SCENARIO #4: **Website Fingerprinting Scenario**

WARNING:

Warning: Any use of penetration testing techniques on a live network could result in expulsion and/or criminal prosecution. Techniques are to be used in lab environments, for educational use only or on networks for which you have explicit permission to test its defenses.

Level:

☐Beginner

☒Intermediate

☐Advanced

Time Required: 120 minutes

Audience: ☒Instructor-led ☐Self-taught

Scenario Learning Outcomes: Upon completion of this scenario, students will be able to:

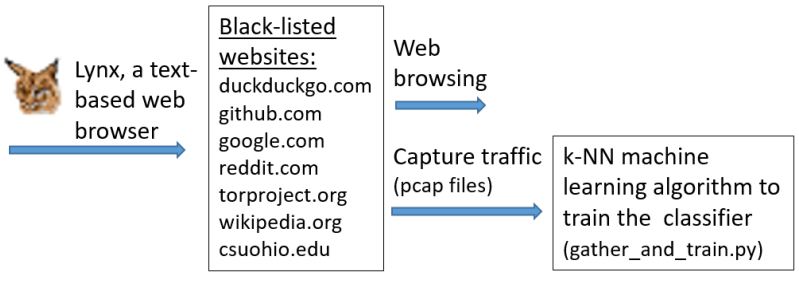
Demonstrate the execution of Website Fingerprinting attacks using Kali Linux

Materials List:

- Computers with Internet connection
- Browsers: Firefox (preferred), Google Chrome, or Internet Explorer
- Python version 3.6 or higher
- "Intro to Ethical Hacking" lab environment

Introduction

During this lab session, we will perform a website fingerprinting attack using the k-Nearest Neighbor (k-NN) algorithm. Website fingerprinting is a method used to identify users accessing a black-listed website by analyzing the distinctive web traffic patterns associated with that website. The k-NN algorithm is a machine learning technique primarily designed for classification tasks, with 'k' representing the number of nearest neighbors based on the chosen distance metric.



We will capture traffic data from seven websites using Lynx, a text-based web browser. The collected data will be stored in pcap files, which will then be utilized to train the classifier. This trained classifier will subsequently be employed to determine if a user is accessing a black-listed website.

Systems and Tools Used:

- Kali Linux through the OCRI virtual machine platform (u: root, p: toor)
- Power down all other systems

Environment Setup:

Please refer to the OCRI Sandbox Setup document. In this lab, we will be using Kali Linux machine instances.

Environment Setup Verification: Before starting this pen test make sure the environment setup is done and deployment is successful.

- Verify that the remote connection to Kali Linux machine is successful and the machine is connected to internet without any error. You can confirm the status of the internet connection by launching the Firefox web browser and attempting to access a website, such as https://csuohio.edu Successful access to the website will indicate that the internet connection is functioning correctly, and the deployment has been successful. If there is an error in the Internet Connection that means the deployment is not proper.
- Do not start the pen test if deployment is not successful. Delete the deployment and redeploy it again.

Steps to execute "Website Finger Printing" in the Sandbox:

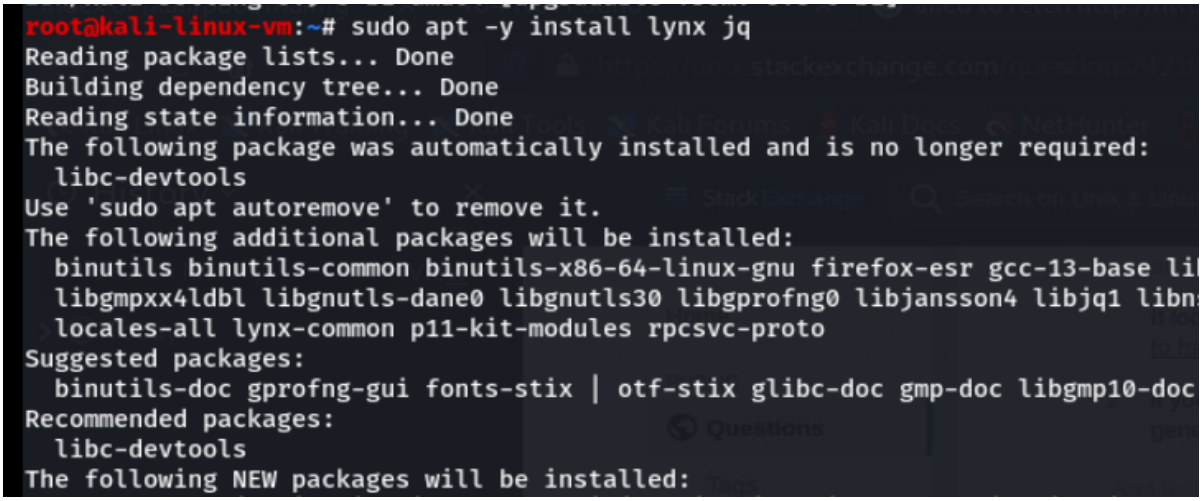
In this scenario, we will use the Kali Linux machine. This scenario is executed in 5 steps. Starting with **Part One**, we download the necessary software, including the Lynx browser and essential Python modules. **Part Two** is for you to familiarize with network data capture (capture.sh) while the Lynx browser accesses a specific website. It is expanded in **Part Three** to capture network traffic over multiple websites (mass-capture.sh). This script automates the capture process for multiple domains listed in the config.json file. The aim is to generate diverse and abundant data for training a robust classifier, which is performed in **Part Four**. Finally, **Part Five** involves testing the trained classifier.

At the end of each step, tasks are present. Please complete the tasks.

Part One: Download the Required Software

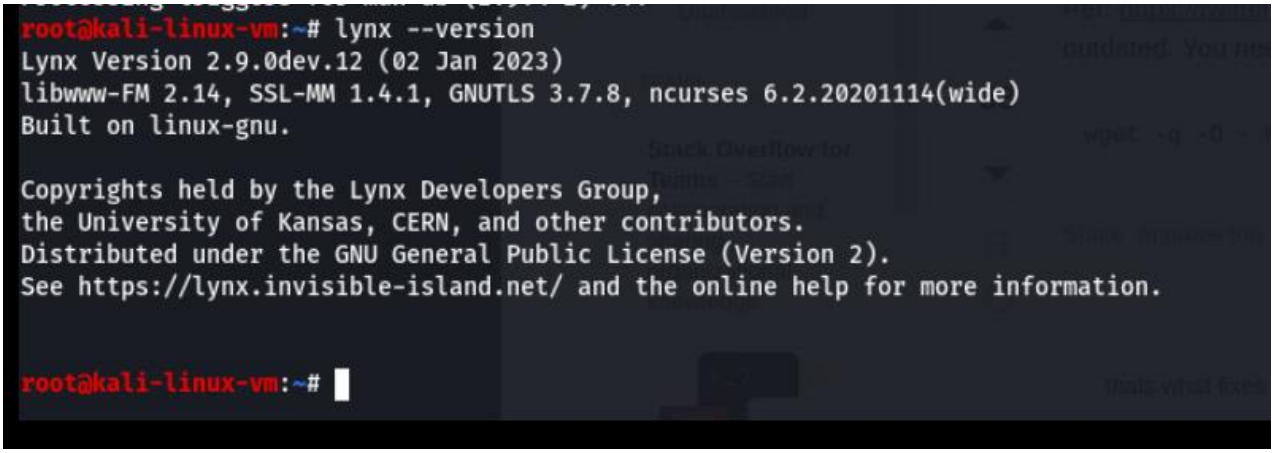
1. Login to Kali Linux machine (username: root; password: toor)

2. Go to activities and open a new terminal and enter the following command to install the lynx browser and jq. (Note that lynx is a text-based web browser and jq is a lightweight and flexible command-line JSON processor.)
   sudo apt -y install lynx jq



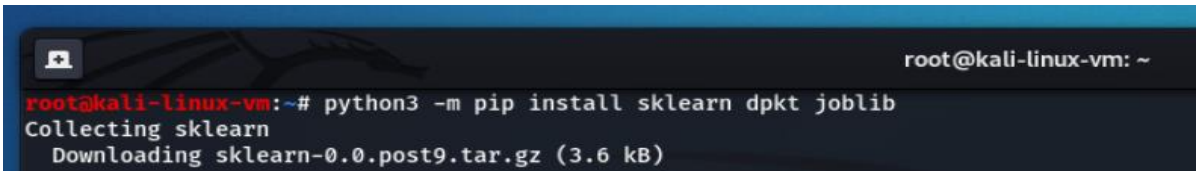
3. Once the installation is complete, please confirm the successful installation by executing the following command. This command will display the version of the installed Lynx browser.

   lynx --version

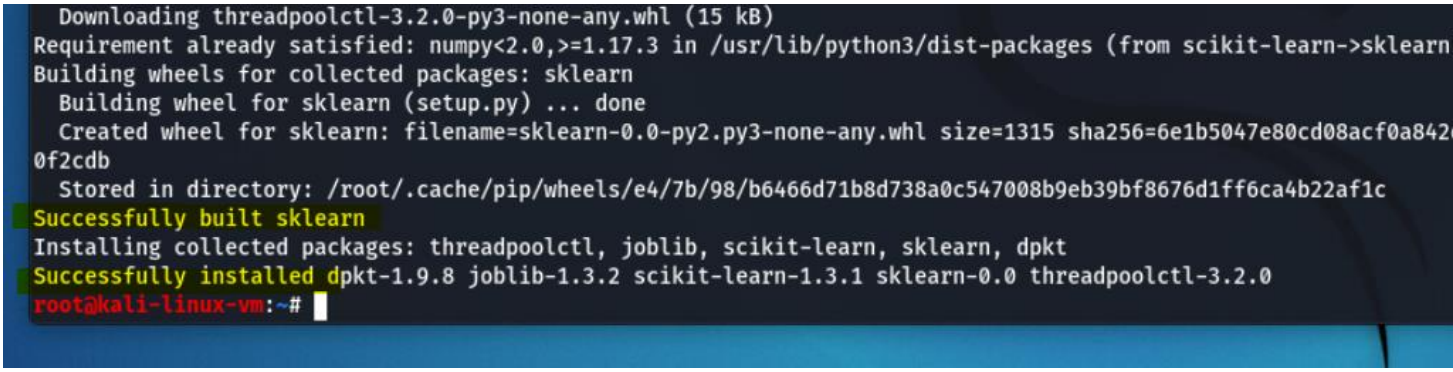


4. Install the necessary python3 modules using the following command. After the successful installation, we will see “Successfully installed” message as shown below. We just show the top and bottom portion of the screen due to the space limit.

   python3 -m pip install sklearn dpkt joblib



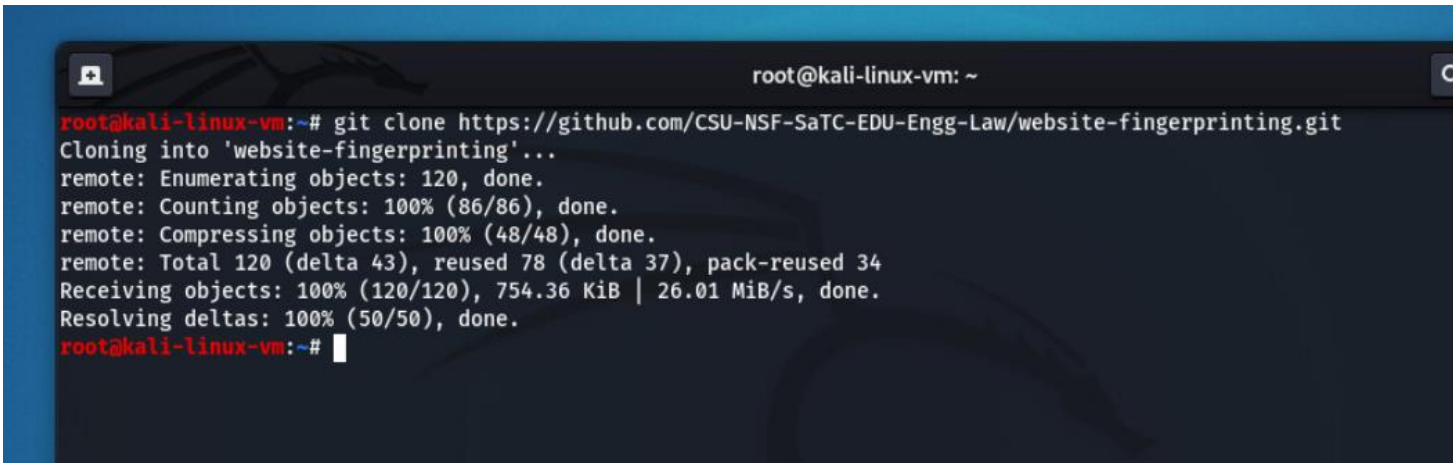
●●●



   Note that sklearn is a python module for machine learning library, featuring various classification, regression and clustering algorithms. dpkt is a pythin module for analyzing TCP/IP packets. joblib helps executing the python modules faster.

5. Clone the website fingerprinting code from GitHub using the following command.

   git clone https://github.com/CSU-NSF-SaTC-EDU-Engg-Law/website-fingerprinting.git



Tasks:

1. Please provide screenshot of successful installation of lynx browser in step 3.

Part Two: Getting Familiar with Network Traffic Capturing (single website)

We will capture network traffic data (step 7) when the lynx browser accesses github.com (step 8).

6. Navigate to the directory where the website fingerprinting modules are present.

   cd website-fingerprinting

7. Execute the capture.sh script, where the command line command specifies the folder name for storing the collected data, which is "./pcaps/github.com" in our case. We chose this folder name for our convenience because the lynx browser accesses github.com website in step 8.

   ./capture.sh github.com lynx

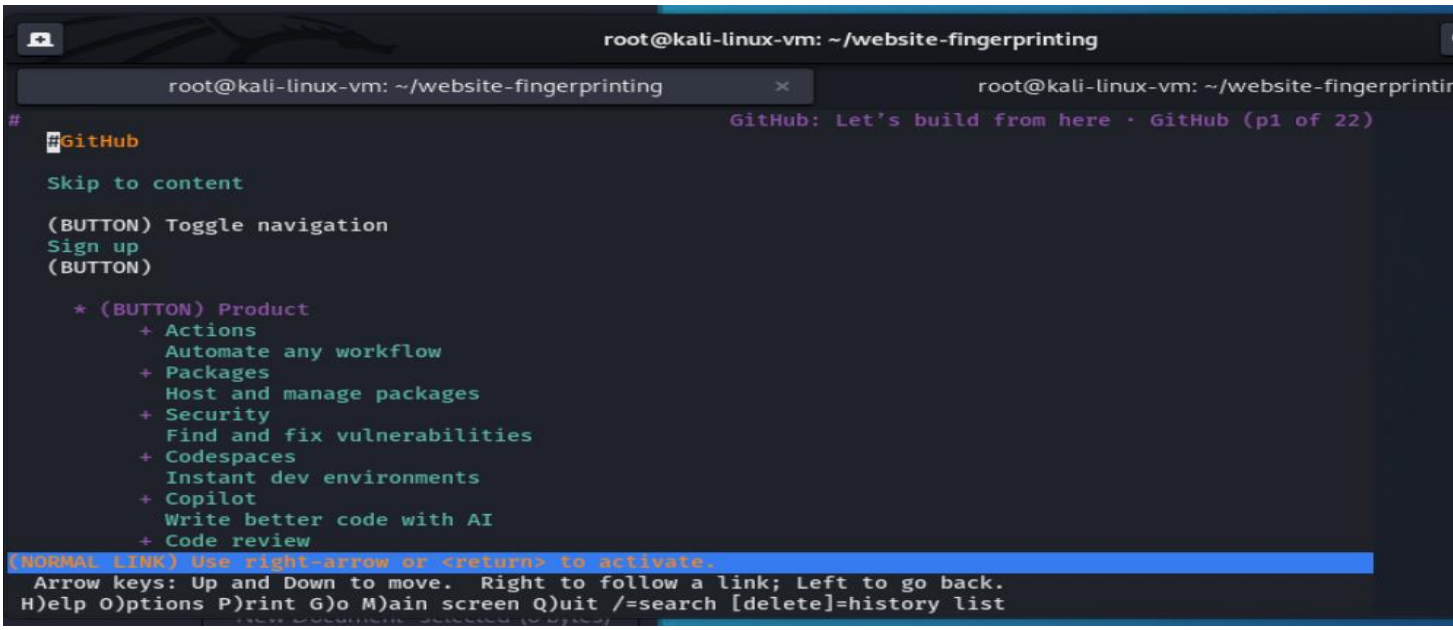
```
root@kali-linux-vm:~/website-fingerprinting# ./capture.sh github.com lynx
mkdir: created directory './pcaps'
mkdir: created directory './pcaps/github.com'
PCAPs in ./pcaps/github.com:  0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
Got 0
```

8. Open a new terminal. Use lynx to access "github.com". (Terminal #2)

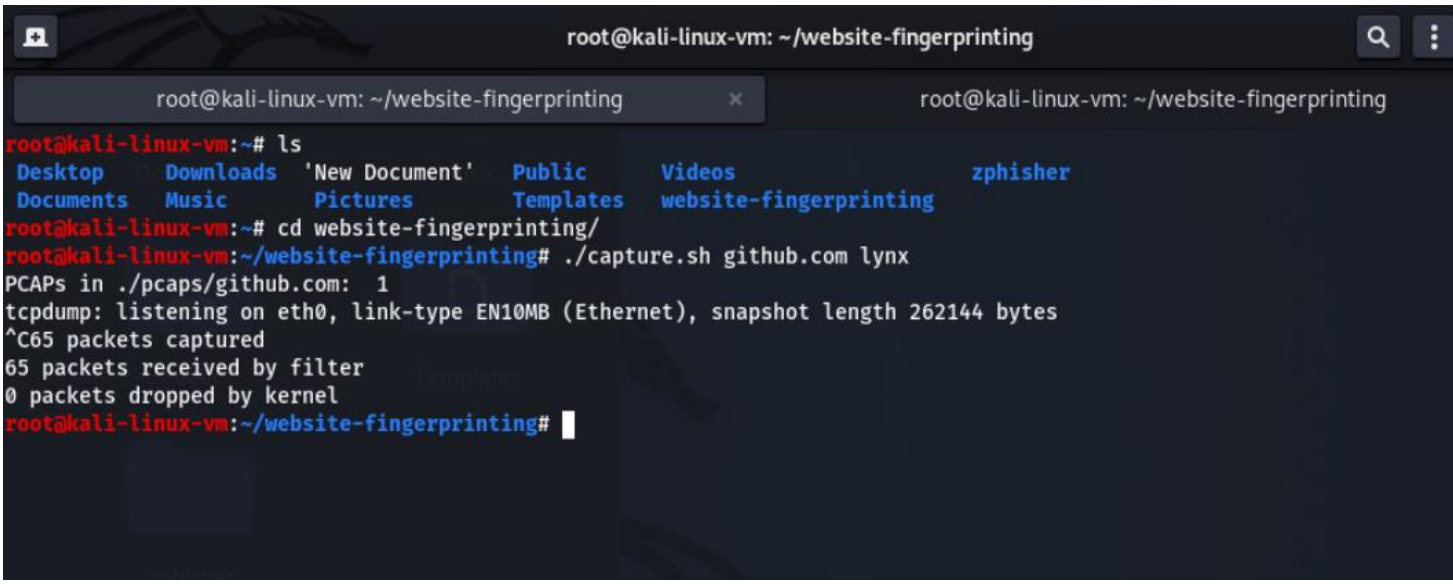
   lynx -accept_all_cookies https://github.com

```
root@kali-linux-vm:~/website-fingerprinting# lynx -accept_all_cookies https://github.com
```

9. We can see the "Github" website is opened in the Lynx browser. (Terminal #2)



10. In the first terminal we can see that the network traffic is being captured for “github.com” website. Once the website has finished loading, quit the capture.sh script using Ctrl+c (press both Ctrl and c keys on the keyboard) and close the lynx browser by pressing “q” (lynx command for quit). (Terminal #1)



Warning: The folder name we utilized for storing network traffic data when accessing the "github.com" website is "github.com." While it may seem confusing, this designation was chosen for organizational purposes.

Tasks:

2. Please provide the screenshot of the domain opened in lynx browser at step 9.

3. Please provide the screenshot of the network traffic captured at step 10.

Part Three: Capturing Network Traffic Data Over Multiple Websites

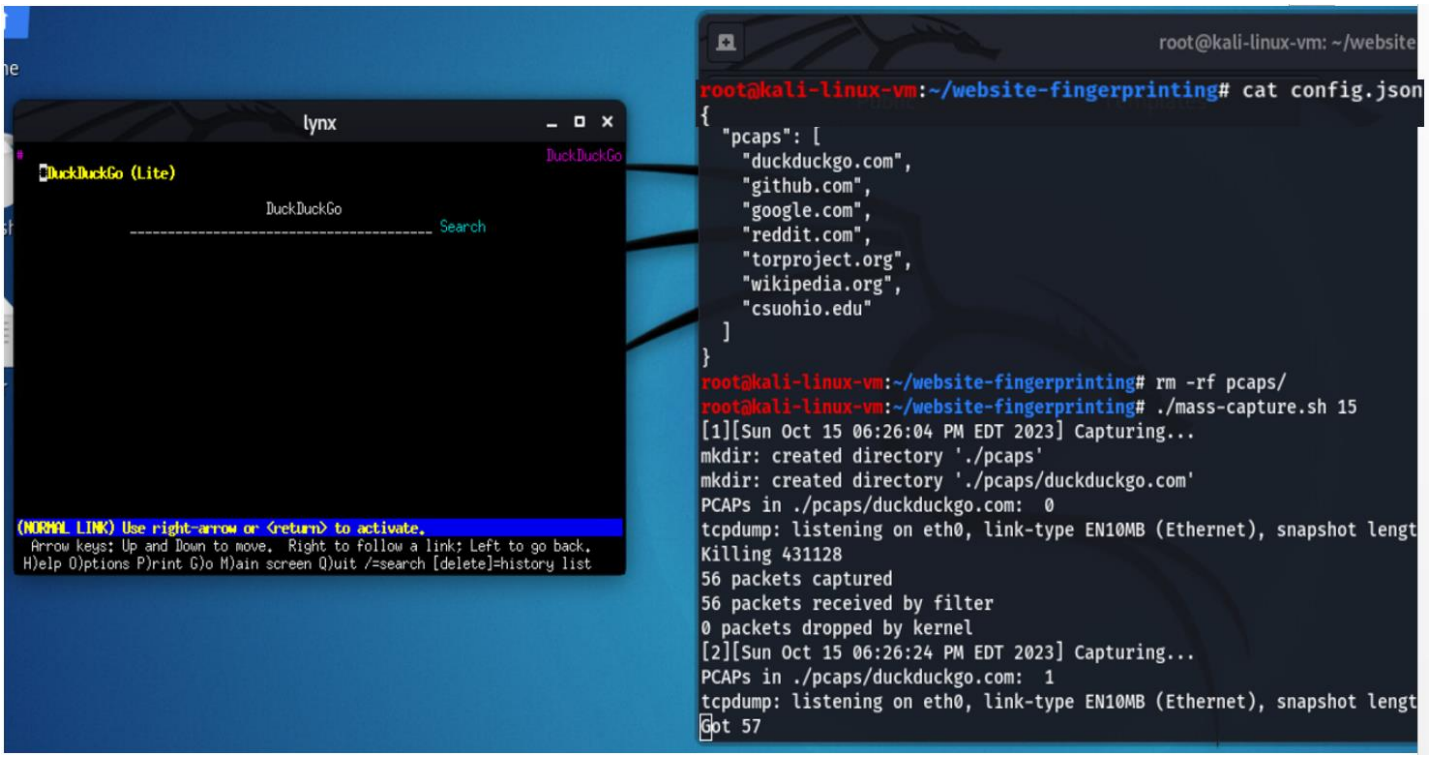
The process of capturing a website in Part Two is tedious especially when considering that we would like to capture many "black-listed" websites. In this part, we will use the mass-capture.sh script to automate the traffic capture for all the websites listed in the config.json file. This is similar to step 7 (capture.sh) and step 8 (lynx browser) in Part Two over multiple websites.

11. Close the two terminals and open a new one (Terminal #1). Navigate to the directory where the website fingerprinting modules are present as in step 6. Display the config.json file using the "cat" command, which shows the seven websites to be accessed. Next, clear the existing files in the pcap folder using the "rm" command (remove). Please refer to the screen shot in step 12.

    cd website-fingerprinting
    cat config.json
    rm -rf pcaps/

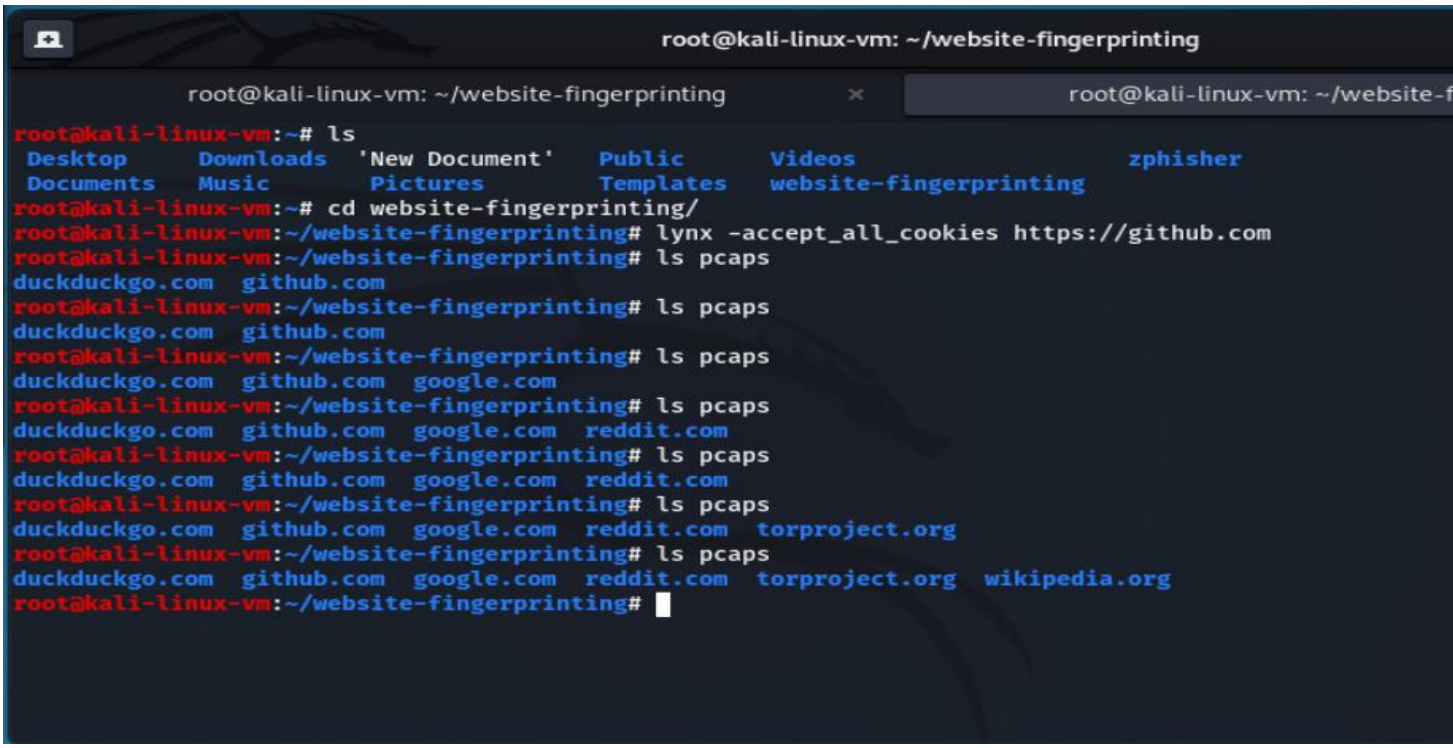
12. Execute mass-capture.sh. It will capture traffic and place all captures into the pcaps folder. It takes one argument, the amount of captures we will use per each website. In our scenario, we will do 15 captures per website to make sure our classifier is accurate (see the terminal on the right below; Terminal #1). For each website, a new instance of the terminal will be launched for lynx to browse the website (see the terminal on the left below; Terminal #2) – this allows you to watch and verify that the script is cycling through each website.
    Note that each capture takes max 20 seconds. This means max 300 seconds or 5 minutes per each website because it takes 15 captures. Altogether, it takes max 35 minutes because we access 7 websites.

    ./mass-capture.sh 15



13. Open a new terminal in order to check the progress. Execute the following command to view the websites being fingerprinted. We can keep executing the same command every few seconds until all seven directories are created, as shown below. Please allow an additional 5 minutes for the last website to be accessed and the corresponding traffic data is captured. (Terminal #3)

ls pcaps



Warning: The folder names we utilized for storing network traffic data when accessing the "duckduckgo.com", "github.com", etc. are "duckduckgo.com", "github.com", etc. While it may seem confusing, this designation was chosen for organizational purposes.

Tasks:

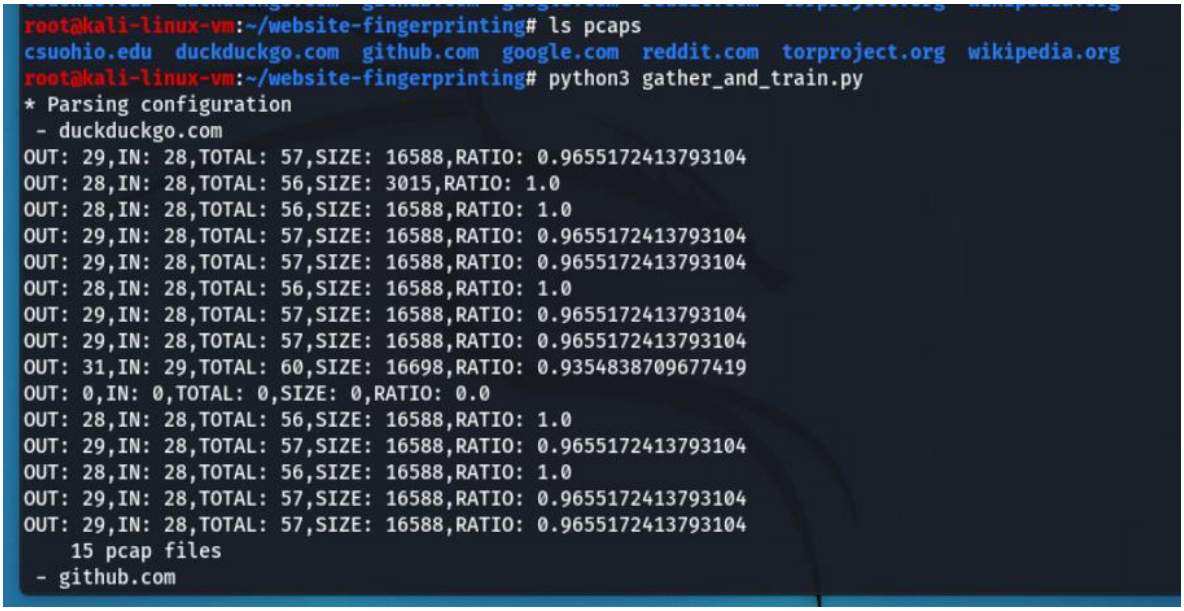
4. Please provide the screenshot of any website being captured as in step 12 (screen on the right).

5. Please provide the screenshot of all the websites being captured at step 13.

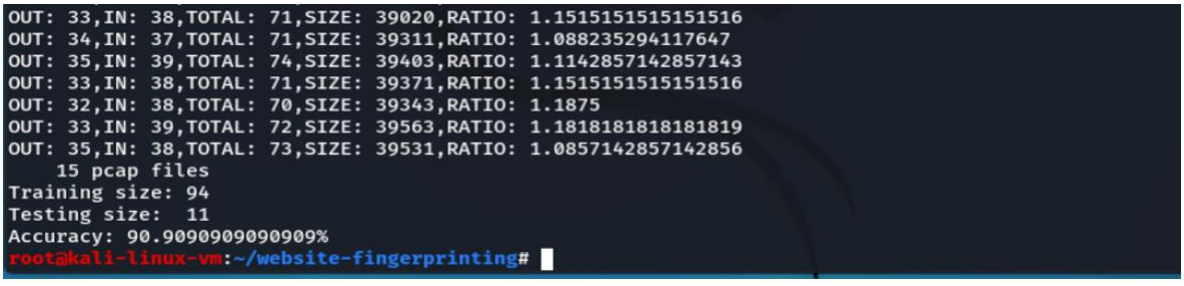
Part Four: Training the Website Fingerprinting Classifier

14. This step involves using the gather_and_train.py script. The script will analyze the .pcap files belonging to each website and train the classifier using the k-NN (k-Nearest Neighbors) machine learning tool. Execute the following command.
    Note the accuracy reported at the bottom of the screen below. In order to obtain the accuracy, 70% of captures were used for training while the remaining 30% were used for accuracy testing.

    python3 gather_and_train.py



•••



Tasks:

6. Please provide the screenshot that shows the accuracy of the trainer which is obtained at step 14.

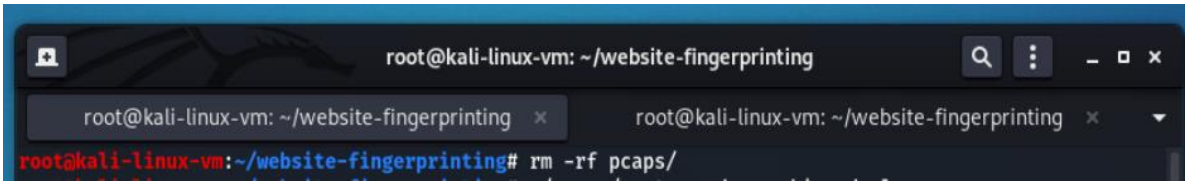
7. What does it mean by the prediction accuracy of 90.9%?

Part Five: Testing the Classifier

Consider that a user accesses a certain website and we want to identify if he/she accesses one of the seven "black-listed" websites using the classifier we developed in Part Four. Here, we collect the traffic data (Terminal #1) while the user accesses "csuohio.edu" (Terminal #2), which are similar to step 7 and step 8, respectively. Now, we will use the classifier to identify which website the user has accessed.

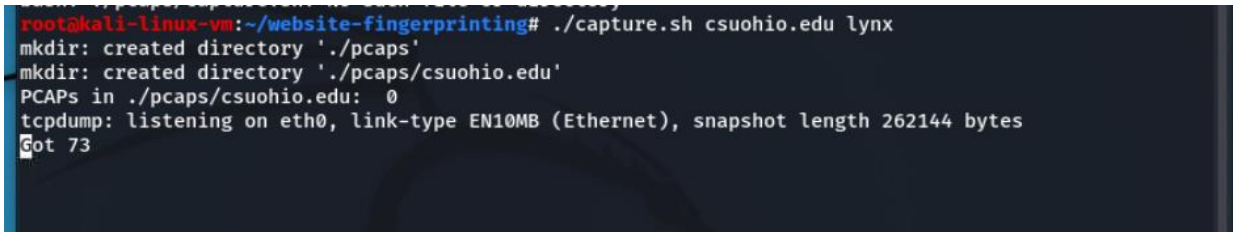
15. Remove all existing .pcap files using the following command. (Terminal #1)

    rm -rf pcaps/



16. Execute the following command to create a new capture for the domain of our choice: "csuohio.edu". (Terminal #1)

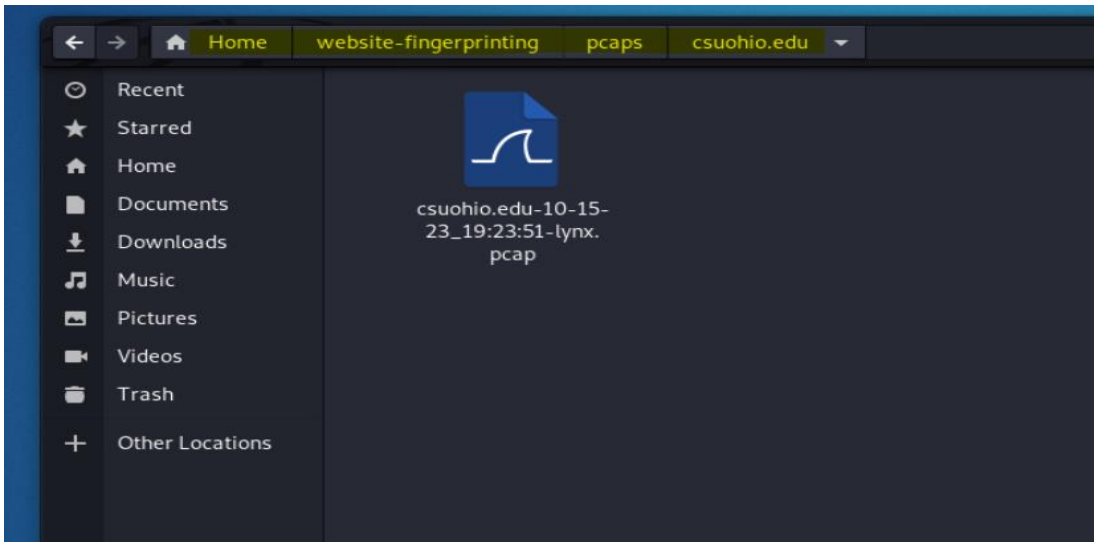
    ./capture.sh csuohio.edu lynx



17. Execute the following command in a different terminal. (Terminal #2)

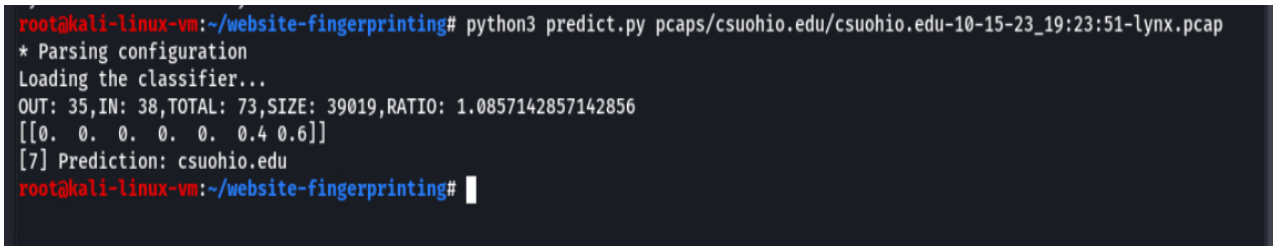
    lynx -accept_all_cookies https://csuohio.edu

18. We can see that the pcap file is created in the "csuohio.edu" folder as shown below.



19. Now we run the classifier against the pcap file created in steps 16-18 to know if the user accesses one of the seven black-listed websites.

python3 predict.py pcaps/csuohio.edu/<new .pcap filename>



20. From the above screenshot, we can see that the classifier correctly identifies that the user has accessed the “csuohio.edu” website.

Tasks:

8. Please provide the screenshot of the prediction obtained at step 20.

9. Website contents change over time. If we use the classifier one month later, can it still predict correctly? Justify your answer.

Glossary:

Website Fingerprinting

Website Fingerprinting is a technique used in the field of cybersecurity to identify and track users based on the unique patterns of their web traffic. It involves analyzing the distinctive features and characteristics of a user's interactions with websites to create a recognizable fingerprint or signature. This fingerprint can then be used to distinguish and trace individual users even when other traditional methods of anonymization, such as using VPNs or proxy servers, are employed.

The process typically involves monitoring various aspects of network traffic, including the size and timing of data packets exchanged between the user and websites. By analyzing these patterns, a fingerprinting system can create a profile unique to each user. Website fingerprinting has both security and privacy implications. On one hand, it can be employed for legitimate security purposes, such as identifying and tracking potential threats. On the other hand, it raises concerns about user privacy, especially when used without consent or in contexts where anonymity is expected.

Lynx Browser

Lynx is a text-based web browser designed for use in a command-line interface (CLI). Unlike graphical web browsers that display web pages with images and other multimedia elements, Lynx focuses on providing a straightforward and efficient means of browsing the web using only text.

Key characteristics of the Lynx browser include:

Text-Based Interface: Lynx operates entirely in a text-based environment, making it suitable for use in terminal or command prompt windows. Users navigate through links and access information using keyboard commands.

Accessibility: Lynx is highly accessible, especially for users with visual impairments or those who prefer a simplified browsing experience. It doesn't rely on graphics, making it compatible with screen readers.

Lightweight: Lynx is a lightweight browser, requiring minimal system resources. This makes it suitable for older computers, servers, or systems with limited graphical capabilities.

Speed: Due to its focus on text-only content, Lynx can be faster in loading web pages compared to graphical browsers, especially in situations with slow or limited internet connections.

Scripting Support: Lynx supports scripting languages and can be used in automated processes or scripts to perform tasks such as web scraping or data retrieval.

Cross-Platform: Lynx is cross-platform and can be installed on various operating systems, including Unix-like systems (Linux, BSD), macOS, and Windows.

While Lynx may not provide the rich multimedia experience offered by graphical browsers, it serves specific use cases where simplicity, speed, and accessibility are prioritized. It is commonly used in scenarios where a graphical interface is not available or practical, such as in remote server administration, text-based environments, or certain cybersecurity and research applications.

PCAP files

PCAP files (Packet Capture files) are a standard data format used for storing network traffic captured during packet sniffing or network monitoring. These files contain a detailed record of the data packets exchanged between devices on a network. PCAP files are widely used in cybersecurity, network analysis, and troubleshooting to examine and analyze the flow of data on a network.

Key characteristics of PCAP files include:

Packet Information: PCAP files store information about individual packets of data, including details such as source and destination IP addresses, port numbers, protocols used, packet size, and timestamp information.

Capture Tools Compatibility: PCAP is a common format supported by various packet capture tools, making it easy to exchange and analyze data captured by different applications.

Wireshark Compatibility: Wireshark, a popular network protocol analyzer, is commonly used to open and analyze PCAP files. Wireshark provides a graphical interface for inspecting the contents of captured packets.

Timestamps: Each packet in a PCAP file is timestamped, allowing analysts to understand the timing and sequence of events during network communication.

Platform Independence: PCAP files are platform-independent, meaning they can be generated and analyzed on different operating systems.

Protocol Support: PCAP files capture data at the packet level, providing insights into various network protocols such as TCP, UDP, ICMP, and more.

Uses of PCAP files include:

Network Troubleshooting: Analyzing PCAP files helps diagnose network issues by examining the characteristics of network traffic.

Security Analysis: PCAP files are crucial for identifying and investigating security incidents, including malicious activities and potential threats.

Forensic Analysis: In digital forensics, PCAP files aid in reconstructing and analyzing network activities during a specific period.

Protocol Development: Network developers use PCAP files to test and debug network protocols and applications.

Understanding and interpreting PCAP files is a fundamental skill in network analysis and cybersecurity, as it allows professionals to gain insights into the communication patterns and behaviors within a network.