SecureSet Intro to Linux H101

Notes

1. Intro
    a. Expose you to Linux
        i. How am I going to cram years of information and experience into a few hours?
        ii. Well, I'm going to curate the information to give you the best and most effective understanding of computers so that you can enter any of SecureSet's programs with a solid foundation.
        iii. I'm not going to be able to go very deep into these subjects, I'm going to expose you to them, connect them together, it is going to be your own responsibility if you have additional questions or are attracted to or are interested in an area to start conducting your own research.
        iv. Just doing the bare minimum is not going to be a successful cybersecurity strategy. You are going to need to keep going, keep learning, do more ON YOUR OWN. No one is going to tell you to do it, push you or encourage you.
        v. Find your passion and follow it.
2. Windows vs. Linux
    a. Desktops
    b. Servers
    c. WebServers
    d. AppServers
3. Both Operating systems are very important from a Cybersecurity standpoint and from SecureSet's perspective you need to know how to operate in both Operating Systems worlds and HOW they interact with each other.
4. **Linux 101**
    a. Linux is as much a phenomenon as it is an operating system.

b. To understand why Linux has become so popular, it is helpful to know a little bit about its history. The first version of UNIX was originally developed several decades ago and was used primarily as a research operating system in universities. High-powered desktop workstations from companies like Sun proliferated in the 1980s, and they were all based on UNIX.
c. A number of companies entered the workstation field to compete against Sun: HP, IBM, Silicon Graphics, Apollo, etc. Unfortunately, each one had its own version of UNIX and this made the sale of software difficult.
d. Windows NT was Microsoft's answer to this marketplace. NT provides the same sort of features as UNIX operating systems -- security, support for multiple CPUs, large-scale memory and disk management, etc. -- but it does it in a way that is compatible with most Windows applications.
e. The entry of Microsoft into the high-end workstation arena created a strange dynamic. The proprietary operating systems owned by separate companies and the lack of a central authority in the UNIX world weaken UNIX, but many people have personal problems with Microsoft.
f. Linux stepped into this odd landscape and captured a lot of attention.
g. The Linux kernel, created by Linus Torvalds, was made available to the world for free.
h. Torvalds then invited others to add to the kernel provided that they keep their contributions free.
i. Thousands of programmers began working to enhance Linux, and the operating system grew rapidly. Because it is free and runs on PC platforms, it gained a sizeable audience among hard-core developers very quickly.
j. Linux has a dedicated following and appeals to several different kinds of people:
k. People who already know UNIX and want to run it on PC-type hardware
l. People who want to experiment with operating system principles
m. People who need or want a great deal of control over their operating system
n. People who have personal problems with Microsoft
o. In general, Linux is harder to manage than something like Windows, but offers more flexibility and configuration options.

5. Kernel, how does it work?

a. With over 13 million lines of code, the Linux kernel is one of the largest open source projects in the world, but what is a kernel and what is it used for?
b. So, what is the Kernel?
c. <u>A kernel is the lowest level of easily replaceable software that interfaces with the hardware in your computer</u>.
d. It is responsible for interfacing all of your applications that are running in "<u>user mode</u>" down to the physical hardware, and allowing processes, known as servers, to get information from each other using inter-process communication (IPC).

6. **GUI**
   a. The graphical user interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.
   b. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces which require commands to be typed on a computer keyboard.
   c. A GUI uses a combination of technologies and devices to provide a platform that users can interact with, for the tasks of gathering and producing information.
   d. A series of elements conforming a visual language have evolved to represent information stored in computers. This makes it easier for people with few computer skills to work with and use computer software. The most common combination of such elements in GUIs is the **windows**, **icons**, **menus**, **pointer** (**WIMP**) paradigm, especially in personal computers.
7. **CLI**
   a. A command-line interface or command language interpreter (CLI), also known as command-line user interface. A program which handles the interface is called a command language interpreter or shell (computing).
   b. Command-line interfaces to computer operating systems are less widely used by casual computer users, who favor graphical user interfaces or menu-driven interaction.
   c. Compared with a graphical user interface, a command line requires fewer system resources to implement. Since options to commands are given in a few characters in each command line, an experienced user finds the options easier to access. Automation of repetitive tasks is simplified - most operating systems using a command line interface

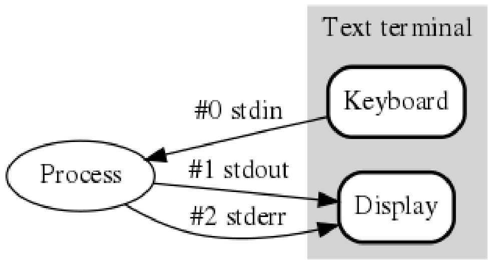
support some mechanism for storing frequently used sequences in a disk file, for re-use; this may extend to a scripting language that can take parameters and variable options. A command-line history can be kept, allowing review or repetition of commands.

d. A command-line system may require paper or online manuals for the user's reference, although often a "help" option provides a concise review of the options of a command.
e. The command-line environment may not provide the graphical enhancements such as different fonts or extended edit windows found in a GUI.
f. It may be difficult for a new user to become familiar with all the commands and options available, compared with the drop-down menus of a graphical user interface, without repeated reference to manuals.
g. The interface is usually implemented with a command line shell, which is a program that accepts commands as text input and converts commands into appropriate operating system functions.
h. Command-line interfaces are often preferred by more advanced computer users, as **they often provide a more concise and powerful means to control a program or operating system**.
i. Programs with command-line interfaces are generally easier to automate via scripting.
j. GUI interfaces have to output interpreted commands into the CLI for computers to work.
k. If an advanced user understands the commands in the CLI well enough, there is more power and flexibility directly from the Command Line that is ever implemented in a GUI. This is why from SecureSet's perspective true mastery over computers comes from becoming familiar and effective from the command line.

8. Time

a. For Unix like operating systems, the epoch is 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970.date
b. is a system for describing a point in time, defined as an approximation of the number of seconds that have elapsed since epoch time.
c. However, Unix time is not a true representation of UTC, as leap seconds are not accounted for. A leap second in UTC shares the same Unix time as the second which came before it.
d.

9. **Command Line Basics**
    a. The bash shell is one of several shells available for Linux. It is also called the Bourne-again shell, after Stephen Bourne, the creator of an earlier shell (/bin/sh). Bash is substantially compatible with sh, but it provides many improvements in both function and programming capability.
10. To OPEN a TERMINAL
    a. If you're running Unity: open the dash, type terminal, hit Return.
    b. If you're on the old style menus, Applications → Accessories → Terminal.
    c. Control + Alt + T.
11. A simple command consists of a sequence of words separated by spaces or tabs. The first word is taken to be the name of a command, and the remaining words are passed as arguments to the command.
12. Before we delve deeper into bash, recall that a shell is a program that accepts and executes commands. It also supports programming constructs, allowing complex commands to be built from smaller parts. These complex commands, or scripts, can be saved as files to become new commands in their own right. Indeed, many commands on a typical Linux system are scripts.
13. **Shells** have some built-in commands, such as cd, break, and exec. Other commands are external.
14. **Shells** also use three standard I/O streams:
    a. the standard input stream, which provides input to commands is called **stdin**
    b. the standard output stream, which displays output from commands is called **stdout**
    c. the standard error stream, which displays error output from commands is called **stderr**
    d. Input streams provide input to programs, usually from terminal keystrokes. Output streams print text characters, usually to the terminal. The terminal was originally an ASCII typewriter or display terminal, but it is now more often a window on a graphical desktop.
        i. In order to understand about Redirecting operators in Linux we should know how we communicate with a computer.
            1. When we are communicating with a computer there should be a way to do it and this is achieved by STDIN (0), STDOUT (1), STDERR (2) file descriptors.



2.
3. These file descriptors are assigned with a number as shown below:
4. STDIN –> 0
5. STDOUT –> 1
6. STDERR –> 2
7. STDIN: stands for STandarD INput.
    a. By using this we can give an input to the computer to do some task. Whatever device we used to give input to a computer will come under STDIN.
    b. A STDIN can be A keyboard A mouse A scanner A floppy A CD/DVD ROM A touch screen A barcode reader or a card reader.
8. STDOUT: The abbreviation is STandarD OUTput.
    a. By using this we can see the output from a computer for the it just processed or executed. Whatever device we used to get the output from a computer will come under this STDOUT.
    b. A STDOUT can be A monitor A speaker Or a printer
9. STDERR: This is abbreviated as STandarD ERRor.
10. By using this, A computer can communicate with user to give him warning/error etc. that something went wrong when executing a task.
11. A STDERR can be A monitor A printer A log file A LED indicator Or a speaker.
12. Why we require redirecting operators? We require redirecting operators in some situations where our standard communication with computer will not meet our requirement.
13. For example, sometimes we want to store an error which is popping on a screen to a file for future reference.

14. At that time, we can use one of these redirecting operators to change the default way of communication between a user and a server.

e.

15. **Shells** 2.0

a. Every user has a unique username. When they logon to the system, they are placed in a HOME directory, which is a portion of the disk space reserved just for them.

b. When you log onto a UNIX system, your main interface to the system is called the UNIX SHELL.

c. This is the program that presents you with the dollar sign ($) prompt.

d. This prompt means that the shell is ready to accept your typed commands.

e. We know in Ubuntu Server or Desktop you will using one of the most standard UNIX shells called the Bourne Shell.

f. UNIX commands are strings of characters typed in at the keyboard. To run a command, you just type it in at the keyboard and press the ENTER key.

g. **UNIX extends the power of commands by using special flags, arguments or switches**.

h. These operators are one of the most powerful features of UNIX commands.

i. Switches are usually preceded with a dash (-) and precede any filenames or other arguments on the command line.

j. Unlike the DOS (or Windows) command line, UNIX systems are case sensitive (upper and lower-case characters are considered different).

k. Nearly all command names and most of their command line switches will be in lowercase.

l. In this class all of the UNIX commands should be typed in lowercase characters unless explicitly instructed otherwise.

m. UNIX systems have a hierarchical directory structure. This means that the hard disk area is divided into directories, much like a book is sub-divided into chapters and paragraphs. The directories form a tree-like structure, which simplifies the organization of the files on the system.

16. **LINUX FILE SYSTEM**

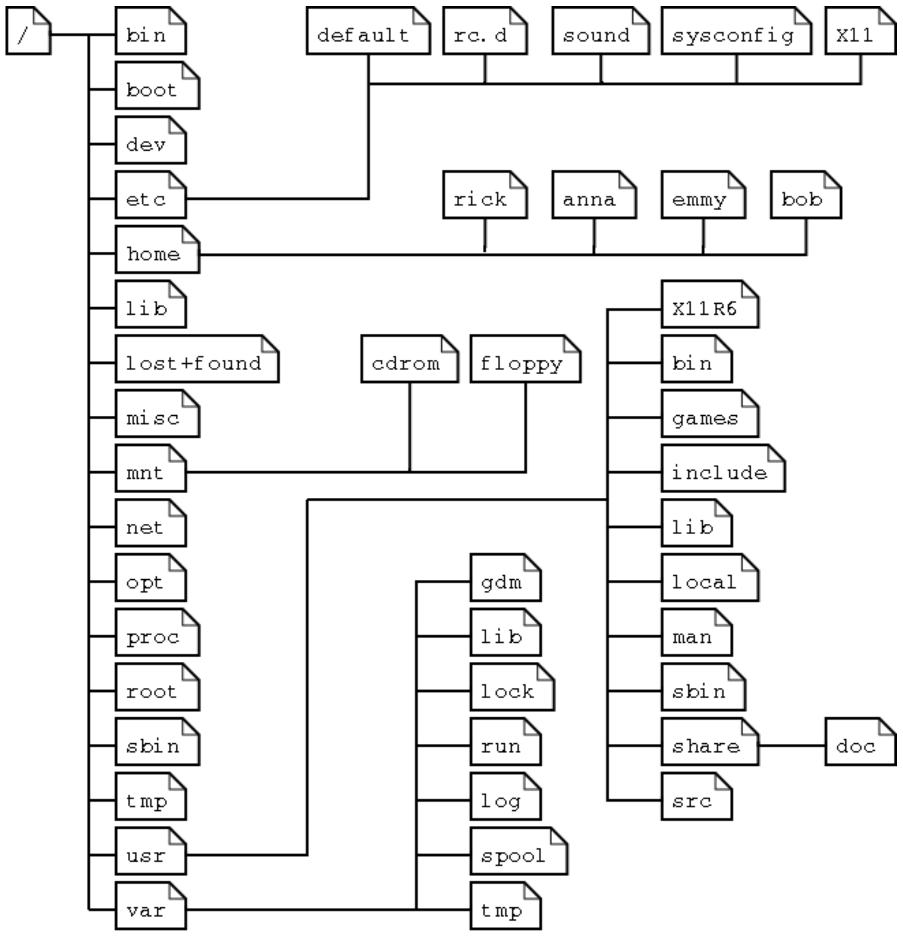
a. Linux Saying

i. "**On a UNIX system, everything is a file; if something is not a file, it is a process**."

ii. This statement is true because there are special files that are more than just files but to keep things simple, saying that everything is a file is an acceptable generalization. A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system.

iii. In order to manage all those files in an orderly fashion, man likes to think of them in an ordered tree-like structure on the hard disk, as we know from MS-DOS (Disk Operating System) for instance. The large branches contain more branches, and the branches at the end contain the tree's leaves or normal files.

b. For now, we will use an image of the tree, to wrap our heads around this.



c.

d. Depending on the system admin, the operating system and the mission of the UNIX machine, the structure may vary, and directories may be left out or added at will. The names are not even required; they are only a convention.

e. MAIN DIRECTORIES

    i. /bin is a place for most commonly used terminal commands, like ls, mount, rm, etc.
    ii. /boot contains files needed to start up the system, including the Linux kernel, a RAM disk image and bootloader configuration files.
    iii. /dev contains all device files, which are not regular files but instead refer to various hardware devices on the system, including hard drives.
    iv. /etc contains system-global configuration files, which affect the system's behavior for all users. – How its pronounced.
    v. /home home sweet home, this is the place for users' home directories.
    vi. /lib contains very important dynamic libraries and kernel modules
    vii. /media is intended as a mount point for external devices, such as hard drives or removable media (floppies, CDs, DVDs).
    viii. /mnt is also a place for mount points, but dedicated specifically to "temporarily mounted" devices, such as network filesystems.
    ix. /opt can be used to store additional software for your system, which is not handled by the package manager.
    x. /proc is a virtual filesystem that provides a mechanism for kernel to send information to processes.
    xi. /root is the superuser's home directory, not in /home/ to allow for booting the system even if /home/ is not available.
    xii. /run is a tmpfs (temporary file system) available early in the boot process where ephemeral run-time data is stored. Files under this directory are removed or truncated at the beginning of the boot process.
        1. (It deprecates various legacy locations such as /var/run, /var/lock, /lib/init/rw in otherwise non-ephemeral directory trees as well as /dev/.* and /dev/shm which are not device files.)
    xiii. /sbin contains important administrative commands that should generally only be employed by the superuser.

xiv. /srv can contain data directories of services such as HTTP (/srv/www/) or FTP.
xv. /sys is a virtual filesystem that can be accessed to set or obtain information about the kernel's view of the system.
xvi. /tmp is a place for temporary files used by applications.
xvii. /usr contains the majority of user utilities and applications, and partly replicates the root directory structure, containing for instance, among others, /usr/bin/ and /usr/lib.
xviii. /var is dedicated to variable data, such as logs, databases, websites, and temporary spool (e-mail etc.) files that persist from one boot to the next. A notable directory it contains is /var/log where system log files are kept.

17. ABSOLUTE PATH

a. An absolute path name, pointing to what is normally an executable file on an Ubuntu system:
    i. /usr/bin/test
b. An absolute path name, but pointing to a directory instead of a regular file:
    i. /usr/bin/
c. A relative path name, which will point to /usr/bin/test only if the current directory is /usr/:
    i. bin/test
d. A relative path name, which will point to /usr/bin/test if the current directory is any directory in /usr/, for instance /usr/share/:
    i. ../bin/test
e. A path name using the special shortcut ~, which refers to the current user's home directory:
    i. ~/Desktop/
f. Path names can contain almost any character, but some characters, such as space, must be escaped in most software, usually by enclosing the name in quotation marks:
    i. "~/Examples/Experience ubuntu.ogg"
g. or by employing the escape character \:
    i. ~/Examples/Experience\ ubuntu.ogg

   h. ./config means you're calling something in the current working directory. In this case config is an executable. You have to specify the path for executables if they're outside your $PATH variable and that's why config isn't enough.
   i. ../config would be used if the config executable were in the parent of the current working directory.

18. CLI START
   a. When you first log into a UNIX system, you are placed in your own personal directory space called your HOME directory. On most UNIX systems, the user HOME directories are located under the /home directory.
   b. Types of prompts
      i. user@computername~$
      ii. ajay@linux1:~$ Home Directory
      iii. ajay@linux1:/$ Root of File System
      iv. ajay@linux1:/home   Home folder, but not my user home folder.
      v. root@linux1:/#   If you are logged in as root or a superuser instead of $ you will get a #
      vi. The root user has considerable power, so use it with caution. When you have root privileges, most prompts include a trailing pound sign (#). Ordinary user privileges are usually delineated by a different character, commonly a dollar sign ($). Your actual prompt may look different than the examples in this tutorial. Your prompt may include your user name, hostname, current directory, date, or time that the prompt was printed, and so on. We will discuss root and superusers shortly.
      vii. The shell's main function is to interpret your commands so you can interact with your Linux system.
      viii. If a line contains a # character, then all remaining characters on the line are ignored. So, a # character may indicate a comment as well as a root prompt. Which it is should be evident from the context.
      ix. You need to quote strings, using either double quotes (") or single quotes ('). Bash uses white space, such as blanks, tabs, and new line characters, to separate your input line into tokens, which are then passed to your command.

Quoting strings preserves additional white space and makes the whole string a single token.

x.

19. CLI NAVIGATION
    a. echo
        i. The echo program displays text. It's a handy way to create customized output in your terminal.
        ii. echo is a fundamental command found in most operating systems. It is frequently used in scripts, batch files, and as part of individual commands; anywhere you may need to output text.
        iii. echo [SHORT-OPTION]... [STRING]...
    b. pwd (print working directory) – WHERE THE HECK AM I? =)
        i. Prints the current working directory on your screen. This is the directory where you are currently located. When you manipulate files and sub-directories, this is where they will (by default) be.
        ii. Here is an example of using pwd:
            1. pwd
            2. /homeb/bpowell
    c. cd (change directory)
        i. This command is used to change the current working directory.
        ii. Here are some examples of using cd:
            1. cd datafiles   cd ..   cd /   cd $HOME
        iii. The last example shows the use of a UNIX environment variable. The $HOME variable always contains the location of your HOME directory. (We will discuss BASH variables later in more detail)
        iv. When an environment variable is used in a command, the contents of the variable and substituted for its name, so the above example would end up doing the same thing as if you typed in:
    d. cd /homeb/bpowell
    e. env
        i. You can print all of your environment variables by running the "env" command.
    f. mkdir (make directory)

        i. This command makes a sub-directory under the current working directory.
        ii. mkdir junk

g. rmdir (remove directory)
    i. This command removes (deletes) a sub-directory under the current working directory. The directory to be removed must be empty of all files and sub-directories.
    ii. rmdir junk

h. touch
    i. touch changes file timestamps. It is also an easy way to create empty files.

i. date
    i. date will display the time and date as well as time zone information.

j. Manipulate files
    i. ls (list files)
        1. This command is similar to DIR in DOS; it displays a list of all the files in the directory. New users don't have any files in their home directory.
        2. Here is an example of using this command;
        3. ls
        4. You will note that the shell prompt reappears, and there is no file listing of the directory contents. This does not mean that there are no files stored there. Just like DOS, UNIX systems support hidden files.
        5. Here is an example of using the command with switches:
            a. ls -la
        6. The switch l stands for long listing, **and the a switch is for all files, including directories and hidden files.**
        7. UNIX responds with the following listing of the directory
            a. ls -la
            b. total 6 -rw-rw-r-- 1 jmsmith staff 526 Apr 15 11:03 myletter
    ii. less (page through a text file)
        1. This command allows you to view a text file without fear of accidently modifying it, as you would with a text editor.
        2. Example:

        a. less /etc/hosts

iii. cat (concatenate files)

1. This command is used to combine files or to print files to the screen. By default, the cat command sends its output to your screen (in UNIX we call this standard-output or stdout for short).
2. The following command can be used to view the file, ".profile" on stdout:
3. cat .profile
4. The format of this command specifies that cat will use the file, ".profile" as its input, and send the output to your screen.
5. Note: if the file is very large or is not a plain text file, cat will try to print it to the screen anyway, sometimes with undesirable results. The less command is better suited for viewing files than the cat command.

iv. cp (copy files)

1. This command stands for copy, and is used for copying one file to another.
2. Example:
    a. cp .profile temp2
3. This copies the file .profile to another called temp2. If temp2 had already existed, its previous contents would've been erased.
4. Files can also be copied to another directory. The command
5. cp * /usr/tmp
    a. would copy all the files in the current directory to the directory /usr/tmp.

v. mv (move files)

1. The mv command is used for moving or renaming files.
2. Example:
    a. mv temp temp2
3. This renames the file temp to temp2.
4. As an example (do not type this), the command:
5. mv temp2 /tmp

6. would move the file temp2 into the directory /tmp (it would no longer appear in your home directory).

vi. rm (remove files)

1. The rm utility is used for erasing files and directories.
    a. Example:
        i. rm temp2
2. This removes the file. Once a file is removed, it cannot be restored. To cover situations where mistakes might occur, a switch -i appended to this command will request a Yes or No response before deleting the file.
3. Example:
    a. rm -i temp1
4. NOTE that switches are written before the filenames. Answer Y to the prompt so that temp1 is removed.
5. rm -rf /
    a. Why you should do it once. On a VM, you don't care about.

vii. head

1. This is used to view the first few lines of a file. It accepts a switch specifying the number of lines to view. The command
2. head -2 temp
3. would list the first 2 lines of the file temp on your screen.

viii. tail

1. This is used to view the last few lines of a file. It accepts a switch specifying the number of lines to view. The command
2. tail -2 temp
3. tail -f logfile.log
4. This forces the tail to follow the log, so it shows items being written to that file in real-time.

ix. diff

1. diff analyzes two files and prints the lines that are different. Essentially, it outputs a set of instructions for how to change one file to make it identical to the second file.
2. diff file1.txt file2.txt

x. clear
    1. Wipes the screen squeaky clean and starts at the top.

xi. who (display a list of users)
    1. This command will display users currently on the system.

xii. help (obtain help on command usage)
    1. Most commands provide a short summary of their switches and arguments when you give them the "--help" switch.
    2. Example:
        a. To obtain help for the command grep type:
            i. grep –help