Canberra Computer Science Enrichment: Unix 1

Dirk Pattinson and Nicholas Miehlbradt

Apr 1, 2022

1 Basics

We're going to remote-login to a number of servers, solving challenges as we go along. Technically, we're using a secure shell connection which gives us a command line, and hence need to use command line tools.

Secure Shell - SSH. Let's start at the front door. If you have an account on a Unix system, the Secure Shell (SSH) protocol allows you to log in, transfer data, and execute commands on that system.

MacOS and Linux operating systems have SSH clients built in, but we're going to use the PuTTY tool on Windows.

If you want to find a computer system, you need to know either its name or address. Every computer has a unique address within its network; it may also have a unique IP address that can be accessed from any other computer on the Internet.

With an SSH client, you can use the IP address to specify which computer you'd like to connect to. Computers listen to the network over multiple ports. Commonly used ports include 80 (HTTP), 443 (HTTPS), and 22 (SSH). You can specify which port you'd like to connect to in Putty.

Log In. Go to `overthewire.org` and select the 'bandit' challenge and go to 'Level 0'. You see a hostname to log in to, a username, a password and a port. Use PuTTy to log in.

2 Basic Tools

Communication with the machine on the other side is via the *shell*. The shell reads commands that you type in, and executes it for you. For levels 1 to 3, you need:

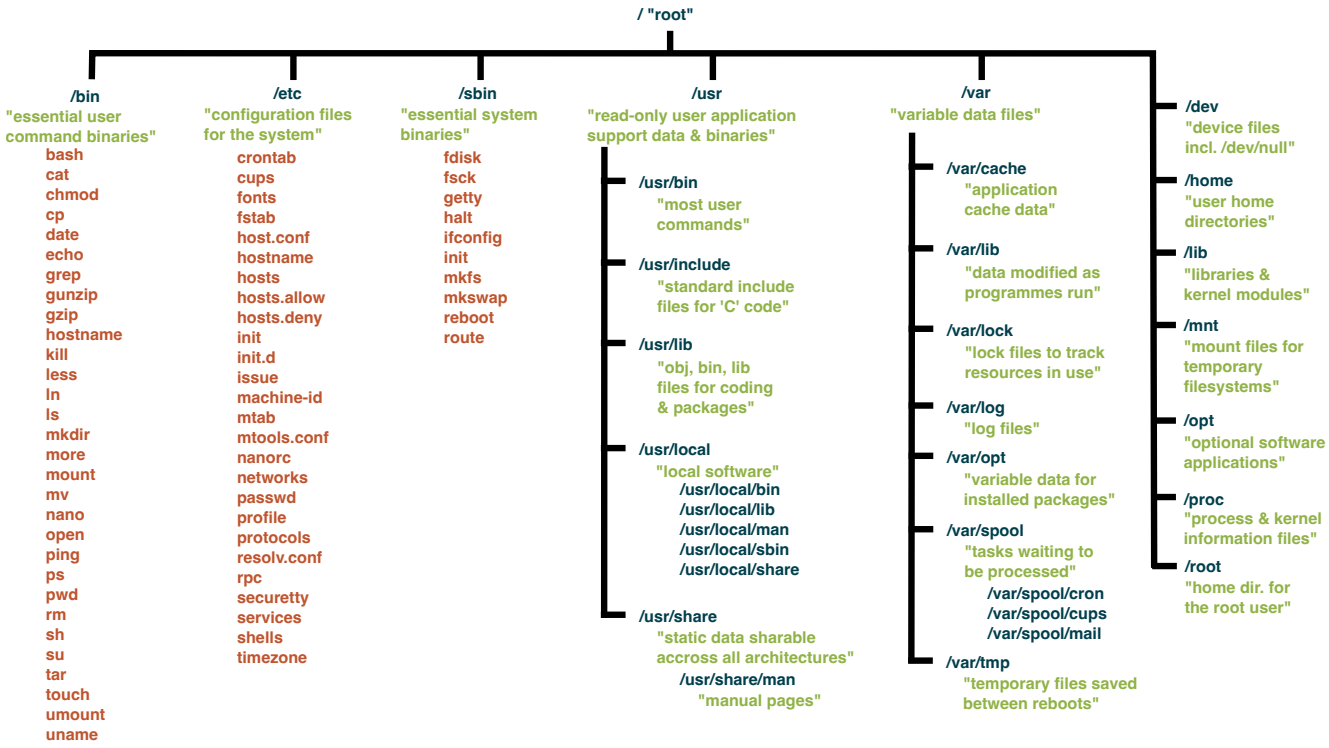
- `ls` – list the files in the current directory
- `cat` – print the content of a file to the console
- `man` – read the manual page.

Try `man ls` or `man cat` to see how these commands work (or google 'man ls'). You can also do `man man`!

Solve challenges to get to level 3 in bandit. You will need that the commands accept *options*, usually specified with a hyphen (try e.g. `ls -l` or `ls -a`). But there's a problem when a filename begins with a `-`, or when a filename contains spaces. For example `ls word with spaces` instructs the shell to list three different files, and you'd need to use a mechanism called 'quoting' to make the shell interpret it as a single file name (with spaces).

3 The File System

Almost all resources in a Unix system can be accessed through the filesystem. This includes data files (text, images, video, etc.), programs, hardware devices, and system information. The filesystem is organized into a tree of directories. Most Unix systems use a standard set of directory names to help users navigate around.



Filenames are case-sensitive and unique within a directory. Filenames do not usually contain spaces or other special characters as this can make them hard to deal with. If a filename starts with `.` it is a hidden file, requiring special effort to see or manipulate. The shell always 'stands' in a directory (called the 'working directory') and you can walk up and down the filesystem. For the next levels, you might want to use the following:

- `cd` – change directory
- `file` – analyse the file type
- `find` – search for a file with specific attributes

Solve. You should now be able to solve levels 4–7. One aspect that you might want to investigate is that the shell *interprets* the commands you give it. For example, it expands `*` to the list of all (non-hidden) files in a directory, and `a*` to the list of all files starting with 'a'. And `find` really accepts a lot of different options ....

4 Analysing Files

Information is often hidden in files. There are a number of tools that help to manipulate files, or extract information.

- `grep` – find a word in a file
- `sort` – sort files line by line, alphabetically
- `uniq` – remove duplicate lines and more

Solve. Try your hands and get to level 10. One trick is that `grep` can do more than just find words, the `re` in `grep` stands for regular expressions. And also, commands an be chained with a `|`. If we run `command1 | command2`, then `command2` will use the output of `command1` as its input.

5 Simple Coding and Decoding

By now, you don't really need our help any longer, and can solve the next levels using the hints on the `overthewire` level descriptions. The *tar* command creates or extracts an archive ...

Solve. The levels 11 to 13 should be eminently doable!

6 Public Key Cryptography

Secure shell, as many other cryptographic schemes, have two keys: the *public* key of a participant is used by others to encrypt messages, and the *private* key can then be used by the recipient for decryption. Instead of a password, you can also specify a secret (key) to log in with ssh.

We then use other ways to communicate with the server: feed in information at a specific port, either directly, or encrypted. And to see where in the first place where a server accepts a connection (is listening).

Solve. Try your hands at levels 14 – 18.

7 Shell Intricacies

The shell executes a number of commands when you log in. You can add to them by placing them in a `.bashrc`. What can we do if the commands in that file log you out immediately? Another intricacy are the permissions. You can make a command execute as a different user by setting the `s`-bit (setuid) bit. This can be exploited. Then, there are commands that we want to run on a regular basis, using `cron`. Google and the man pages will tell you everything about it!

Solve. Everything up to and including level 24!

8 Shell Programming

So far, we have just combined commands by connecting output and input. We can also write programs in shell! This amounts to putting all the commands into a text file, and have the special characters `#!/bin/bash` right at the beginning. If we then make the file executable, the machine knows (from the first line) that we want everything in the file executed via the shell (bash, in this example). Then again, the shell that we get at login might be different?

Solve. The levels up to and including 27 are yours now!

9 If you've made it this far ...

... you won't need our help to solve the rest of the challenges! Some of them involve `git` repositories. They are awesome for developing software, and using them with the command line is so much quicker.