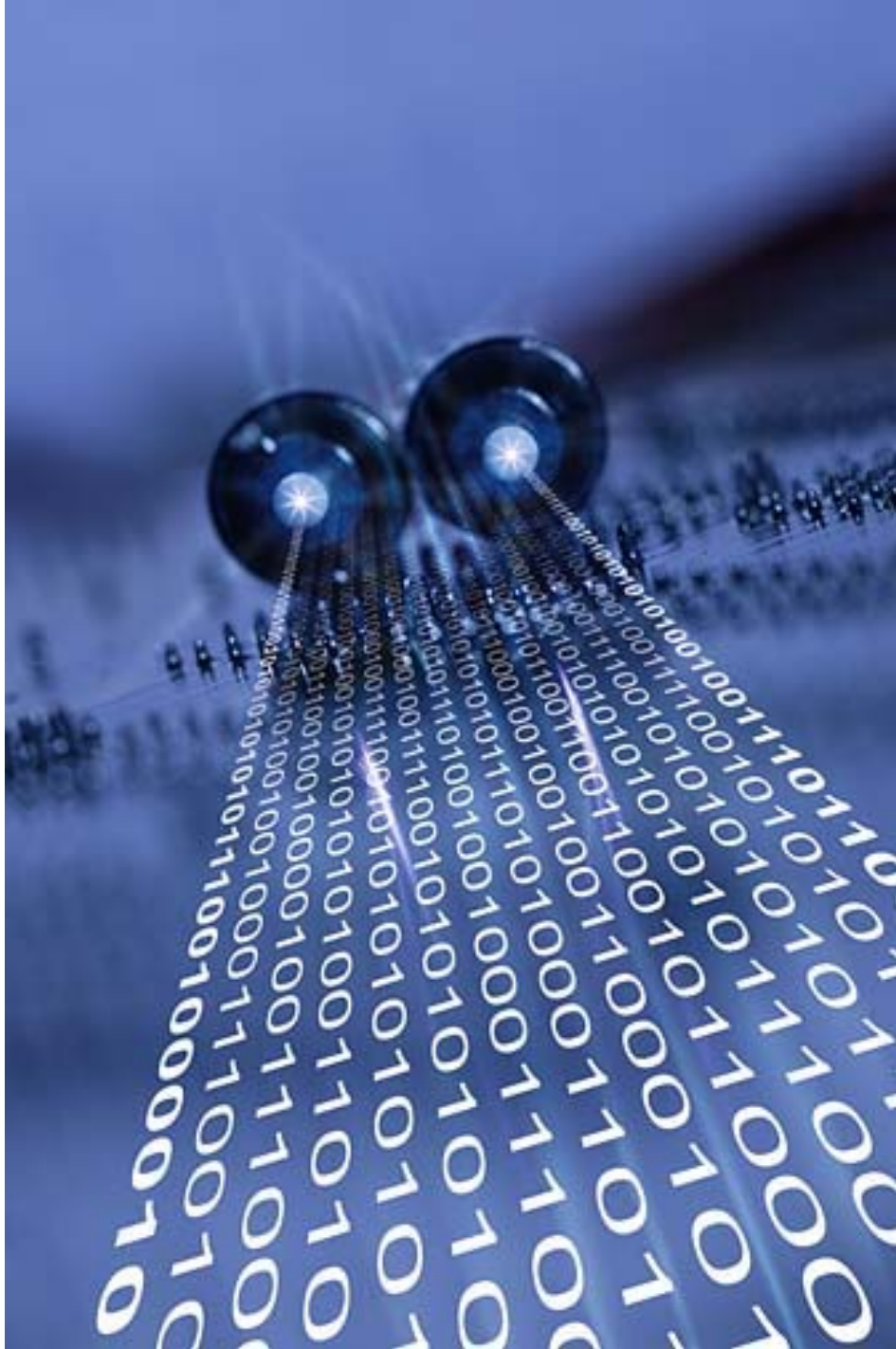


Network Security Laboratory Book



Student Name: _____

G#: _____

Table of Contents

Ethics Statement	3
Unix Background Information	4
Unix/Linux Lab	16
Nmap Lab	20
Nessus Lab	23
iptables Lab	25

Ethics Statement

Network Security Student Certification and Agreement

I, _____, hereby certify that I read the following:

University Policy Number 1301: Responsible Use of Computing

<http://www.gmu.edu/facstaff/policy/newpolicy/1301gen.html>

I understand that GMU takes its ethical obligations very seriously and violations will not be tolerated. I fully understand that GMU and its students must conduct the Program's activities in accordance with the highest possible ethical and legal standards. I know that I am responsible for ensuring that my personal conduct is above reproach. As a condition of studying in the ISA Program at GMU, I agree that violations of the standards described in the Code of Conduct shall be made known immediately to my appropriate faculty member(s) and that violations will result in dismissal from the Program and failure to receive the degree. I understand that this is a zero tolerance policy and that no second chances are given.

I agree to take all reasonable precautions to assure that sensitive University or faculty information, or information that has been entrusted to my fellow students or me by third parties (such as the students' employers), will not be disclosed to unauthorized persons. I understand that I am not authorized to use sensitive information for my own purposes, nor am I at liberty to provide this information to third parties without the express written consent of the faculty or the person who is the designated information owner or custodian.

I also understand specifically that GMU provides computer systems and networks for my use in academic studies and that I am not permitted to use those computer systems and networks for personal business or for any activities not related to my academic studies. I understand that GMU audits and monitors the use of those computer systems and networks and that I have no right to privacy or expectation of privacy when I use computer systems and networks provided to me by GMU.

Signature

Printed Name

G Number

Date

Unix Background Information

Purpose:

Since many of the labs will require knowledge of Unix/Linux, we have included some useful background information. Feel free to explore the references listed as well utilize google.com to expand on any topic.

PART 1 – Unix-Linux History

In order to understand the popularity of Linux, we need to travel back in time, about 55 years ago...

Imagine computers as big as houses, even stadiums. While the sizes of those computers posed substantial problems, there was one thing that made this even worse: every computer had a different operating system. Software was always customized to serve a specific purpose, and software for one given system didn't run on another system. Being able to work with one system didn't automatically mean that you could work with another. It was difficult, both for the users and the system administrators.

Technologically the world was not quite that advanced, so they had to live with the size for another decade. In 1969, a team of developers in the Bell Labs laboratories started working on a solution for the software problem, to address these compatibility issues. They developed a new operating system, which was

1. Simple and elegant.
2. Written in the C programming language instead of in assembly code.
3. Able to recycle code.

The Bell Labs developers named their project "UNIX."

The code recycling features were very important. Until then, all commercially available computer systems were written in a code specifically developed for one system. UNIX on the other hand needed only a small piece of that special code, which is now commonly named the kernel. This kernel is the only piece of code that needs to be adapted for every specific system and forms the base of the UNIX system. The operating system and all other functions were built around this kernel and written in a higher programming language, C. This language was especially developed for creating the UNIX system. Using this new technique, it was much easier to develop an operating system that could run on many different types of hardware.

The software vendors were quick to adapt, since they could sell ten times more software almost effortlessly. Weird new situations came in existence: imagine for instance computers from different vendors communicating in the same network, or users working on different systems without the need for extra education to use another computer. UNIX did a great deal to help users become compatible with different systems.

Throughout the next couple of decades the development of UNIX continued. More things became possible to do and more hardware and software vendors added support for UNIX to their products.

UNIX was initially found only in very large environments with mainframes and minicomputers (note that a PC is a “micro” computer). You had to work at a university, for the government or for large financial corporations in order to get your hands on a UNIX system.

But smaller computers were being developed, and by the end of the 80s, many people had home computers. By that time, there were several versions of UNIX available for the PC architecture, but none of them were truly free and more important: they were all terribly slow, so most people ran MS DOS or Windows 3.1 on their home PCs.

By the beginning of the 90s home PCs were finally powerful enough to run a full blown UNIX.

While there was an academic UNIX-lookalike called Minix available at the time, its creator, Andrew S. Tanenbaum did not allow modifications that would make it more generally usable. He wanted his system to stay “clean”, since he created it in order to teach computer science with it.

Linus Torvalds, a young man studying computer science at the University of Helsinki, used the Minix, and when he felt too constrained by its limitation, he started to code his own UNIX-lookalike operating system.

From the start, it was Linus’ goal to have a free system that was completely compliant with the original UNIX. That is why he asked for POSIX standards, POSIX still being the standard for UNIX.

In those days plug-and-play wasn’t invented yet, but so many people were interested in having a UNIX system of their own, that this was only a small obstacle. New drivers became available for all kinds of new hardware, at a continuously rising speed. Almost as soon as a new piece of hardware became available, someone bought it and submitted it to the Linux test, as the system was gradually being called, releasing more free code for an ever wider range of hardware. These coders didn’t stop at their PCs; every piece of hardware they could find was useful for Linux.

Two years after Linus’ post, there were 12000 Linux users. The project, popular with hobbyists, grew steadily, all the while staying within the bounds of the POSIX standard. All the features of UNIX were added over the next couple of years, resulting in the mature operating system Linux has become today. Linux is a full UNIX clone, fit for use on workstations as well as on middle-range and high-end servers. Today, a lot of the important players in the hardware and software market each have their team of Linux developers; at your local dealers you can even buy pre-installed Linux systems with official support—though there is still some hardware and software not supported.

PART 2 – Linux Pros

A lot of the advantages of Linux are a consequence of Linux’ origins, deeply rooted in UNIX, except for the first advantage, of course:

- Linux is free:
 - If you want to spend absolutely nothing, you don't even have to pay the price of a CD. Linux can be downloaded in its entirety from the Internet completely for free. No registration fees, no costs per user, free updates, and freely available source code in case you want to change the behavior of your system.
 - Most of all, Linux is free as in free speech: The license commonly used is the GNU Public License (GPL). The license says that anybody who may want to do so, has the right to change Linux and eventually to redistribute a changed version, on the one condition that the code is still available after redistribution.
- Linux is portable to any hardware platform:
 - A vendor who wants to sell a new type of computer and who doesn't know what kind of OS his new machine will run (say the CPU in your car or washing machine), can take a Linux kernel and make it work on his hardware, because documentation related to this activity is freely available.
- Linux was made to keep on running:
 - As with UNIX, a Linux system expects to run without rebooting all the time. That is why a lot of tasks are being executed at night or scheduled automatically for other calm moments, resulting in higher availability during busier periods and a more balanced use of the hardware. This property allows for Linux to be applicable also in environments where people don't have the time or the possibility to control their systems night and day.
- Linux is secure and versatile:
 - The security model used in Linux is based on the UNIX idea of security, which is known to be robust and of proven quality. But Linux is not only fit for use as a fort against enemy attacks from the Internet: it will adapt equally to other situations, utilizing the same high standards for security.
- Linux is scalable:
 - From a Palmtop with 2 MB of memory to a petabyte storage cluster with hundreds of nodes: add or remove the appropriate packages and Linux fits all. You don't need a supercomputer anymore, because you can use Linux to do big things using the building blocks provided with the system. If you want to do little things, such as making an operating system for an embedded processor or just recycling your old 486, Linux will do that as well.
- The Linux OS and most Linux applications have very short debug-times:
 - Because Linux has been developed and tested by thousands of people, both errors and people to fix them are usually found rather quickly. It sometimes happens that there are only a couple of hours between discovery and fixing of a bug.
- Open Source products are trustworthy:
 - How can something that is free also be reliable? Linux users have the choice whether to use Linux or not, which gives them an enormous advantage compared to users of proprietary software, who don't have that kind of freedom. After long periods of testing, most Linux users come to the conclusion that Linux is not only as good, but in many cases better and faster than the traditional solutions. If Linux were not trustworthy, it would have been long gone, never knowing the popularity it has now, with millions of users. Now users can influence their systems and share their remarks with the community, so the system gets better and better every

day. It is a project that is never finished, that is true, but in an ever changing environment, Linux is also a project that continues to strive for perfection.

- The discussion about being trustworthy is often brought up by vendors of closed-source software. But can you trust software you buy? How many applications are on the market that “phone home”, sending data about you back to the vendor? Open Source software makes the sources available to the many eyes that want to see them. And even though regular users will not check every application’s source, for each critical package there are people who do so.
- In this area where trustworthiness is critical, professionals **insist** on open sources. Cryptographical algorithms have to be disclosed for peer-review and analysis before they will be considered secure.

PART 3 – Linux Cons

- There are far too many different distributions:
 - At first glance, the amount of Linux distributions can be frightening, or ridiculous, depending on your point of view. But it also means that everyone will find what he or she needs. You don’t need to be an expert to find a suitable release.
 - When asked, generally every Linux user will say that the best distribution is the specific version he is using. So which one should you choose? Don’t worry too much about that: all releases contain more or less the same set of basic packages. On top of the basics, special third party software is added (e.g. configuration helpers). However, the differences are likely to be very superficial.
- Linux is not very user friendly and confusing for beginners:
 - It must be said that Linux, at least the core system, is less user-friendly than MS Windows and certainly more difficult than MacOS, but... In light of its popularity, considerable effort has been made to make Linux even easier to use, especially for new users. More information is being released daily, to help fill the gap for documentation available to users at all levels.

PART 4 – Linux Filesystem

Linux maintains a hierarchical file system, similar to what you are used to with Windows or Mac OS. The file system is called hierarchical because directories (or folders) can be created inside other directories to create a hierarchical tree-like structure. Unlike Windows and Mac OS, however, this tree has only one root, called the *root directory*, represented by / (a forward slash).

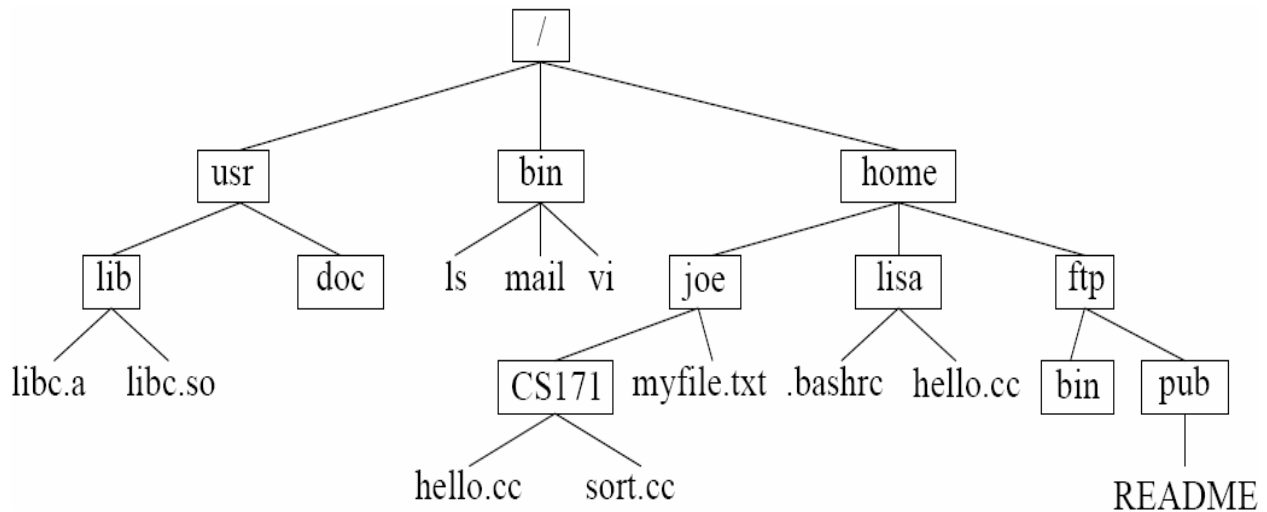


Figure 1: Example of Linux hierarchical file system.

A simple example of a hierarchical file system is shown in Figure 1. Each boxed name represents a directory, while the unboxed names are files.

Linux file names are case sensitive and may contain almost any character. File names may or may not be followed by an extension like .txt or .cc or html. In fact, the period in a file name is not given any special significance by a shell, and extensions are rarely required for a file to be opened by a particular application. However, it is usually a good idea to include an extension for a file so it is easier for you to figure out what kind of file it is. By convention, executable programs in Linux usually have no extension.

Any directory that is not the root is usually called a subdirectory. For example, in Figure 1, usr is a subdirectory of / and doc is a subdirectory of usr. The directory usr is also called the parent directory of doc and / is the parent directory of usr. The root directory is the only directory without a parent; by convention, the root directory is its own parent.

In a Linux filesystem, the bin subdirectory contains programs that correspond to core Linux commands. The usr subdirectory contains many other parts of the basic Linux system. The home subdirectory contains the home directories of all the users with accounts on the system. If your username were joe, you could store your files in the joe subdirectory of home.

The pathname of a file contains a sequence of directories to follow to reach the file. For example, the pathname of the joe subdirectory is /home/joe. The pathname of the file myfile.txt in the joe subdirectory is /home/joe/myfile.txt. The pathnames above are called absolute pathnames because they contain all the information needed to find a file. On the other hand, a relative pathname gives the information necessary to find a file from a particular point in the tree. For example, from the directory /home, the relative pathname of myfile.txt is just joe/myfile.txt. Notice that you can tell the difference between an absolute and a relative pathname by looking for the leading forward slash.

PART 5 – Linux Disks and Partitions

Disks

Linux treats its devices as files. The special directory where these "files" are maintained is `/dev`.

- Floppy (a:) /dev/fd0
- Floppy (b:) /dev/fd1
- 1st Hard disk (master, IDE-0) /dev/hda
- Hard disk (slave, IDE-0) /dev/hdb
- Hard disk (master, IDE-1) /dev/hdc, etc.
- 1st SCSI hard disk /dev/sda
- 2nd SCSI hard disk /dev/sdb, etc.

Partitions

- 1st Hard disk (master, IDE-0) /dev/hda
 - 1st Primary partition /dev/hda1
 - 2nd Primary partition /dev/hda2, etc.
 - 1st Logical drive (on ext'd part) /dev/hda5
 - 2nd Logical drive /dev/hda6, etc.
- 2nd Hard disk (slave, IDE-0) /dev/hdb
 - 1st Primary partition /dev/hdb1, etc
- CDROM or 3rd disk (master, IDE-1) /dev/hdc
- CDROM (SCSI) /dev/scd0
- 1st SCSI disk /dev/sda
 - 1st Primary partition /dev/sda1, etc.

The pattern described above is fairly easy to follow. If you are using a standard IDE disk, it will be referred to as "hdx" where the "x" is replaced with an "a" if the disk is connected to the primary IDE controller as master and a "b" if the disk is connected to the primary IDE controller as a slave device. In the same way, the IDE disks connected to the secondary IDE controller as master and slave will be referred to as "hdc" and "hdd" respectively

Note: Before a filesystems on devices can be used, they must be mounted. In order to mount them, you must know what they are called. So for example, if you use a parallel ZIP drive or USB disk (thumb drive, memory stick, etc.), it will be accessed as `/dev/sda` (assuming no other SCSI devices) or `/dev/sdb`.

PART 6 – Linux File Permissions

Every file or folder in Linux has access permissions. There are three types of permissions (what allowed to do with a file):

- read access
- write access
- execute access

Permissions are defined for three types of users:

- the owner of the file
- the group that the owner belongs to
- other users

Thus, Linux file permissions are nine bits of information (3 types x 3 type of users), each of them may have just one of two values: allowed or denied.

Simply put, for each file it can be specified who can read or write from/to the file. For programs or scripts it also can be set if they are allowed to be executed.

It is used in Linux long directory listings. It consists of 10 characters. The first character shows the file type. Next 9 characters are permissions, consisting of three groups: owner, group, others. Each group consists of three symbols: **rwX** (in this order), if some permission is denied, then a dash "-" is used instead. Example:

```
-rwxr--r--
0123456789
```

- Symbol in the position 0 ("-") is the type of the file. It is either:
 - d** = directory
 - = regular file
 - l** = symbolic link
 - s** = Unix domain socket
 - p** = named pipe
 - c** = character device file
 - b** = block device file
- Symbols in positions 1 to 3 ("rwx") are permissions for the owner of the file.
- Symbols in positions 4 to 6 ("r--") are permissions for the group.
- Symbols in positions 7 to 9 ("r--") are permissions for others.

r Read access is allowed

w Write access is allowed

x Execute access is allowed

- Replaces "r", "w" or "x" if according access type is denied

Examples:

File,
-rwxr-xr-x owner has read, write, execute permissions,
 group: only read and execute permissions,
 others: only read and execute permissions.

Directory,
dr-x----- owner has read and execute access,
 group and others have no access

If a numeric representation is used (like in `chmod` command, for example), then it is in the octal format (with the base of 8), and digits involved are 0 to 7. Octal format is used for the simplicity of understanding: every octal digit combines read, write and execute permissions together. Respective access rights for owner, group and others (in this order) are the last three digits of the numeric file permissions representation. Example: "0644". Here the second digit ("6" in the example) stands for rights of the owner, the third digit ("4" in the example) stands for rights of the group, the fourth digit ("4" in the example) stands for rights of others.

This table shows what numeric values mean:

Octal digit	Text equivalent	Binary value	Meaning
0	---	000	All types of access are denied
1	--x	001	Execute access is allowed only
2	-w-	010	Write access is allowed only
3	-wx	011	Write and execute access are allowed
4	r--	100	Read access is allowed only
5	r-x	101	Read and execute access are allowed
6	rw-	110	Read and write access are allowed
7	rwX	111	Everything is allowed

We see that "1" stands for execute only, "2" stands for write only, "4" stands for read only. To combine the permissions you can simply add 1, 2 and 4 to get a needed combination. For instance, to get read and write permissions, you add 4 (read) and 2 (write), thus getting 6 (read and write). To get read and execute permissions, you add 4 (read) and 1 (execute), thus getting 5 (read and execute).

Examples:

- owner: read and write permissions,
- 644** group: only read permissions,
- others: only read permissions.
- owner: read, write and execute permissions,
- 755** group: read and execute permissions,
- others: read and execute permissions.

Access permissions for files and folders mean different things from the user standpoint. The table below shows the difference.

Access type	File	Folder
Read	If the file contents can be read	If the directory listing can be obtained

Write	If user or process can write to the file (change its contents)	If user or process can change directory contents somehow: create new or delete existing files in the directory or rename files.
Execute	If the file can be executed	If user or process can access the directory, that is, go to it (make it to be the current working directory)

PART 7 – Useful Commands

Directories

Linux "folders" are called *directories*. The top-level, root directory is called */*. Your home directory is */home/username*. From anywhere you can get back there by typing simply

cd

The short-hand name for the directory you happen to be in at any time is called "." and the directory in which the current directory resides is called "..". Typing "**cd ..**" will move you to the next higher level directory. Several useful commands for directories are listed below.

Command	Function	Examples
cd	Change directory	cd , cd .. , cd /home/catyp
pwd	Print working directory	pwd
mkdir	Make a new subdirectory	mkdir <i>newdirectory</i>
rmdir	Remove a directory	rmdir <i>emptydirectory</i>
ls	List files in a directory	ls , ls -l

Files:

Files reside in directories. Use the **ls** command (or **ls -l** for more information) to see all the files in a directory. Useful commands for manipulating files include:

Command	Function	Examples
mv	Rename (move) a file	mv <i>oldname newname</i>
cp	Copy a file	cp <i>oldname newname</i> cp <i>oldname dirname/</i>
rm	Delete (remove) a file	rm <i>filename</i> rm <i>file1 file2 file3</i> rm -r <i>dirname</i>
cat	Output the contents of a file to the screen	cat <i>filename</i>
file	Identify the type of file	file <i>filename</i>
head	Display the first few lines of a text file.	head <i>filename</i>
tail	Display the last few lines of a text file.	tail <i>filename</i>
chmod	Change access permissions on file(s)	chmod <i>mode filename</i>
ln	Creates symbolic link	ln -s <i>targetfile linkname</i>

Other Commands:

Command	Function	Examples
passwd	Change your password.	Passwd
ps	List running processes of the current terminal. If you want to see all processes of the current user, use “-a”, if you're still not satisfied with the output, try “-e”.	ps -aux
kill	Stop a process by passing its process-id (shown by ps as PID).	kill -TERM process-id kill -9 process-id
tar	Create / expand / query archives.	tar cvf arch.tar somedir/ tar xfv arch.tar
mount	Make a device visible in the filesystem. For users, this is typically used to access CD-Roms or floppies.	mount /mnt/floppy
umount	Un-mount a device from a given point in the filesystem.	umount floppy
df	Reports on used disk space on the partition containing file.	df file
find	Find files in the file system hierarchy	find path expression
grep	Print lines in file containing the search pattern.	grep PATTERN file
ifconfig	Configures a network interface	ifconfig
exit or logout	Leave this session	Exit

PART 8 – Other Useful Info

Cutting and Pasting:

In the windowing system and on the text interface, it is possible to cut and paste text. First, select the text with the mouse. Once you select the text, it will automatically be copied to the clipboard. You can now paste the text elsewhere by clicking the middle mouse button. A duplicate of the text will now appear at the text cursor position.

Some Linux applications also have a Windows style clipboard available. To copy to this clipboard, press CTRL+C. Then, select your paste location and press CTRL+V.

Special Characters:

Character	Function	Examples
\	Escape character. If you want to reference a special character, you must “escape” it with a backslash first.	touch /tmp/filename\ <i>*</i>
/	Directory separator, used to separate a string of directory names.	/usr/src/linux
.	Current directory. Can also “hide” files when it is the first character in a filename.	.htaccess
..	Parent directory	cd ..

~	User's home directory	cd ~
*	Represents 0 or more characters in a filename, or by itself, all files in a directory.	More details in next section
?	Represents a single character in a filename.	hello?.txt can represent hello1.txt, helloz.txt, but not hello22.txt
[]	Can be used to represent a range of values, e.g. [0-9], [A-Z], etc	hello[0-2].txt represents the names hello0.txt, hello1.txt, and hello2.txt
	“Pipe”. Redirect the output of one command into another command.	ls more
>	Redirect output of a command into a new file. If the file already exists, over-write it.	ls > myfiles.txt
>>	Redirect the output of a command onto the end of an existing file.	echo .Mary 555-1234. >> phonenumbers.txt
<	Redirect a file as input to a program.	more < phonenumbers.txt
;	Command separator. Allows you to execute multiple commands on a single line.	cd /var/log ; less messages
&&	Command separator as above, but only runs the second command if the first one finished without errors.	cd /var/logs && less messages
&	Execute a command in the background, and immediately get your shell back.	find / -name core > /tmp/corefiles.txt &

Wildcards:

If you need to refer to a list of multiple files, you can use **wildcards**. These are characters that have a special meaning to the shell. The two most important wildcards are the asterisk (*) and question mark (?). An asterisk will match any string of zero or more characters in a file name, while the question mark will match any single character in a file name. For example:

The pattern...	Will match...	But not...
f*	f1, f2, f3, fruit.txt	somefile.txt, 1f
*.txt	somefile.txt, pasta.txt	pasta.txt.bak, somefile
f?.txt	f1.txt, f2.txt, fz.txt	f1, f1.txt.bak, f.txt
.	fruit.txt, pasta.txt, pasta.txt.bak	f1, somefile

Linux "help"

If you don't know how to do something, Linux has a built-in help system that you can use. The most important tools are listed below:

man <i>command</i>	Display the manual page for a command.
apropos <i>keyword</i>	Print a list of commands apropos to a keyword with a one-line explanation for each command.
whatis <i>command</i>	Print the one-line explanation for a command.
which <i>command</i>	Shows the full path to command .

REFERENCES:

1. <http://linuxbasics.org/>
2. http://personal.denison.edu/~bressoud/cs372-f04/linux_tutorial.pdf
3. <http://www.e-fense.com/helix/Docs/Law.Enforcement.Linux.Intro.2.0.5.pdf>
4. <http://vic.gedris.org/Manual-ShellIntro/1.2/ShellIntro.pdf>
5. <http://www.zzee.com/solutions/linux-permissions.shtml>

Note: Much of the material in the Unix Background Information section extracted directly from the sources listed above.

Unix/Linux Lab

Purpose:

This lab will cover basic commands used in Unix/Linux. You will also be introduced to running and using VMware. We will run a VMware image containing Linux Fedora Core 5.

Software Requirements:

VMware Workstation (Installed in the labs-commercial) or
VMware Server <http://www.vmware.com/products/server/> or
VMware Player <http://www.vmware.com/products/player/>

VMware Image called: “fc5mini.zip” will be provided.

OPTIONAL:

PuTTY <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

REFERENCES:

1. <http://www.vmware.com/virtualization/> (VMware)
2. <http://www.chiark.greenend.org.uk/~sgtatham/putty/> (Putty)

PART 1 – Running the VMware Image

NOTE: Guest O/S refers to the VMware image running on the Host O/S. As an example, if the Labs run Windows XP Professional and we run a Linux VMware image on top of that, then:

The Host O/S is: Windows XP Professional and

The Guest O/S is: Linux Fedora Core 5

1. Start the Host O/S and log in.
2. Download “fc5mini.zip” and extract it to the root of your C: drive.
3. Inside the folder, double click the “Other Linux 2.6.x kernel.vmx” file.
4. Power On the Virtual Machine by clicking “Start this virtual machine”
5. Log in to Linux with the following credentials:
 - a. Username: **root**
 - b. Password: **isa.566**

PART 2 – Getting Familiar with Linux-Exercise 1

Enter these commands at the UNIX prompt, and try to interpret the output. Ask questions and don't be afraid to experiment:

Note: the ← represents and enter on your keyboard.

- echo hello world ←
- passwd ←
- date ←
- hostname ←
- arch ←
- uname -a ←
- dmesg | more ← (you may need to press q to quit)
- uptime ←
- who am i ←
- who ←
- id ←
- last ←
- finger ←
- w ←
- top ← (you may need to press q to quit)
- echo \$SHELL ←
- echo {con,pre}{sent,fer}{s,ed} ←
- man "automatic door" ←
- man ls ← (you may need to press q to quit)
- man who ← (you may need to press q to quit)
- lost ←
- clear ←
- cal 2000 ←
- cal 9 1752 ← (do you notice anything unusual?)
- bc -l ← (type quit ← or press Ctrl-d to quit)
- echo 5+4 | bc -l ←
- yes please ← (you may need to press Ctrl-c to quit)
- time sleep 5 ←
- history ←

PART 3 – Getting Familiar with Linux-Exercise 2

Try the following command sequence:

- `cd`
 - `pwd`
 - `ls -al`
 - `cd .`
 - `pwd` (where did that get you?)
 - `cd ..`
 - `pwd`
 - `ls -al`
 - `cd ..`
 - `pwd`
 - `ls -al`
 - `cd ..`
 - `pwd` (what happens now)
 - `cd /etc`
 - `ls -al | more`
 - `cat passwd`
 - `cd ~`
 - `pwd`
1. Change back into your home directory.
 2. Make subdirectories called `work` and `play`.
 3. Delete the subdirectory called `work`.
 4. Copy the file `/etc/passwd` into your home directory.
 5. Move it into the subdirectory `play`.

Part 4 – Questions

1. What is the difference between regular users and the root? Can I have multiple users administering the system? (root is a super-user, having access to all files and devices, all users that belong with id 0 or in some Unix flavors to group 0 (wheel) are also superusers)
2. What command can be used to isolate the line in /etc/passwd that contains your login details? _____ . (grep)
3. What is the command to change access to the passwd file in your home directory so that only you can read or write to it? _____ chmod 600 passwd
4. Explore /dev.
 - o Which is your identify your tty (terminal) device? _____ tty1 (whoami or w)
 - o Is your tty character-oriented or block-oriented? _____ character (ls -l)
 - o Who is the owner of your tty? _____ root (ls -l)
4. What is the command to find all files larger than 1 MB? _____ find . -type f -size 200
5. What PID is Portman using? _____ 1337
6. What command can be used to display the current memory usage? _____ top

Nmap Lab

Purpose:

This lab will guide the student in installing the nmap tool and GUI in Windows. The student will then perform a set of exercises that will familiarize them with the basic functionality of the nmap scanning tool using both Linux and Windows.

SOFTWARE REQUIREMENTS:

Nmap for Windows <http://download.insecure.org/nmap/dist/nmap-4.20-setup.exe>

VMware Image called: “fc5mini.zip” will be provided. It has nmap already installed.

OPTIONAL:

MS .NET Framework 2.0 (Needed by Nmap GUI to work properly)

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5>

Nmap GUI for Windows <http://www.craftysoftwares.com/SWdownload/nmapview.zip>

REFERENCES:

1. <http://insecure.org/nmap/> (Nmap)

Part 1 – Installing Nmap on the Windows Guest

1. On the Windows Host O/S, download Nmap for Windows from the link above.
2. Double-click on the nmap-4.20-setup.exe file to install it. Follow the install wizard and accept the default values.

Part 2 – Obtaining Your IP addresses

1. Use the ifconfig command in Linux and the ipconfig command in Windows to determine your IP address and record it below:

Windows (Host O/S): _____

Linux (Guest O/S) : _____

Part 3 – Performing a Scan of the Local Network

1. For the following steps, please use the nmap command line tool installed on the Linux Guest.
2. Scan your subnet to determine how many hosts can be found. For example, if you are on the 192.168.1.0 subnet, you would enter the following command:
nmap -sP 192.168.1.*

What is your subnet? _____

How many hosts were found? _____

3. Next perform a stealth scan (Please use the IP for your subnet):

nmap -sS -P0 -p 192.169.1.*

4. Now, you'll perform an OS identification. Use the Linux O/S to scan your Windows machine:

nmap -O Windows_IP_ADDRESS

OS Type 1: _____

Now we want to use the Windows machine to scan the Linux O/S. Go to a Windows DOS prompt and enter the following command:

nmap -O Linux_IP_ADDRESS

OS Type 2: _____

5. Now we will perform a service selection scan. Let's scan for all computers with FTP running. We would do that as follows:

nmap -p21 192.168.1.*

List the IP addresses with that has the FTP open: _____

Part 4 – Questions

For these questions please make use of the nmap man file.

1. What is the difference between a TCP-connect scan and a SYN scan? **SYN Scan is a half open scan that is less likely to be logged; however it requires a privileged user account to use it. The TCP connect make a full connection. This does not require a privileged account but is likely to be logged.**
2. What is the purpose of the `-sP` command line switch? **Ping scan. Used to determine which hosts are up.**
3. What is the purpose of the `-sS` command line switch? **TCP SYN scan, A.K.A half open scan.**
4. How accurate is the OS identification facility of nmap? **The OS identification is only an estimate and as good as the OS fingerprint database.**
5. What is the name of the file that OS fingerprints are stored in for nmap? Where is it located? **`/usr/share/nmap/nmap-os-fingerprints`**
6. What are the three main functions of nmap? **Host scanning, port scanning and OS Identification**
7. What command would you issue to scan for computers running web servers? **`nmap -p80 192.168.0.*`**
8. What nmap command was issued to receive the following output (find ???)?
`nmap -????? -???? 10.0.0.108`

```
Starting Nmap 4.20 ( http://insecure.org ) at 2007-09-05 05:02 Eastern Daylight Time
Initiating ARP Ping Scan at 05:02
Scanning 10.0.0.108 [1 port]
Completed ARP Ping Scan at 05:02, 0.30s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 05:02
Completed Parallel DNS resolution of 1 host. at 05:02, 0.13s elapsed
Initiating ACK Scan at 05:02
Scanning 10.0.0.108 [1697 ports]
Completed ACK Scan at 05:02, 0.11s elapsed (1697 total ports)
Host 10.0.0.108 appears to be up ... good.
All 1697 scanned ports on 10.0.0.108 are UNfiltered
MAC Address: 00:0C:29:00:DF:CA (VMware)
```

```
Nmap finished: 1 IP address (1 host up) scanned in 0.687 seconds
Raw packets sent: 1698 (67.922KB) | Rcvd: 1698 (67.922KB)
```

`nmap -sA -v 10.0.0.108`

9. What is the purpose of the command illustrated in question 8? **This is an ACK scan. Is used to map out firewall rulesets.**

Nessus Lab

Purpose:

In this lab we will explore vulnerability scanning using an open source tool called Nessus. This tool is already installed in your Linux VMware image. You will need to install the Windows client on your Host O/S and then use it to scan your Windows Machine.

SOFTWARE REQUIREMENTS:

nessuswx-1.4.5d.zip

REFERENCES:

1. <http://www.nessus.com> (Nessus)

Part 1 – Installing Nessus Client on the Windows Host

1. Point your browser to:
`http://Linux_IP_ADDRESS`
2. Download the nessuswx-1.4.5d.zip file by clicking on “[Windows Remote Console](#)” and extract it to the desktop of you host machine.
3. The NessusWX.exe file is the executable that we will be using.

Part 2 – Connecting from the Windows client to the Linux nessus server.

1. In the Windows host start the NessusWX client.
2. Click on the Communications window and select connect
3. In the name field type the IP address of the Linux guest machine that is running the nessus server.
4. Select the radio button labeled, “**Authentication by password.**”
5. Use the following credentials to connect (click save password):
 - a. Username: **root**
 - b. Password: **nessus**
6. Click on the connect button.
7. You will be prompted about a New Server Certificate, click the Accept & Save button.
8. Click on the **Session** menu and select **New**
9. Give the session a name and click the **create** button.
10. Add a target(s). This could be the address of your Linux guest, Windows Host and/or Windows host computer.
11. Click on the Plugins tab and check the “**Use session-specific plugin set**” checkbox.
12. Click on **the Select plugins** button
13. Select the **Enable Non-DoS** button
14. Select the **No** button for the warning box.
15. Click the **Close** Button
16. Click the **OK** Button
17. Right click on the session icon that you created and select the **Execute** option.
18. Click the **Execute** button.
19. After the scan has finished click the **close** button.

20. Click on the view button to view the report(s).
21. Save one of your reports to a pdf file and print it out to hand in with the lab.

Part 4 – Questions

1. What kind of architecture is nessus based on? **Client / Server Architecture**
2. What is the advantage of the type of architecture referred in question 1? **This allows the central aggregation of your vulnerability assessments.**
3. How is nessus different than nmap? **nmap is a scanning tool while nessus provides enumeration and vulnerability detection.**
4. What is a plugin? **A plugin is the code that allows nessus to test for a specific vulnerability on a target machine.**
5. What formats can you save your reports in? **HTML, PDF and plain text.**
6. List the worst vulnerability that you detected, if any? **This is a freebie question.**
7. Why did we select Non-DoS vulnerability scans? **If we were to scan for DoS vulnerabilities we could potentially crash the machine we are scanning.**

iptables Lab

Purpose:

In this lab you will explore the Linux firewall *iptables*. The first part of the lab will describe *iptables*, then basic commands, followed by some exercises.

SOFTWARE REQUIREMENTS:

None

REFERENCES:

1. <http://www.siliconvalleyccie.com/linux-hn/iptables-intro.htm>
2. <http://www.iptables-tutorial.frozentux.net/iptables-tutorial.html>
3. <http://www.yolinux.com/TUTORIALS/LinuxTutorialIptablesNetworkGateway.html>

Part 1 – Background on iptables

Originally, the most popular firewall/NAT package running on Linux was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called iptables, giving it such improvements as:

- Better integration with the Linux kernel with the capability of loading iptables-specific kernel modules designed for improved speed and reliability.
- Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and in certain cases will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services.
- Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses.
- System logging that provides the option of adjusting the level of detail of the reporting.
- Better network address translation.
- Support for transparent integration with such Web proxy programs as Squid.
- A rate limiting feature that helps iptables block some types of denial of service (DoS) attacks.

Considered a faster and more secure alternative to ipchains, iptables has become the default firewall package installed under RedHat and Fedora Linux.

Part 2 – Basic Commands

The executable binary for the iptables application is the */sbin/iptables* file. So when execute *iptables* this is the file that is executed. RedHat also has an executable script in the */etc/init.d/iptables* file so that it can start its default firewall at startup. An easy way to reset the firewall rules is to use this script as

```
/etc/rc.d/init.d/iptables stop
```

This flushes all the rules in memory and gives you a clean start.

1. You can flush individual rules with

```
iptables -F flushes all chains
```

```
iptables -X <chain name> deletes a chain
```

```
iptables -X deletes all chains
```

2. To check if there are any current rules, type

```
iptables -L
```

It'll show you a couple of chains, but there won't be any rules in them.

You might have to do */etc/init.d/iptables restart* to apply these changes (note: this is a different command than the one you used above).

3. You can start, stop, and restart iptables after booting by using the commands:

```
service iptables start
```

```
service iptables stop
```

```
service iptables restart
```

4. To get iptables configured to start at boot, use the chkconfig command:.

```
chkconfig iptables on
```

5. To get iptables configured to start at boot, use the chkconfig command:.

```
service iptables status
```

Part 3 – Exploring iptables

1. Start the VMware Linux guest O/S and log in.
2. Determine the IP address by using **ifconfig**.
3. Use putty (refer to page 15) to connect to your guest O/S from your Windows O/S.
4. Navigate to the following directory: */root/iptables*
5. Enter the following command **ping gmu.edu**
6. Now let's block access to this site, enter the following command:
./block-ip 129.174.1.52
7. Now try to **ping gmu.edu** again. (What happened?)
8. Explore the contents of the four files listed in that directory: */root/iptables*
9. Enter the following command:
./accept-all
10. Again try to **ping gmu.edu**
11. Now go ahead and enter this command:
./deny-all (what happened?)
12. You will need to access the Linux guest O/S through the VMware interface and enter this command:
/root/iptables/accept-all
13. Now you should be able to log in through putty again.

Part 4 – Questions

1. What is the command to block the following subnet: 192.168.2.x?
2. Describe what the following rules do:

```
iptables -A INPUT -i lo -p all -j ACCEPT
```

Rule for your computer to be able to access itself via the loopback

```
iptables -A INPUT -p all -s localhost -i eth0 -j DROP
```

Deny packets which claim to be from your loopback interface.

```
iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT
```

iptables is being configured to allow the firewall to accept TCP packets coming in on interface eth0 from any IP address destined for the firewall's IP address of 192.168.1.1. The 0/0 representation of an IP address means any.

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP \  
--sport 1024:65535 --dport 80 -j ACCEPT
```

iptables is being configured to allow the firewall to accept TCP packets for routing when they enter on interface eth0 from any IP address and are destined for an IP address of 192.168.1.58 that is reachable via interface eth1. The source port is in the range 1024 to 65535 and the destination port is port 80 (www/http).

3. Assume you are running a web server that you would like to secure with iptables. Discuss in detail how you would go about doing that.