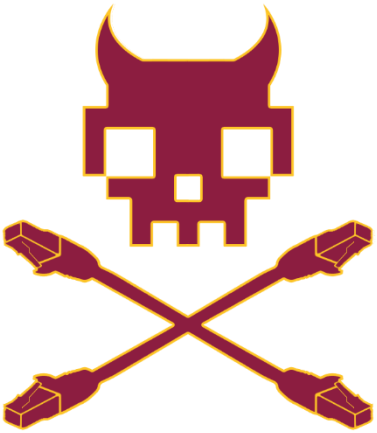
Week 1: Linux



If you haven't already join the slack, discord or the club sync itself:

Slack:
https://join.slack.com/t/devilsec/shared_invite/enQtNjMzNTQ3OTQyMDA0LTViOGYzYzUwOGEwMWU0N2EzNWZmYmVmM2QxMWIwMTBkZjk4NGI5ZjA3MjQwODNiOWZhMTQ0NzJmNTBmMjU4YTE

Discord: https://discord.gg/MjNpC2C

club: https://orgsync.com/181537/chapter

Table of Contents

Everything that we will need to deal with in linux is dealt with in the shell. What is the Shell? The shell is a program that will take in a command from your keyboard (or standard in/file descriptor 0 but more on that during the scripting section) and send them to the operating system to perform. The program you use to do these things are normally called things like Terminal or Console. I will only be talking about Bash (Bourne Again Shell) as an intro but there plenty of other like zsh, tsch, ksh, sh, etc.

File System

Linux's file system is organized in a hierarchal directory tree. The first directory in the tree is your root (/) directory.

Example directories:

- /
    - "Root" (not to be confused with the root user account or the /root/ directory). Base of the entire linux dir structure.
- /root/
    - "Slash root" – the home dir for the **root user** account.
- /etc/
    - "slash et-cee" host specific configs.
- /var/
    - "slash var" – variable data files – **SOMETHING THAT WE WILL USE A LOT HERE IS /VAR/LOG**
- /mnt/
    - "slash mount" – where devices (file systems) are connected to, usually temoporarily like CDROMS, usb drive, so on.


LINUX Directory Structure
Linux Directory
Function
/
The top directory of Linux
/bin
Store binary files which related to the system such as mount, ls, rm, etc
/boot
Store files related to boot process
/dev
Store information about all devices which connected to your Linux
/etc
Store configuration files about Linux and its application
/home
User directory
/lib
Store library files
/lost+found
lost+found is the directory in which fsck (filesystem check) will put files it restores from orphaned blocks
/media
Usually used as a moint point for external media such as CD/DVD ROM
/mnt
Used as a mount point directory, but it more likely a place that "temporarily mounted" device such as network shares.
/opt
Store files which not handled by package manager
/proc
A virtual filesystem which used to provide information about the system
/root
As root home directory
/sbin
Store a binary files which usually can be run by superuser only
/selinux
Store information about Security Enhanced. Some Linux distribution may not have this directory
/srv
Store data services which used by system
/sys
Store information related about your Linux system
/tmp
Used as a temporary folder for applications
/usr
Store user utilities and applications
/var
Store variable data files
LinOxide.com
More detailed : http://linoxide.com/how-tos/linux-directory-structure/

Linux Directory	Function	Comparison with Microsoft Windows 7
/	The top directory of Linux	C:\
/bin	Store binary files which related to the system such as mount, ls, rm, etc	C:\Windows
/boot	Store files related to boot process	C:\Windows
/dev	Store information about all devices which connected to your Linux	C:\Windows
/etc	Store configuration files about Linux and its application	C:\Windows
/home	User directory	My Documents
/lib	Store library files	C:\Windows\system
/lost+found	lost+found is the directory in which fsck (filesystem check) will put files it restores from orphaned blocks	Found.000
/media	Usually used as a moint point for external media such as CD/DVD ROM	D: or E: drives
/mnt	Used as a mount point directory, but it more likely a place that "temporarily mounted" device such as network shares.	A mapped drive such as X: , Y:, Z:
/opt	Store files which not handled by package manager	None
/proc	A virtual filesystem which used to provide information about the system	C:\Windows\system or C:\Windows\System32
/root	As root home directory	My Documents for Administrator
/sbin	Store a binary files which usually can be run by superuser only	C:\Windows
/selinux	Store information about Security Enhanced. Some Linux distributionmay not have this directory	None
/srv	Store data services which used by system	None
/sys	Store information related about your Linux system	C:\Windows\system or C:\Windows\System32
/tmp	Used as a temporary folder for applications	C:\Windows\Temp
/usr	Store user utilities and applications	C:\Program Files or C:\ProgramData
/var	Store variable data files	None

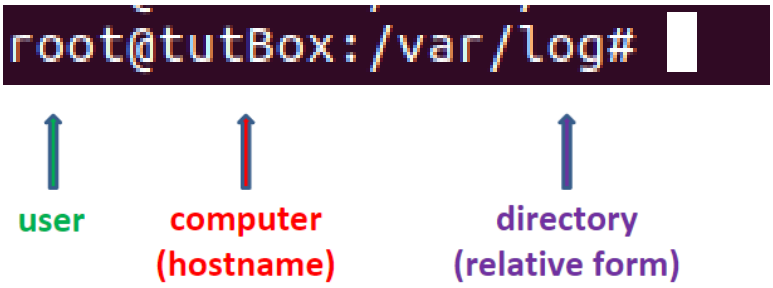
Paths

Absolute paths

When you work with directories you access them using an absolute or relative path.

Absolute paths start with the ***root directory***, which is /

All directories are children (sub directories) of the root dir, to some degree. For example lets go into the logs/ directory. This directory is a child of the var directory which is a child of the / directory. Using the cd command (change directory) we would type **FOR AN ABSOLUTE PATH :"**cd /var/log **"**



Relative Paths

Relative paths are based on your current location, with absolute paths your current directory doesn't matter. For example say you are in /var currently and want to go into log, for a relative path you can just type: "cd log" and it will take you there.

Remotely Accessing a system

SSH

Sometimes you need to log into a system that you don't have physical access to, this is where SSH or secure shell comes in. Using the basic structure of ssh username@host -p [PORT] you can remotely connect to a system that has port 22 open. SSH is secure because it provides "encrypted communication between two untrusted hosts over an insecure network." SSH can also allow for something called ssh tunneling/ port redirection but we will go over that more when we start learning hacking.

Telnet

THIS IS NOT SECURE. Telnet is a command for interactive communication with another host using the TELNET protocol. If left at the base config the communication is not encrypted and not secure, the commands you send over the internet can be sniffed and grabbed by a third party if anyone is listening. Starts in prompt mode "telnet> ". To connect to an ip type: telnet> "open [IP ADDRESS] [[-l] user] [-port].

There are other ways to remotely access a system like VNC or RDP or netcat/nc or popping your own shell through a vulnerability. Even bash has a way of opening remote shells.

Man Pages

THIS IS THE MOST IMPORTANT COMMAND YOU WILL EVER USE IN LINUX LEARN IT BREATHE IT BE IT. The **man** command stands for manual. If you are unsure on how *ssh* for example works then type "man ssh" into your terminal and it will bring up the manual for how to use that command. Normally for ssh you can just use "ssh user@ip" and it will prompt for the password, however if ssh is not on port 22 and using some unstandard port you can **search through the man page by typing "/port"** and it will highlight and occurance of port in that man page. To go to the next highlighted item you can press "n" for next, and "N" (shift + n) to go back one. There are also several different regex search things you can do in the man page but I won't go into that.

If there is no man command linux follows a 99% rule of naming convention where most commands are gonna be what you think they are, -h = help, -p = port or password, etc.

OverTheWire Bandit

Intro

Overthewire.org is a wargames website that provides interactive labs on a variety of subjects like web app security (natas), binary exploitation(like most of the other things I think), and linux (bandit).

http://overthewire.org/wargames/bandit/

bandit is a wargame organized into levels, you start at level 0 and try to beat it. Finishing a level results in the password/info for the next level, **so make sure you save all of the passwords you get so you don't have to restart every time**. You will need to re-ssh into every level, it follows this basic structure "ssh banditX@bandit.labs.overthewire.org [-p port]"

Intro Levels

Bandit0

http://overthewire.org/wargames/bandit/bandit0.html

First level is to log into their server for the labs. SSH is normally on port 22, but in the top left of the page it says it is on port 2220, so we know we have to specify a different port somehow. By using "man ssh" we can see that -p specifies the port, so we can use "ssh bandit0@bandit.labs.overthewire.org -p 2220" and the password is bandit0 for this level. If this is your first time sshing into the machine, you will be prompted to save their host as a trusted dude, just type yes and hit enter. Now that you have done level0, head over to level0 -> level1.

Bandit0 -> Bandit1

http://overthewire.org/wargames/bandit/bandit1.html

The password for the next user/level is in a file called readme located in the home directory (/hom/currentuser **or** the alias ~). So whenever you **ssh** into a machine you are *normally* dropped into that user's home directory, but just in case we can type **pwd** to print our current directory. We want to list the contents of the current directory, so we can use **ls** to *list* the contents. Cool, we see readme. From here there are a few ways we can go.

The standard way

Cat is a command to read out the contents of a file we specify, so we can do "cat readme" and it will print the content of that file. You will use this command a lot, the man page explains cat as "concatenate files and print on the standard output." Sounds spoopy and complex, just know that it reads out the content of a file.

Non standard ways but also acceptable

Crash course:

- Less readme
- More readme
- Vi readme
- Vim readme

- Nano readme
- Head readme
- Tail readme
- Like a ton more linux be whack.

Cool, looks like gibberish but that is the password for the next level so lets copy that (to copy in terminal you need to use ctrl + shift + c because ctrl +c will send an interrupt and just stop whatever youre doing) and paste it into leaf/cherrytree/libreoffice/just vim a file or whatever you want to keep track in. now we will want to type "exit" to log out of the current shell (ssh) and if we hit the up arrow we get the command we last submitted on our machine, so lets just change the banti0 to bandit1. "ssh bandit1@bandit.labs.overthewire.org -p 2220"

Bandit1 -> Bandit2

This one is weird and I don't like that they throw something this weird as the level 2. There are 3 ways that data can be handles in linux, standard in (0) standard out(1) standard error(2). Your stdin is just your keyboard/input method, your stdout is what is displayed to your monitor, and your stderr is what happens when stuff breaks, also gets printed to your monitor. For the linux gurus that are reading this I know that isn't always the case but lets just say it is for now. If you "cat –" the – also means stdin, so when you just do that it will hang because it is expecting standard input. So we have to cancel the – somehow to mean just a filename and not stdin. We can accomplish this by using a weird relative path to it. If you do ls -la you will see two directories called "." And "..", "." Means your current directory while ".." means one directory back. By typing "cat ./-" we are telling it to read the a *file* called – and not some weird symbol that is analogous with something else. Dope, we moving along.

Closing

I hope that's enough of an intro to overthewire to get you started, but if you are still confused, google is the best teacher, and you can also shoot any of the guys in slack a message, or ask a question in the #Questions chat.

Cheat Sheet

Useful commands:

- **pwd** (print working directory): prints where you are currently at in the file system. (pwd)
- **cd** (change directory): Changes the directory by taking the argument specified. (cd /var)
- **ls** (list): list the files in the current directory. (ls -la means 'list long all' and will list even hidden files (files preceded by a .) in a prettier way)
- **su** (switch user): switches to the user you specified and prompts for the password.
- **Sudo** (super user do): if you are part of the super user group and need to edit a root/restricted file this gives you the perms to do so, for example if you are in the root directory and want to list the files you would do sudo ls.
- **Ssh** (secure shell): allows you to remotely access a system.
- **Vi** (visual interface): allows you to edit files.
- **Vim** (visual interface *moBetter?)*: Improved version of vi, "A programmer's text editor", "for cool kids". Quick crash course: "i" insert mode/type text, spam "esc" to get out of whatever mode you're in, ":" command mode, ":w" write/save file, ":q" close file, ":wq" save then quit, ":!" force, ":q!" force quit without saving.

- **cat**: reads a file to the screen/stdout.