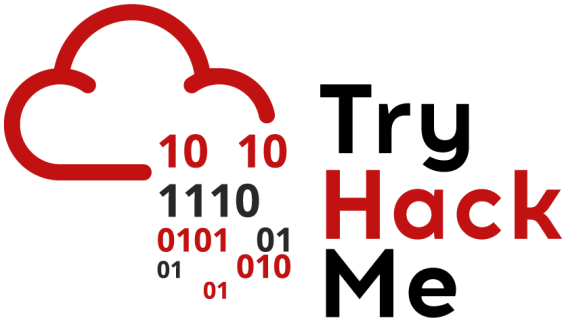


Today we'll be looking at using the Linux command line and understanding the basics of Linux. Please note, there's a lot more to Linux than what's on the supporting material. This guide is intended to help you start off with using the command line.

File Structure

Linux uses the filesystem hierarchy standard which defines the way directories/folders are laid out. It's important to understand this structure to understand how the operating system works. The top most directory is called the root directory(/). Since the file system is organised in a hierarchy, every file and folder is located under the root directory. Here's the most common layout of the root directory:

- **/bin** - contains programs in the form of binary files that user can run
- **/boot** - contains the files needed to start the system
- **/dev** - used to represent devices(mostly virtual) that correspond to particular functions
- **/etc** - contain configuration files for services on a computer and some system files
- **/usr** - contain executable files(programs) for most system programs
- **/home** - contains home directories for personal users
- **/lib** - contain libraries(which contain extra functions) that are used by executable files in the /bin directories
- **/var** - mostly contain files that store information about how services run(also known as log files)
- **/proc** - contains information about processes running on the system
- **/mnt** - used to mount file systems. Mounts are usually used when users want to access other file systems on their system
- **/opt** - contains optional software
- **/media** - contains removable hardware e.g. USB
- **/tmp** - contains temporary files. This folder is usually cleared on reboot so doesn't store persistent files
- **/root** - contains files created by the super user(more on this later)

Now that we have a brief explanation about the file structure, let's dive straight in. To learn basic commands, we'll assume the scenario of what an attacker would do if they gain access to a linux system

The first thing an attacker would do is gain access to the system. This can be done in many different ways, but the most common way is using a shell. A shell is used to run system commands(it can also be thought of as the terminal or as the actual command line). Shells are also programs and the most common shells are:

- /bin/sh
- /bin/bash

While the shells are quite similar, they have subtitles in the way they operate. Like mentioned earlier, a shell will be used to run programs(they can be stored anywhere on the file system, but in most cases, they will be stored in the /bin or /usr folder). You can run programs from the shell, but the shell would need to know where these programs are actually stored. It uses the $PATH variable to do this. Users can add and remove location on the path(we won't be doing this for this exercise).

Now that an attacker has a shell on the system(we'll assume that an attacker somehow got remote access to the machine as a particular user and log in as a user), they would commonly land in the user's home directory(in the location /home/username). Users store a lot of files in the home directory, so an attacker would want to see what files are there. They do this using the *ls* command:

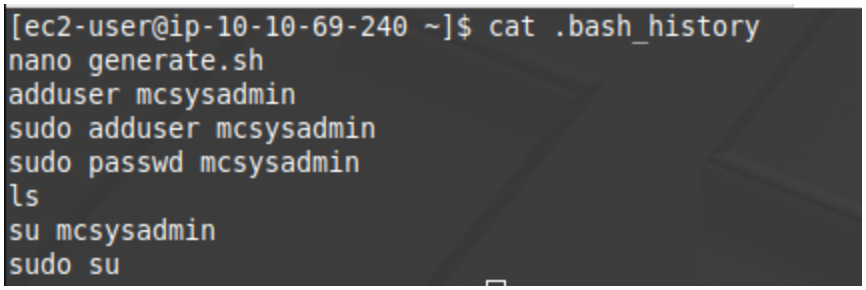
```
[ec2-user@ip-10-10-69-240 ~]$ ls -la
total 16
drwx------ 3 ec2-user ec2-user  95 Dec  4 08:04 .
drwxr-xr-x 4 root     root      40 Dec  4 07:27 ..
-rw------- 1 ec2-user ec2-user 108 Dec  4 08:04 .bash_history
-rw-r--r-- 1 ec2-user ec2-user  18 Jul 27  2018 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user 193 Jul 27  2018 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user 231 Jul 27  2018 .bashrc
drwx------ 2 ec2-user ec2-user  29 Dec  4 07:23 .ssh
```

Notice here that the command has other options(called flags). Flags pass in extra options to the command.

Linux commands can be very daunting, but the best thing about them is their *man pages*. A man page gives information about a particular command and how it is used. Most commands have man pages attached to them. If you want more information about *ls*, you just need to run *man ls*

As you can see, this page shows information about the command and various flags that can be used. To quit the man page, type *q.*

The next thing an attacker would do would try find files of interest. On common file of interest is the *.bash_history* file which keeps track of what commands the user has run. To list of the contents of a file, use the *cat filename* command. In this case, the output would be:



This file is useful to check as it gives an indication of what a user has been doing - sometimes a user may access a potentially interesting file or run commands.

Another file an attacker would access would be the /etc/passwd file. This file gives information about all the users on the machine. The format of the file would be:

username[1]:x[2]:userid[3]:groupid[4]:useridinfo[5]:/folder/location[6]:/shell/location[7]

- [1] username
- [2] usually an *x* character and is used to represent their passwords
- [3] User ID
- [4] Group ID
- [5] Extra comments about a user
- [6] Home Directory
- [7] Shell Location - Most actual users will use the aforementioned shells, but accounts can also belong to particular services. These services won't have paths to shells but files like /sbin/nologin which means that the user can't access this account through a shell(sometimes not at all)

Another note to mention is that the /etc/passwd file is world readable. This is a useful file to display as a proof of concept to show that you have access to a system. A similar such file is the /etc/shadow file that actually contains password hashes of the user.

An attacker may find a large file that they want to search - it could be source code(and source code can contain API keys and other sensitive information). An attacker could use the grep tool to extract a particular value from a file using the command *grep 'string' filename*. Grep is even more powerful and can recursively search through files with more powerful pattern matching using regex. We won't dive into regex but you can find some information on getting started here and here.

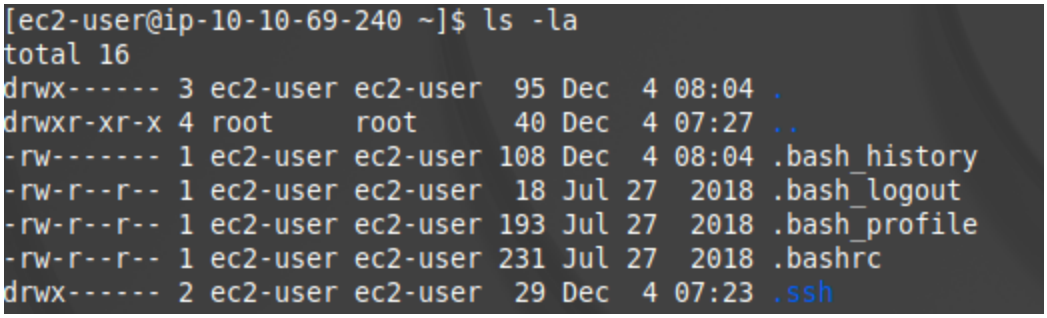
On computers, users would usually leave files lying around. Common files include credential files and backup files(with the extension .bak). A convenient way of searching for files uses the *find* command. A common break down of the command is
find location -name file_name

With this you can also use what we call wildcards. Filename usually have a particular format of: *Name.extension*
If you want to find all the files with the *jpg* extension, you can use **.jpg* as the filename where the * character represents every file and the .jpg extension represents a jpg file.

It's important for an attacker to be able to read, write and access particular files. To do this, they would have to check file permissions. Here's a breakdown of what Linux file permissions mean. File permissions are displayed as:
drwx rwx rwx username groupname number-of-files size date-last-modified file-name

A file can either **read**, **write** or **executable** permissions. The permissions are for
- the user who owns the files
- the user who is part of a group that has particular permissions
- anyone else who doesn't fall into those criteria

In that particular order. The first character can either represent a file(-) or a directory(d).



From the image above, for the *.bash_history* file, we can see that:

- The ec2-user user can read and write to the file
- Any user part of the ec2-user group doesn't have any access to the file
- Anyone else doesn't have access to the file

Like any computer system, there has to be an administrator user. In linux, this user is referred to as the super user and commonly has the username root. **The root user will always have an ID of 0.** Normal users can also be given administrator privileges, and this is shown in the /etc/sudoers file. The files contain entries in the form

user (host)=(user:group) commands

This means the user on the host name is allowed to run the command as a particular user.

Redirection Operators

This is more related to the user of Linux as opposed to what an attacker would do. Commands usually require input and produce output. Common operators include:

- >
    - This is used to redirect the output of a command to a file.
- <
    - This is used to provide input from a file to a command
- |
    - This is used to pass the output of one command to another

The best way to understand how redirection operators work is to use them in practise!