EDURange Student's Manual

Introduction

EDURange is an NSF-funded project that is both a collection of interactive, collaborative cybersecurity exercises and a framework for creating these exercises. This suite of exercises is intended to help supplement classroom lectures, labs, and other activities.

We believe EDURange has a potential to significantly advance the integration of cybersecurity into the undergraduate computer science curriculum. By providing interactive, competitive exercises, it will enhance the quality of instructional material, and increase active learning for students. The ease of use for instructors will encourage them to integrate cybersecurity into the current core curriculum. These exercises will also provide rapid feedback to students and faculty, which will aid in assessment of student learning.

A key question we ask ourselves in our design process is: *What kind of analysis skill do we expect the students to acquire from running (and re-running) this scenario*?
Our central and very intense focus is on creating exercises that support and nurture the development of analysis skills rather than memorized scripts, recipes, or standard command line and GUI settings for a particular tool. Though some exercises revolve around using a specific tool, the main learning goal is the development of the analytical skills and understanding of the complex system which that tool acts upon.

By keeping this focus, EDURange helps students buy into the process of sharpening their information security analysis skills and makes them a partner in evaluating and understanding the limits of those skills.

Our cybersecurity exercises cover the topics of Network Analysis and Reconnaissance, Malware Detection and Analyzation, Network Traffic Analysis and Defense, Social Engineering, and Web

Security. Though our exercises can be done in any order, some can be good building blocks that work towards the more advanced ones. Most of the exercises require a minimal level of understanding of some standard Linux tools. We have provided some basic tutorials for Linux use at the end of this document.

Using EDURange

Your instructor will give you an access code that you can use to register for EDURange. You will register at http://cloud.edurange.org. When your instructor creates an exercise and places you in a group, it will be accessible to you in the scenarios section of your EDURange home page. From there you will be able to see your login credentials as well as your initial instructions. You can use that to connect via ssh to the public IP address of a gateway for that exercise. Some exercises will also have questions to be answered via the EDURange scenario page. Others have questions listed in this manual that your instructor might have separate instructions for.

Exercises

Getting Started

Description

Getting Started is a scenario to help you learn the basics of Linux/Unix command line.

Learning Objectives

- Understanding find
- Understanding case-sensitivity
- Understanding file
- Understanding the terminal
- Understanding the file structure of a Linux computer

Instructions

The beginning

The beginning

Congratulations recruit and welcome to CyberSec. As you should know we provide security services to you, our communities, and small businesses. You've made it this far, I suppose that means you may be helpful. But first comes the training. We need to strengthen your skills before you can truly be of use. Go through each tab to the left and at the end we will test what you have learned. Remember, this world is being controlled by the malicious corporations and individuals and we must learn to protect ourselves from their invasive attacks. We must protect our data. This is why you are here. To protect yourself, your company and your community. And it all starts here, at the beginning, with a terminal command.

Terminal

When using your computer (Mac, Linux, Windows) you typically are using a GUI (Graphical User Interface). It's a pretty representation of how your computer works. To really get into the 'guts' of your computer and to really learn how to control it we will learn how to use the terminal. The terminal is a texted-based representation of your computer (rather than graphical). Learning to use the terminal will help you along your path to protecting your community and your self.

Why should you learn the command line?

- You gain greater control over the system (computer)
- A GUI interface just doesn't have the power needed to run repetitive tasks.
- Doing anything from a simple task of renaming a file, changing a user information and searching for files is faster and easier through command line once it is learned.
- Scalability
- Scriptability
- Simple design
- Simple interface
- Stable design

In summation it allows you to do stuff faster than GUI and provides an amazing automation support built in.

Open up terminal for the rest of this training. Or ssh into the server for the command line experience.

Linux File System

Hierarchy

Linux folders and files are arranged like an upside down tree. Where the slash / is called the root, or beginning, of all your files in the entire computer. The root is the base of the tree and as you go up it keeps splitting into branches and leaves. The leaves would be a file and the branches are folders.

/ vs. logging in as root

[illegible]

The root, signified by a / , is the beginning of your files. But you can also log in as the root user. When you do this, your home directory (where your files are typically saved) is in the folder /root not at, /. The /root folder is not to be confused with the slash (root) the beginning of all the files. Just like if you were logged in as bermic you would typically save your files in /home/bermic, whereas the root user saves their files in /root.

Important, a root user is someone who has access to everything on the computer. They could even delete everything in a computer. It is best practices to disable root or use a VERY strong password. For example using numbers, letters, capitals, special symbols and a random sampling of each, and no dictionary words.

4 Commands

What is a command? A command is something you type into your terminal to make something happen. That's a bit vague but essentially the idea. You can do a lot of things with commands. Let's get started right away and use two different commands. Let's go to your home directory.

1 cd

Type each of the following. One at a time. Hitting enter after each

```
~$ cd /
~$ cd /root
~$ cd
~$ cd ../
```

The first command sends you to the root of your entire file system.

The second command sends you to the user root folder

The third command sends you to your home directory

The fourth sends you backwards (up) a level.

TASK: cd to /bin then cd back to your home directory.

2 pwd

Now type in

```
~$ pwd
```

Hit enter, pwd is a command to print your working directory. In other words, it prints your location so you can see where you are. You should see something similar to /home/yourusername. To learn more type man pwd , then to get out of that page type q

ls

Type

```
~$ ls
```

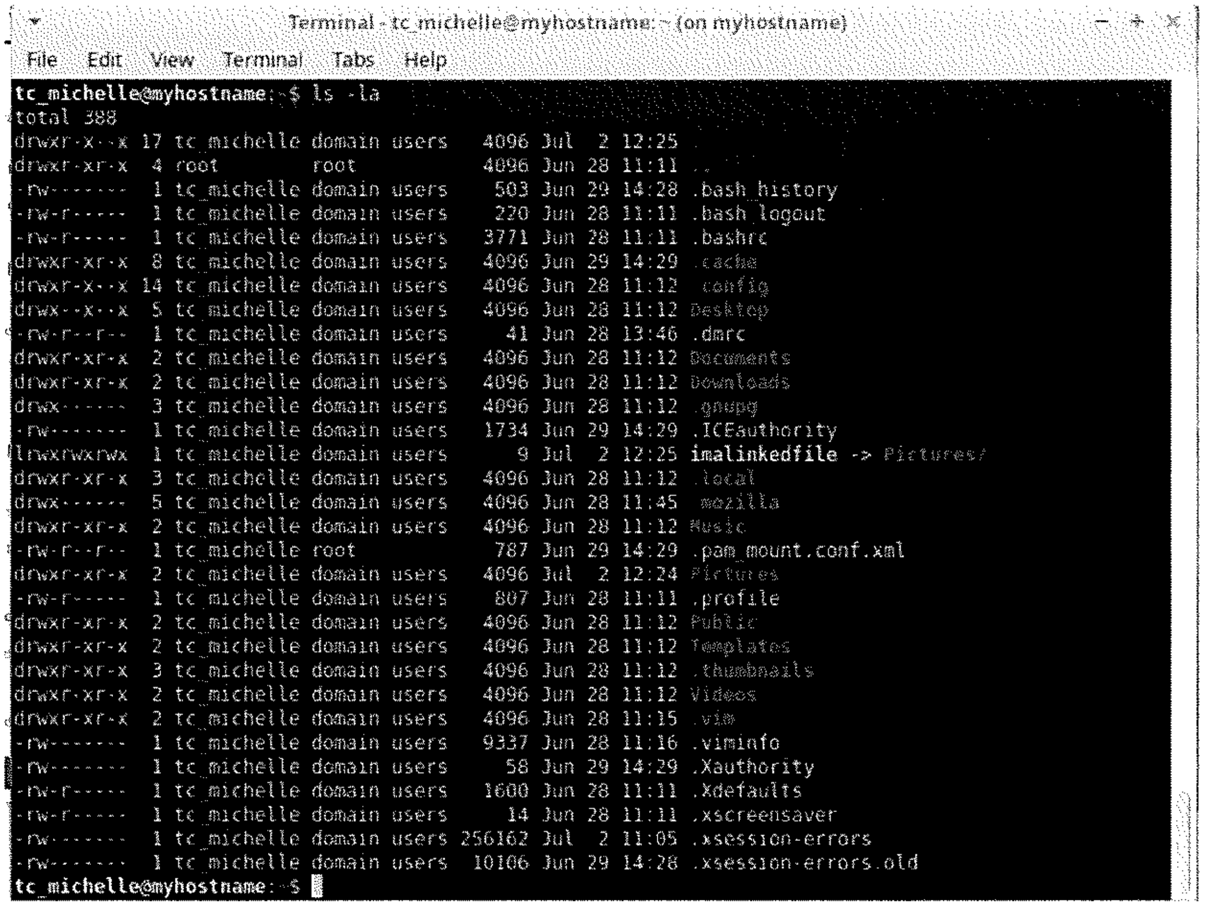
Hit enter, ls lists the files and directories of where you are now.

There are parameters and options you can give a command. What if you wanted to list the permissions of a file and find hidden files? (Yes there are hidden files!)

Now type

```
~$ ls -la
```

Then hit enter.


Terminal - tc_michelle@myhostname: ~ (on myhostname)
File Edit View Terminal Tabs Help

```
tc_michelle@myhostname:~$ ls -la
total 388
drwxr-x--x 17 tc_michelle domain users   4096 Jul  2 12:25 .
drwxr-xr-x  4 root        root           4096 Jun 28 11:11 ..
-rw-------  1 tc_michelle domain users    503 Jun 29 14:28 .bash_history
-rw-r-----  1 tc_michelle domain users    220 Jun 28 11:11 .bash_logout
-rw-r-----  1 tc_michelle domain users   3771 Jun 28 11:11 .bashrc
drwxr-xr-x  8 tc_michelle domain users   4096 Jun 29 14:29 .cache
drwxr-x--x 14 tc_michelle domain users   4096 Jun 28 11:12 .config
drwx--x--x  5 tc_michelle domain users   4096 Jun 28 11:12 Desktop
-rw-r--r--  1 tc_michelle domain users     41 Jun 28 13:46 .dmrc
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:12 Documents
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:12 Downloads
drwx------  3 tc_michelle domain users   4096 Jun 28 11:12 .gnupg
-rw-------  1 tc_michelle domain users   1734 Jun 29 14:29 .ICEauthority
lrwxrwxrwx  1 tc_michelle domain users      9 Jul  2 12:25 imalinkedfile -> Pictures/
drwxr-xr-x  3 tc_michelle domain users   4096 Jun 28 11:12 .local
drwx------  5 tc_michelle domain users   4096 Jun 28 11:45 .mozilla
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:12 Music
-rw-r--r--  1 tc_michelle root            787 Jun 29 14:29 .pam_mount.conf.xml
drwxr-xr-x  2 tc_michelle domain users   4096 Jul  2 12:24 Pictures
-rw-r-----  1 tc_michelle domain users    807 Jun 28 11:11 .profile
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:12 Public
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:12 Templates
drwxr-xr-x  3 tc_michelle domain users   4096 Jun 28 11:12 .thumbnails
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:12 Videos
drwxr-xr-x  2 tc_michelle domain users   4096 Jun 28 11:15 .vim
-rw-------  1 tc_michelle domain users   9337 Jun 28 11:16 .viminfo
-rw-------  1 tc_michelle domain users     58 Jun 29 14:29 .Xauthority
-rw-r-----  1 tc_michelle domain users   1600 Jun 28 11:11 .Xdefaults
-rw-r-----  1 tc_michelle domain users     14 Jun 28 11:11 .xscreensaver
-rw-------  1 tc_michelle domain users 256162 Jul  2 11:05 .xsession-errors
-rw-------  1 tc_michelle domain users  10106 Jun 29 14:28 .xsession-errors.old
tc_michelle@myhostname:~$
```

That's a lot of info! What you see is all the files and folders in the folder you are at currently.

- The first column is the type of file followed by permissions. – means a regular file. d is a directory (folder). l is a link. rwx are the permission for each file. rwx stands for 7 so rwxrwxrwx would be 777. These correspond to binary. There are 3 bits. 000 would stand for 0. 111 is 7. 101 is 5, etc. Each file has visible 3 permissions User, Group, Anyone.

- The next column is the number of links or directories in the folder
- The 3rd column is the user that owns the folder/file
- The 4th column is the group that owns the folder/file
- The 5th is the size of the file/folder
- The 6th is the month day and time it was last edited/touched
- And finally the file name

TASK: cd into the /tmp folder and then type ls and you will see a directory called "follow_Me". Travel as deep as that folder will go. When you get to the end there is a file whose name is a randomized number. Find that.

sudo

Sometimes commands can only be run as a super user. This is when the command sudo comes to use (which stands for: superuser do). This gives unprivileged users access to privileged commands. If you try to do something and it says Permission denied, try again with sudo in front of the command.

Man pages

Man Pages is short for manual pages. These are text documents with lots of information on commands. Remember the command we did for listing our files? ls! Let's find that man page.

Type

```
~$ man ls
```

and hit enter

Remember we typed ls -la? Let's learn what -l and -a is!

-l
To search inside a man page, you use a / . Now type /-l and hit enter. You will probably see the page move to the first occurrence found and on top of that you should see that -l was highlighted*. To move around your search keep hitting n (stands for next) until you see -l highlighted to the left. This will be above -L
Long listing stands for listing the items in a row as seen in figure 2

- a

Type /-a and hit enter. Hit n till you can't go any further. Now hit b (stands for back) until you find the entry for -awhich is above -A
The man page is telling you that files that start with . like .bashrc (which are typically hidden) are now going to be displayed.

q

When in a man page and you need to get out, just type q

TASK: Open the man page for "file". Can you give a brief description of what the command "file" does?

* If you don't see it highlighted, you may have typed something by mistake or your console colors may not be optimized. If you typed something by mistake, just retype /-l etc.

6

File Types

Not all files appear as they really are. Just because you see a file that says, imanimage.png does not mean that it is an image. It could be a text file or a harmful file if executed! So... how do you protect yourself? One way is with the filecommand!

5 **file**

To find out what a file really is regardless of its extension is file. Check out the man page. Give it a peruse by typing in man file. What type of options are there with file? Now let's test it. Type q to get out of the man page.

There are 2 files in your Linux box in a folder at the location /tmp/toLearn . One is called cat.jpg and the other is dog.jpg

Both look like images to me! But if you type in ls -l you will notice that one is a lot larger in size than the other. One is about 25,000 bytes whereas the other is only about 20. Now let's see what is really going on.

Type

```
~$ file dog.jpg
```

You'll see something like, dog.jpg: ASCII text

Now type

```
~$ file cat.jpg
```

You'll see something like, cat.jpg: JPEG image data, Exif standard: [TIFF image data, ... etc.

6 **cat**

Now let's learn a new command, cat. cat prints out the text from a file.

Type

```
~$ cat dog.jpg
```

You should see something like:

```
meow I am a doggo
```

TASK: In the tmp directory there is a folder called stuff. Open that up and find out what file types are in there. One is a text file (ASCII). cat that and find the secret code inside.

Case Sensitivity / touch / echo / Carrots

Case Sensitivity

Case sensitivity means that HoW yoU labEL yoUR files matters. If you search for a file called hiya.docx, it would not be the same as finding a file, hiyA.docX.

touch

touch is a command that 'touches' a file. If the file exists it updates its modified date. If the file does not exist, then the file will be created with nothing in it. man touch to learn more.

echo

echo will copy what you write to stdout (standard out, explained more later). You can use this in many different ways.

Type

```
~$ echo "This is echoed"
```

You will see that it was repeated back to you!

Carrots

Carrots are good for you, but I am talking about > >> < << . These are called carrots. Carrots have many uses. > Will replace a file with what you input. If the file already existed > will delete **everything** in that file and replace it with what you sent it. In contrast if you use >> , this will append what you sent to the bottom of the file, leaving the rest of the file intact. Let's give it a try.

Type

```
~$ echo "This is cool" > newfile
~$ cat newfile
```

Now type

```
~$ echo "This is cool too" > newfile
~$ cat newfile
```

You can see that > will replace any text with what you send it. While >> will append to a file

Type

```
~$ echo "This is another thing" >> secondfile
~$ echo "Hello World" >> secondfile
~$ cat secondfile
```

Now let's combine the two files.

Type

```
~$ echo newfile >> secondfile
~$ cat secondfile
```

You can see the newfile appended to the end of secondfile whereas if you cat newfile it will still only have what we added to it earlier.

TASK: There is a folder in the /tmp directory called textfiles. There are three files, append them all to a new file called, alltogether.txt

There are many ways to accomplish this in one line. It's not necessary but here is a hint for one way, type

echo "one"; echo "two"; echo "three";

vim, Regular expressions and find

vim

vim is a program that is used to edit files, and will hopefully be your new best friend! There are different editors out there for example, nano and emac. To create a file just type

```
~$ vim mynewfile.txt
```

Or

```
~$ vim thisisfun
```

To edit a file that is already created it's the same procedure, just make sure not to misspell it or you'll create a new file with that spelling.

Once you are in vim the main key strokes to editing a file are:

i - This puts you in edit mode to type and delete text like you normally would

esc - Hitting the escape key will take you out of edit mode

:w - These keystrokes will save the file. w stands for write.

:q! - To quit without saving. Did you edit a file and don't want to commit that change? These keystrokes will exit vim and NOT save your file.

:q - These keystrokes will quit the vim program. You can also do :wq to save the file and quit right away. :q will not work unless you have saved your file or you have made no changes what-so-ever.

dd - When you are NOT in edit mode this will delete an entire line. (Make sure your cursor is on the line you want to delete)

0 - zero will take you to the beginning of a line

$ - will take you to the end of a line

There is a LOT that vim can do but we won't list it all here. Do a search on the internet to learn more! You can also check out a vim command cheat sheet here and here. But at the end of this lesson will be a couple more commands that you will find to be amazingly helpful! You can also check out vimtutor:

```
~$ vimtutor
```

TASK: There is a file in /tmp/editme called editme.txt Open that up in vim. Delete lines 4 and 5 and add 2 more lines of anything you would like at the end of the file. Don't forget to save.

wild card

Regular expressions

Regular expressions are used to help you find something on your computer and can be used in programming to enhance your programs. There are a LOT of websites out there that teach you all about it but the gist is that you can use symbols like a * to mean something when parsing through text. For example the * is a wild card. Let's say I wanted to find a file with the word spekter in it. But there could be other text before and after the word spekter. So I could say search for, *spekter*. This means, search for spekter and I don't care if there is anything else before or after.

10 **find**

This leads us to find. find is a very helpful tool that you can use to find things on your computer. Take a moment to peruse the man page for find. (man find) Get an idea of how it is used.

An example as given earlier:

```
~$ find . -type f -iname *spekter*
```

- What you see is the command find. The next '.' is telling us where we want to find. It's the path. The dot means, search in this location where I am at. We could also type in /Documents or a full path from the root. Where ever you need to search.
- The parameter -type is telling find that you specifically want to find a type of object, in this case, f stands for a regular file.
- -iname is telling the name of the file we are looking for. You can also use -name but -iname is case insensitive. This means that my search will pull up, SpeKter as well.
- Lastly, the *'s on either side again tell find that I want everything with spekter in it, regardless of what is around it. If I did not add that, my find will pull up nothing.

TASK: Hidden throughout are image files with the name, edurange. Take your skills and find all 5. Create a new file and put the location and type of each in that file. Remember there are many types of file images. Png, jpg, jpeg, and gif to name the widely used ones.

9

More Commands

11 **mv**

mv is used to move a file from one location to another, or to rename a file. Type man mv to learn more.

12 **cp**

cp is used to copy a file to a new location. Type man cp to learn more.

13 **less**

less is more. The command less is used to open larger files, page by page. It allows you more tools to read a file in an organized fashion.

[illegible]

14 **cowsay**

cowsay is probably one of the best commands... ever. Well maybe not ever but it is fun!

```
~$ cowsay "this is fun"
```

15 **fortune**

fortune gives you, well, a fortune! Go ahead give it a whirl!

```
~$ fortune
```

piping

input program

Piping is a new concept but stick with me on this one. When you give arguments to a command it is considered a standard in (STDIN). In other words it is data that is fed to a program/command. Whereas when you see something printed out back to you it is using standard out (STDOUT). A pipe, also recognized as | is used when you want something that is a STDOUT to then be used as the STDIN. To understand this we will use a fun example.

Remember fortune? If we just type fortune we get a fortune back. That fortune we get back is a STDOUT. But when we use cowsay we type something for the cow to say, which is STDIN. So we can use a pipe to take our fortune and have the cow say it!

```
~$ fortune | cowsay
```

More vim

There are a few more really useful commands I would like to teach you about vim.

If you want to open up multiple files at the same time you can do:

```
~$ vim fileone filetwo.c filethree.h filefour.cc
```

Or you can open up another file or create a new file while you are still in vim with:

```
:e anotherfile.js
```

:b - And you can jump from file to file with the :b keystrokes and using tab to go through which file you would like to edit next

Final Mission

Here at CyberSec you have been well trained recruit, now let's put your training to test! This is a real mission of the utmost importance. Complete this and you are assured a spot on our team.

TASK: Our intelligence has told us that somewhere in your computer is an image that is vital to our company. This image has a weird name but we know for certain that it contains the word 'cow' in it. Through our sources we also know that there is ANOTHER file in that folder. **COPY** both the image and second file to the folder, /tmp/final-mission. Then create a new file in the folder, /tmp/final-mission, called 'cowsay.file' with the contents of the cowsay man file in it.