UNIX/Linux Security

35 Points

Due Date: Start of Lesson 5

1 Overview

Even though the Army largely uses Microsoft Windows server systems, many other useful operating systems exist. One of the oldest and most stable operating systems is UNIX. UNIX is often used for enterprise-class servers and computing systems because it is very reliable; was designed with a modular, scalable, and easily programmable architecture; and supports a wider range of device classes than Windows. Many flavors of UNIX exist, including Solaris, IBM AIX, HP-UX, the various Open Source BSDs, QNX, Mac OS X, and Linux. Since the 1990s, Linux has arisen as a widely deployed UNIX clone because it was designed to run on commodity computers and released under an Open Source license.

In this exercise you will learn some of the fundamental skills of administering common UNIX-based operating systems. While the file locations may be different between UNIX flavors, the concepts and practices are the same. Here we will focus on `Ubuntu`, one of the most popular and widely used Linux distribution.

Your deliverables for this lab are your answers to all the questions. Details are at the end of this lab document.

2 Getting Started

The primary method of completing this lab is use the attacker VM on the vSphere cluster. Log in to the your vSphere segment using the procedures on the course webpage:
http://www-internal.eecs.usma.edu/courses/cs482/setup/setup.html. There will also be an OVF available for offline use if needed. Periodically back up your work!

Log into your attacker VM using the following credentials:

```
Username: eecs
Password: scee
```

You should now see the following prompt:

```
eecs@localhost:~$
```

Take a snapshot of your VM before you begin; give it a good clear description in case you need to go back to this snapshot later in the future.

3 UNIX Users and Groups

UNIX user accounts permit accountability on a system, allowing us to apply access controls and track activity on an individual basis. UNIX systems also support groups which provide a means to collect users together to apply access controls to many users at once.

UNIX stores user and group information along with password hashes in plain text files within its filesystem. This makes it easy to enumerate the user configuration of the system, but it also makes it relatively

easy for an attacker to steal the password hashes for a dictionary-based attack. You can list the user accounts in `Ubuntu` by typing `users`. A users account informationsuch as his name, home directory, and shell is stored at `/etc/passwd`. You can also see which users are currently logged into the system with the 'users' command.

We will now examine the `/etc/passwd` file. Each line in this file describes one user account on the system. To view the file, execute the command:

```
less /etc/passwd
```

Use the up and down arrows to scroll through the file, and press the '/' or '?'key to search the file down or up, respectively. (Use the 'q' key to exit less.) See page 5-8 of chapter 25.3 (online addition chapter) in your text for a description of the /etc/passwd file. Using the information from the text, answer the following questions:

Question 1: What is the User ID (UID) of the root user?

Question 2: Is it possible for another user account to have the same UID as the root user? What would be the implications of this?

Question 3: What is the User ID of your user account (eecs)?

Question 4: What is the Group ID of the root account?

Question 5: What is the Group ID of the primary group for your user account (eecs)?

Question 6: What is the logon shell for the user michael? What does this mean?

Now lets investigate the `/etc/group` file. Use the command to display a file from the previous step, changing the file to `/etc/group`, to view the groups on the system. See chapter 25.3 of your text for a description of the `/etc/group` file.

Question 7: Which users are currently in the instructors group?

Question 8: Which users are currently in the sudo group?

The sudo group (also known as wheel group on other Linux distros like FreeBSD) provides additional abilities to its members. Sudo group members have the ability to elevate their standard user account privileges to root (i.e., system administrator) level. Many applications also use the sudo group as a gateway group to allow users special abilities that normal users would not have. You will learn more about this group when we discuss sudo below.

As we saw above, `/etc/passwd` contains only a placeholder for the password hashes. We will explain hashes and hashing functions later in the course, but they are essentially one-way functions that generate a fingerprint for a possible input, i.e. text like a file or password. It is computationally infeasible to reverse said fingerprint. In modern UNIX/Linux systems the password hashes are not located in `/etc/passwd`, since this file must be world-readable. On older systems the hashes were in /etc/passwd and any user could copy all the user names and password hashes for an off-line dictionary attack (again we will talk about this later in the course). Today, the hashes are shadowed and stored in either the `/etc/master.passwd` file (on older UNIX) or `/etc/shadow` (on Linux). Only users with root privileges can read this file.

4 Using elevated privileges: SU and SUDO

As a best practice, you never want to give your users access to the root account as they could cause unspeakable harm to the system. This is a gross violation of the principle of least privilege. Additionally, you do

not usually want to log in as the root user because a simple typo might cause a system failure (e.g., deleting the /etc directory where all of the system configuration files are stored). UNIX offers security mechanisms, such as `su` and `sudo` that allow non-privileged users to temporarily assume elevated privileges as needed.

One way to temporarily assume elevated privileges is the `su` (substitute user) command. This allows you to assume the identity of the user name specified, defaulting to the root account if no name is specified. Try executing:

```
whoami
su cs482 (the password for cs482 is 284sc)
whoami
exit (you are back to eecs)
su  (the root password is toor)
whoami
exit
```

Note that `su` executes a new shell, running as the requested user and that exit terminates this shell, returning to the shell which had been running as the previous user.

Giving someone the root password is giving him complete control of the system. Additionally, the only way revoke this access is to change the root password. (An inconvenient requirement every time someone leaves an organization, for example.) In general, the root password should be known to only the most trusted administrators and used only in case of a critical system failure.

`Sudo` provides a more powerful and fine-grained method for granting elevated privileges than sharing the root password. `Sudo` uses the file `/etc/sudoers` to delegate specific privileges to users and groups. In `Ubuntu`, members of the `sudo` group are typically permitted to assume full root privileges via `sudo`. (When someone departs your organization, just remove him from the `sudo` group and his privileges are gone; no password changes needed.) Entering: `sudo` command runs command with elevated privileges.

Let us examine user accounts, privileges and using `sudo` by adding a new account to the system. Enter the command `useradd`. This should fail because you are not root. However, your membership in the `sudo` group allows you to use root privileges, so lets try to add a user with our `sudo` privileges. First, type `man useradd` to learn more about the command. When adding a user, make sure you specify the shell terminal `/bin/bash` and create a home directory for the user if it does not exist. After adding the user, make sure to specific that users password with the `passwd <user>` command. The below commands would create the user 'name'.

```
sudo useradd name -m -s /bin/bash
```

This command creates a user 'name'. Now examine the `/etc/shadow` file (you will need to use elevated privilege). Now, lets add a password to the user name with the command:

```
passwd name  {then type newUser twice when prompted}
```

Those two commands create a user 'name' with a password 'newUser'.

Question 9: Look at the password hash for the account `ben`? What does this tell you about his password?

Question 10: What is the password hash and user ID of the account you just created?

Use `su` to assume the identity of the account you just created. Then enter the command:

```
sudo less /etc/shadow
```

Question 11: What result did you get from issuing that command as your new user? Why?

To allow the new account full elevated privileges, we will add it to the `sudo` group. (Note: we could also use the `visudo` command to edit the `sudoers` file for more fine grained assignment of specific privileges.)

As `eecs`, add your new user to the `sudo` group with this command (hint: requires elevated privileges):

```
addgroup name sudo
```

The only guaranteed way to verify that you added the user to the group is to check the `/etc/group` file we looked at earlier. We use grep to limit our results by typing:

```
grep name /etc/group
(Note: this does the same thing as "cat /etc/group | grep name")
```

Linux has various other utilities that come built in on some version like `groups, members, id`, and `lid` that you can test out on your own.

Question 12: Using any method, list all the members of the sudo group?

Now, use `su` to assume the identity of your new user and then as that user execute:

```
whoami
sudo less /etc/shadow
```

The membership in `sudo` should now allow your new user to view this file while running as your new user.

5 Configure Networking

Now that we can know how to use sudo, we can set up networking for your computer (changing most network settings requires elevated privileges). First, we'd like to know what IP address, if any, has been assigned to your VM. At the command prompt, execute the following command:

```
ifconfig
```

You will likely have three interfaces present, a `lo` as well as two ens interfaces (most likely `ens32` and `ens160`).

Question 13: What does the `lo` interface represent? What IP address is assigned to it? What is the IP address and network mask, if any, currently assigned to the other two networking interfaces?

Your VMs come with two network adapters attached. The one with an assigned IP in the `10.4.82.x` range (generally `ens160`) is the interface used for your Internet connectivity and should not be changed. The other interface without an IP address is the one you will be configuring for your internal network and the one you will use during this class.

Now we will configure network settings. It is possible to set the IP address and default gateway using the `ifconfig` and `route` commands; however, such changes will not persist after a reboot. Thus we will modify the `/etc/network/interfaces` configuration file to make persistent changes. (Note that network configuration files are different on depending on the operating system and vendor.) Start editing the file `/etc/network/interfaces` with the command:

```
sudo gedit /etc/network/interfaces //
NOTE: You may substitute another text editor.
```

In this file, change the IP address to 10.172.XXX.10 and network mask to 255.255.255.0 for ens32, where the XXX in the third octet is the number of your group, i.e. '1' for "CS482 cs482.teamb1". Instructions on how to do this are on the course web-page. You will also need to change your hostname in both `/etc/hostname` and `/etc/hosts`. Change your hostname to:

```
firstinitallastname_attacker.eecs.net
```

so my hostname would be:

```
mkranch_attacker.eecs.net
```

Save your changes and then reboot the VM with the command:

```
sudo shutdown -r now
```

Once the system is again running, log in as cs482. Check your network settings with the commands:

```
ifconfig ens32 ; netstat r; and hostname.   (Note: use your interface)
```

At this point, you should the below as your prompt in the command terminal

```
cs482@firstinitallastname_attacker:~$
```

Question 14: Take a screenshot of `ifconfig` after the changes

6 Managing UNIX File Permissions

Managing file permissions is a key to providing control on a defensible system. You don't want random users to be able to modify important system executables or read another users sensitive files. However, you may want to allow groups of users to share common files if, for example, they are working on a joint project. UNIX file systems support a fairly simple set of file permissions. UNIX allows three modes of access for file objects: read(r), write(w) and execute(x). UNIX further defines three types of subjects: owning user, owning group, and others. Thus we can assign one set of permissions for the file object owner, another for the owning group and a third set for everyone else. The list command with the long flag `ls -l` provides directory listings which include file permissions. Ensure you are acting as the cs482 user and answer the following questions:

Question 15: Who are the user/group owners of the directory: `/lab1/`? What are the r/w/x permissions of this directory and why does this affect using `ls` on this directory HINT: simply doing `ls` in a directory does not show you anything about the directory itself. You either need to go a level higher or look into the `-a` attribute for `ls`?

Make the cs482 account the owner of the `/lab1/` directory (hint: read about the `chown` command using `man chown`).

Question 16: What is the full command you used to take ownership of `/lab1/`?

Try to read the file: `/lab1/skippysList.txt` (this should fail). Use `chmod` to grant everyone read access to the `skippysList.txt` file (again, reference the man page if required).

Question 17: What is the full command (or commands) you used to change the permissions on `skippysList.txt`?

If this succeeded, you should be able to read the file.

Question 18: What is the ‘phrase that pays in the skippysList.txt file?

Issue the command: `touch /lab1/wikil.txt`

Question 19: What are the owning user and group and the file permissions on the file `wikil.txt`? What is the file size?

7 Managing UNIX Processes and Services

Now we will look at managing processes and services running on the system. One thing wed like to know is what processes have network ports open. Enter the command:

```
netstat -na
```

Note: the command `netstat -na` *and* `netstat na` *return similar but not exactly the same results. Make sure you include the '-' for more complete results and consult the man page when moving between Operating Systems for the most up-to-date usage parameters.*

This should return lots of information we do not need (we are interested in network ports, but this command also displays local UNIX-domain sockets and other non-network connections). Thus we will ‘pipe the netstat output into grep to filter the results a bit. Try:

```
netstat -na | grep -E "tcp|udp"
```

The -E option on grep allows you to use regular expressions. In this case, the expression “tcp udp” will match and display any line containing the string 'tcp' or the string 'udp'.

Question 20: Reference the man page for netstat. How could you formulate the same logic without using grep? What does the -n and -a options do?

Question 21: On what ports is your server currently listening? What application layer protocols are associated with each of these ports?

With the correct parameters, you should get netstat to display a table very similar to the one below.

This output implies our system is running multiple network services. Since we are not currently using these services and a defensible system is minimized, we will shut them down. (Do not uninstall any services; we will be using them in later labs.)

In general, these services are controlled by init scripts running on the Linux distribution. This scripts can be found in either `/etc/init.d`, `/etc/rc.d`, or, as in `Ubuntu` 16.04, `/etc/rcN.d` where the N corresponds to the run level of the service script (for more on run levels, visit `http://www.tldp.org/LDP/sag/html/run-levels-intro.html`). In addition to the various locations, there are also multiple ways to manipulate these scripts. In older distributions, the most common way was to directly manipulate the `rc.conf` file, but this script now calls numerous other scripts so we would have to change multiple files. If you have worked with services in Linux before, you might be familiar with `upstart`, `checkconf`, `systemd` or `systemctl` which are several other ways of modifying these scripts. The point here is that there are several ways to modify services. We will be showing you one method here, but you may need to research or use a different method to do this same task in the future.

Now that we know there are open ports from `netstat`, we now need to see what services are running that are causing these open points. You can see the running services with the command:

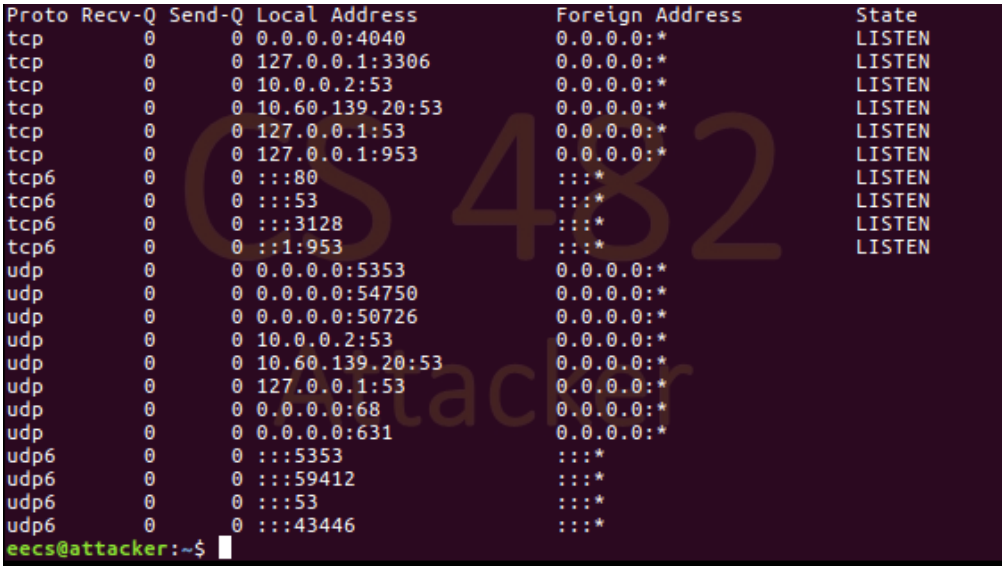

Figure 1: Example output of listening TCP and UDP services with netstat.

```
service --status-all
```

Using grep, we can quickly filter the results to just the running services and see that we have many services like apache2, bind9, and mysql running on our system that we are not currently using. We can stop a single service by issuing the command `sudo service apache2 stop`. If you check, you will not notice we no longer have an active LISTENER on port 80.

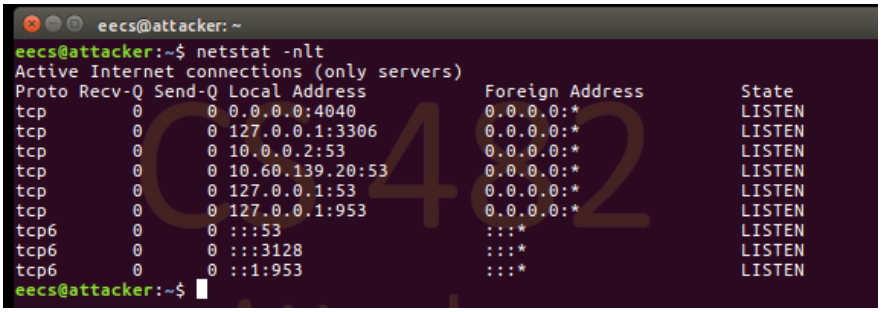

Figure 2: Apache2 no longer running on TCP port 80.

Question 22: Why do you need to use sudo to stop the Apache service?

At this time, reboot your system quickly with the command:

```
sudo shutdown -r now
```

Question 23: After restarting, is Apache2 still running? Why?

The rc.d facility is used to start services when the system boots up. We need to modify these scripts to prevent these services from restarting. We will do this action with the command:

```
sudo update-rc.d -f apache2 remove (Note: the -f option stands for FORCE)
```

Now restart the machine again and check to see if TCP port 80 is open. If you wanted to add Apache2 back into the rc.d scripts so it would automatically start during boot-up, you would use the command:

```
sudo update-rc.d apache2 default (Note: this action uses default run levels)
```

At this time, you need to identify and remove from the start-up scripts the additional services that are opening active ports. You can use the handy "list open files" command, `lsof`, to identify what service is running on a ports since UNIX treats most everything, including network sockets, as a file. To see what is on port 953, try the command:

```
sudo lsof -i:953
```

Question 24: What results for COMMAND and PID did lsof return for port 953?

Using `lsof` and Internet search to look up the process names, associate them with services, and then determine what services are running on all the TCP ports **EXCEPT port 4040 - this is next**. Remember, use netstat to like above to see what services are on what ports.

Question 25: What **four** services did you stop with `service` and remove from startup with `update-rc.d`?

At this point, you should only have one renaming open TCP port on 4040. Let us find out exactly what program is listening on this port by running the following commands:

```
sudo lsof -p <pid>  (Replace <pid> with the process ID you got above.)
ps <pid>
```

This output shows all the open files and file-like objects associated with the process in question. The FD column stands for File Descriptor. The 'cwd' FD is the current working directory of the file and the 'txt' FD is the program text (code and data). The NAME column shows the full path to these fields. The second command, `ps`, is a useful command for investigating processes. You can use `ps aux` will show all the processes running on the machine (`top` is another useful command that shows and updates the top running processes). So, from the `lsof` and `ps` results, we know this process is python code (/usr/bin/python2.7) in the stored in `/lab1/cron` directory.

List the directory contents at the file location with the command:

```
sudo ls -al /lab1/cron/active_listener
```

Question 26: What is this file's permission settings? What does an s in the permissions field indicate? Why is have an 's' with 'root' as owner such a risky security decision?

Well, it looks like this program should not be accessing the network, so lets kill it. Use the kill command:

```
sudo kill <pid>
sudo lsof i:4040
```

The `lsof` command should not return any result since you just terminated the process with the kill command. Wait a minute and then execute `sudo lsof i:4040` again. You should see that another instance of that program is back listening on port 4040 again. Something is restarting the program. Interact with the process using `nc` with the command:

```
nc 127.0.0.1 4040
```

We need to check the system scheduling daemon, cron, to see if a 'cron job' is restarting the program which is listening on port 4040. The settings for cron job are contained file known as a 'crontab'. You can view and modify these files with the crontab command. Remember, each user has their own set of crontabs. Use 'man' and the information from the 'lsof' command to determine what user is running the program listening on port 4040 and check that user's crontab file.

Question 27: What is the cron daemon? Examine the crontab file. What command is set to run every minute by cron? HINT: There is a systemwide crontab and crontabs per user.

We need to edit the crontab to shut this listening process down permanently. Execute:

```
sudo crontab -u <user> -e
```

Comment-out the line in the crontab that is starting the listener and save the file. Kill the process again with `kill`. Then try looking for the service to restart a few times. You should continue see no results even after the cron job would have otherwise restarted; the process is gone for good now.

8 UNIX System Logs

Logging is an important means of providing monitoring for a defensible system. One of the most important skills in a system administrator is the ability to use the logs effectively to troubleshoot and protect a system. Because many applications log to a single location, it is essential that the administrator be able to find important information quickly and easily. In our case, we will investigate the basics of UNIX logging and the tools needed to find useful information.

The first log we will look at is the default logging location in UNIX, the /var/log directory. You will notice that there are a number of different log files, each with a specific purpose. These logs are fairly uniform among most UNIX systems, although slight variations do exist.

First, let's look at some different commands for viewing log files and any other text file. The first command will let us look at the first 10 lines of the file and the second shows the last 10 lines of the file. Try them:

```
head /var/log/syslog
tail /var/log/syslog
```

Question 28: What is the date, time and message of the first (so the oldest) entry in the syslog (see note below)?

Note: this may be syslog.1 or higher, as logs receive increasing integer identifiers as they roll over and syslog (without a number) is the most current file. As such, tail of syslog (with no integer) is the newest syslog information and head of syslog.x where x is the highest integer is the oldest syslog file.

Since log files on busy servers or servers with higher levels of debugging enabled can become very large, it can be very time consuming and error-prone to search a log file line by line looking for certain events. Instead, filtering out events or textual lines allows you to establish a pattern of events that may be buried in other information. Using the following command, we can filter text files when looking for specific events. Try these commands:

```
grep su: /var/log/auth.log (This will show you whos been using su command.)
grep user NOT in sudoers /var/log/auth.log
```

The second command will show you events when an unauthorized user tried to use sudo. You should see such a failed attempt from the account you created earlier. Log files are also not viewable by viewable by all users to prevent tampering. Switch to the cs482 user (remember password '284sc') and try to view the logs.

Question 29: Examine the oldest syslog file with grep. What is the date and time of the first system reboot by the eecs account? What is the full command you used to find this entry?

Another useful feature of tail is the ability to follow a file. In one window enter:

```
tail -f /var/log/auth.log
```

Then, in another terminal window, use some commands (such as su or sudo) that will be logged. You should see the new entries appear in the log file tail as they are created.

9 Wrap Up

1. Check that your VM is on the network with the correct IP address.
2. Confirm that the ports for the DNS service, Apache web server, and that pesky netcat listener are no longer open.
3. Take a 'snapshot' of the attacker VM and name the snapshot "Lab1 Complete". You can snapshot the image by right-clicking on the machine from the VSphere control panel
4. Shutdown your attacker VM.

10 Turn in requirements

10.1 Partner Submission

Provide one written lab report, answering each question properly labelled with the number and original question, per partner team. Be sure to include the time spent on the lab and document any external resources used.

10.2 Individual Submission

Each member needs to submit a detailed lab reflection. This includes

- approximately one half page that talks about the various themes in the Unix design and how it relates to the security principles discussed in lesson 2-3 and/or access control models (Chapter 4).
- any challenging points or thoughts on what you found interesting during the lab
- time spent you personally spent and how much effort you put forth
- time your partner spent, and how much effort they put forth
- be sure document any external resources used.