Chapter 6: Linux Hardening

In this laboratory exercise, the student will be introduced to the basics of Linux Hardening.

6.1 Laboratory Exercise

Linux Hardening

6.1.1 Specifications

The variant of Linux being used for this laboratory exercise is Ubuntu 16.04.6, an older version of the popular Ubuntu operating system. The machine been configured with numerous common vulnerabilities.

6.1.2 Learning Objectives

- Basics of Linux Hardening
- Basics of Linux Usage

6.1.3 Mapping to NIST Nice Framework

This laboratory exercise is intended to increase the student's familiarity with Linux. Additionally, the student will become familiar with common vulnerabilities and how to harden them. This laboratory exercise maps to the following KSAs from the NIST NICE Framework:

- Cybersecurity and Privacy Principles (K0004)

- Cyber Threats and Vulnerabilities (K0005)
- Basic System and OS Hardening Techniques (K0205)
- Recognizing Types of Vulnerabilities (S0078)
- System, Network, and OS Hardening Techniques (S0121)

6.1.4 Necessary Background and Expected Completion Time

This laboratory exercise can be completed by students with varying background and experience. The following categories should help identify approximately how much time (in minutes) will be necessary to complete the laboratory exercise, for a student meeting the criteria for the respective experience level.

- Beginner: A student in this category has little to no experience with Linux and the Linux terminal.
- Intermediate: A student in this category has experience with Linux and the Linux terminal but has little to no experience with Linux hardening.
- Advanced: A student in this category has experience with Linux and the Linux terminal as well as experience with Linux hardening.

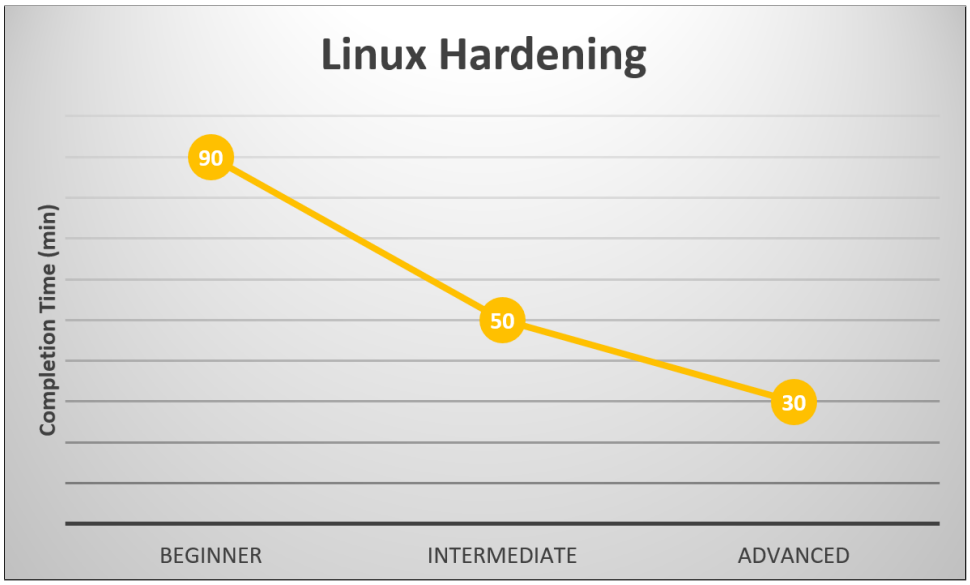

Figure 6.1: *Linux Hardening Laboratory Exercise Expected Completion Time (min)*

6.1.5 Configuration and Setup

The machine used in this laboratory is an installation of Ubuntu 16.04. It will be configured with the vulnerabilities listed in the vulnerability overview. This machine is configured with additional users, automatic updates disabled, weak passwords, erroneous cronjobs, and unprotected SSH enabled. The machine is configured using the initialization script, listing **6.1**

Listing 6.1: *lh-initializationscript.sh*

```
#!/bin/bash

#overhead
sudo dpkg --configure -a
sudo apt-get install git -y
sudo apt-get install openssh-server openssh-client -y
sudo service ssh start
sudo rm -r CYOTEE
sudo git clone https://github.com/CenterForSecureAndDependableSystems/CYOTEE.git

#create users
useradd redteam
useradd guest

#assign passwords to the users
sudo echo -e "redteam\nredteam" | passwd redteam
sudo echo -e "guest\nguest" | passwd guest

#disable auto-updates
sudo rm /etc/apt/apt.conf.d/20-auto-upgrades
sudo rm /etc/apt/apt.conf.d/20auto-upgrades
sudo cp CYOTEE/CYOTEE_Code_Linux/20-auto-upgrades /etc/apt/apt.conf.d/20-auto-
    upgrades

#add a couple of cron jobs

sudo crontab -u thesis -l | { cat; echo "* * * * * touch ~/Desktop/sensitivefile"
    ; } | crontab - -u thesis
sudo crontab -u thesis -l | { cat; echo "*/2 * * * * rm ~/Desktop/sensitivefile";
     } | crontab - -u thesis
```

6.1.6 Vulnerability List

1. Default, Weak, or Common Password
2. Additional Accounts
3. Disabled Automatic Updates
4. SSH
5. Cronjobs

6.1.7 Challenges

1. *Change Password*

   One of the most common vulnerabilities seen in machines and devices is password security. Many devices do not come preconfigured with a password, while others have a default

password. The first thing one should ensure is that their machine or device is password protected, otherwise it would be similar to not placing a lock on your front door, leaving yourself vulnerable to anyone. Once you have ensured you have a password on your machine, then consider: is it a secure password or not? Generally, a password should be changed from the default. Many individuals believe that because their device or machine has a default password, it is secure, however this is not correct. Default passwords are often applied to all devices from a similar batch and are frequently code-like PINs such as 0000 or 1234. For this reason, an individual should ensure that the default password is changed, otherwise it is as though you have bought a lock for your front door but everyone in the neighborhood has the same key which unlocks your door. Changing the default password does that mean your device is secure because your password may be on the list of common passwords. Each year a list of commonly used passwords is published by various organizations on the Internet. Common passwords include the word "password", reusing the username as the password, an empty password, and many others. An individual should consult these lists of common passwords to ensure that they have not accidentally and unknowingly used one of them. Once you have assigned a password, changed the default, and checked that your password is not on a list of commons ones, your password still may not be considered secure. Passwords also vary in what is known as their "strength", or how challenging it would be to crack. Common recommendations for ensuring high strength passwords include having a long length password, varying the characters in the password (uppercase, lowercase, special characters, numbers, etc), and avoiding a password containing personal information such as your pet's name. The default password on this machine is password.

For this task, create a high strength password and reset the current weak password to the new one.

2. *Remove/Disable Unnecessary Accounts*

When one uses a personal computers at home, they often only has one account on the machine: the one they set up themselves. Multi-user machines are a commonplace and often found in shared spaces such as schools, libraries, and cyber-cafés. For example, a

home computer may have two accounts, one for the parents and one for the children, with the childrens' account having less privilege and access than the adults' account. When acquiring a machine, or configuring a new machine, one should always check what user accounts are on that machine. While additional accounts are not inherently malicious, they may be providing an insecure backdoor into the machine. For this reason, one should ensure that unnecessary accounts are removed. If one is unsure whether the additional account is necessary or not, they can first disable or deescalate the privileges it has.

For this task, find the additional accounts that exist. First disable them, then delete/remove them.

3. *Enable Automatic Update Alerts*

   Updates are a very important element of computing. Computers and the software they run are often vulnerable, but updates provide patches for these vulnerabilities, so they can no longer be exploited. Unless update sites are checked daily to know when a new update is available, a critical update, providing a patch for a serious vulnerability, may accidentally be missed. New vulnerabilities are discovered frequently with exploits developed shortly thereafter. Manually checking for updates can result in missed updates which can leave the user susceptible to exploits. For this reason, it is important to have automatic updates enabled and to update regularly.

   For this task, enable the automatic updates on the machine.

4. *Disable or Harden SSH*

   Secure shell, better known as SSH, is a protocol which allows a user to log into a remote machine via a terminal. This can be useful for individuals who work remotely, or need to host a machine remotely. While this service can be useful, it also poses a security risk. Allowing remote access to your machine means that if a malicious user can bypass the password protection, that user now have full access to your machine. When it is not practical to disable SSH, one can improve the security of SSH by using SSH keys for authentication rather than a password.

   For this task, log into the vulnerable machine using SSH. Next, disable the

SSH service on the vulnerable Linux machine. Lastly, restart the SSH service and harden the service using SSH keys.

5. *Remove Unnecessary Cronjobs*

   Cron is a software utility which allows a user to schedule tasks to be performed at specified intervals. This is useful for tasks such as backing up a machine or checking for updates, among other tasks. Users can edit a Crontab file, which is a file containing the tasks which need to be run. These tasks are more commonly referred to as "Cronjobs". The Crontab file uses a specific format, allowing the user to specify the interval over which the Cronjob should be run using the metrics: minutes, hours, day of month, month of year, and day of week. The format for a Cronjob in the Crontab file is as follows:

   ```
   Min.  Hr.  DayOfMonth Month DayOfWeek Command
   ```

   Using an asterisk (*) in place of any of the fields in the format means that any value for that field will be accepted. For example, if you wanted to run the command `ls` 30 minutes after each hour, your formatted Cronjob would be:

   ```
   30 * * * * ls
   ```

   Other symbols, such as the step value symbol (/) can be used to provide further control. For example, to create a backup of your history every thirty minutes, your formatted Cronjob would be:

   ```
   */30 * * * * history > history.txt
   ```

   Cronjobs can be powerful but they can also be a covert way to perform some malicious action at fixed intervals, such as sending collected keystrokes to a remote machine. For this reason, one should regularly check their Crontab file to ensure that no unintended Cronjobs have been added.

   For this task, identify the unnecessary Cronjob running on this machine and remove the job from the Crontab file.