |  | |  |  |



When you enter the at daemon with the specified time, at goes into interactive mode and you are greeted with an at> prompt. Here is where you enter the command you want executed at the specified time:

kali >**at 7:20am**

at >**/root/myscanningscript**

This code snippet will schedule *myscanningscript* to execute today at 7:20 AM.

##### SUMMARY

Managing processes in Linux is a key skill for every Linux user and hacker. You must be able to view, find, kill, prioritize, and schedule processes to manage your Linux instance optimally. A hacker often will need to find processes on the target they want to kill, such as the antivirus software or a firewall. They will also need to manage multiple processes in an attack and prioritize them.

管理Linux中的进程是每个Linux用户和黑客的关键技能。 您必须能够查看，查找，终止，确定优先级并安排进程以最佳方式管理Linux实例。 黑客通常需要在他们想要杀死的目标上找到进程，例如防病毒软件或防火墙。 他们还需要管理攻击中的多个进程并确定其优先级。

**EXERCISES**

Before you move on to hapter 7, try out the skills you learned from this

chapter by completing the following exercises:

* 1. Run the ps command with the aux options on your system and note which process is first and which is last.
  2. Run the top command and note the two processes using the greatest amount of your resources.
  3. Use the kill command to kill the process that uses the most resources.
  4. Use the renice command to reduce the priority of a running process to +19.
  5. Create a script called myscanning (the content is not important) with a text editor and then schedule it to run next Wednesday at 1 AM.