Name: Amit S Terdal
CSU ID: 2887869

Exercise 1: Website Fingerprinting Scenario

1. Part One – Task 1 Please provide screenshot of successful installation of lynx browser in step 3.

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

2. Part Two – Task 2 Please provide the screenshot of the domain opened in lynx browser at step 9.

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

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

lynx -accept_all_cookies https://github.com

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

3. Part Two – Task 3. Please provide the screenshot of the network traffic captured at step 10.

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.

```
┌──(root㉿kali)-[~/website-fingerprinting]
└─# ./capture.sh github.com lynx
mkdir: created directory './pcaps/github.com'
PCAPs in ./pcaps/github.com:  0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^Zt 43
zsh: suspended  ./capture.sh github.com lynx
```

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

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
```

5. Part three – Task 5. Please provide the screenshot of all the websites being captured at step 13.

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.

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

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.p
```

sandbox02.cech.uc.edu/vcac/Log | CleKali264

sandbox02.cech.uc.edu/vcac/VMRC/Vmrc.aspx?ReadOnly=0&VirtualMachineID=6ca27d21-dbd3-4594-b39e-8b...

CleKali264

Connected to VM

English | SEND CTRL+ALT+DEL | SEND CTRL+C | TOGGLE FULL SCREEN

Press Ctrl-Alt to release the cursor from the guest.

1 2 3 4 | 1:37

(myenv)root@kali: ~/website-fingerprinting

File Actions Edit View Help

(myenv)root@kali: ~/website-fingerprinting | root@kali: ~/website-fingerprinting

```
┌──(myenv)─(root㉿kali)-[~/website-fingerprinting]
└─# python3 gather_and_train.py
/root/website-fingerprinting/gather_and_train.py:27: SyntaxWarning: invalid escape sequence '\.'
  pat = re.compile(".*-curl\.pcap$")
* Parsing configuration
 - duckduckgo.com
OUT: 20,IN: 24,TOTAL: 44,SIZE: 16945,RATIO: 1.2
OUT: 278,IN: 284,TOTAL: 562,SIZE: 174433,RATIO: 1.0215827338129497
OUT: 344,IN: 337,TOTAL: 681,SIZE: 198199,RATIO: 0.9796511627906976
OUT: 350,IN: 331,TOTAL: 681,SIZE: 121454,RATIO: 0.9457142857142857
OUT: 13,IN: 766,TOTAL: 779,SIZE: 190663,RATIO: 58.92307692307692
OUT: 10,IN: 13,TOTAL: 23,SIZE: 9013,RATIO: 1.3
OUT: 15,IN: 14,TOTAL: 29,SIZE: 9079,RATIO: 0.9333333333333333
OUT: 498,IN: 451,TOTAL: 949,SIZE: 176454,RATIO: 0.9056224899598394
OUT: 305,IN: 302,TOTAL: 607,SIZE: 110804,RATIO: 0.9901639344262295
OUT: 27,IN: 28,TOTAL: 55,SIZE: 19679,RATIO: 1.037037037037037
OUT: 488,IN: 487,TOTAL: 975,SIZE: 173760,RATIO: 0.9979508196721312
OUT: 379,IN: 375,TOTAL: 754,SIZE: 152324,RATIO: 0.9894459102902374
OUT: 296,IN: 297,TOTAL: 593,SIZE: 108074,RATIO: 1.0033783783783783
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 375,IN: 356,TOTAL: 731,SIZE: 129537,RATIO: 0.9493333333333334
OUT: 446,IN: 443,TOTAL: 889,SIZE: 155542,RATIO: 0.9932735426008968
OUT: 26,IN: 27,TOTAL: 53,SIZE: 19613,RATIO: 1.0384615384615385
OUT: 29,IN: 28,TOTAL: 57,SIZE: 19679,RATIO: 0.9655172413793104
OUT: 341,IN: 332,TOTAL: 673,SIZE: 122503,RATIO: 0.9736070381231672
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 490,IN: 489,TOTAL: 979,SIZE: 161588,RATIO: 0.9979591836734694
OUT: 27,IN: 28,TOTAL: 55,SIZE: 19679,RATIO: 1.037037037037037
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 250,IN: 236,TOTAL: 486,SIZE: 102731,RATIO: 0.944
OUT: 15,IN: 14,TOTAL: 29,SIZE: 9079,RATIO: 0.9333333333333333
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 388,IN: 389,TOTAL: 777,SIZE: 142592,RATIO: 1.0025773195876289
OUT: 13,IN: 13,TOTAL: 26,SIZE: 9013,RATIO: 1.0
OUT: 23,IN: 26,TOTAL: 49,SIZE: 19547,RATIO: 1.1304347826086956
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
```

sandbox02.cech.uc.edu/vcac/Log | CleKali264

sandbox02.cech.uc.edu/vcac/VMRC/Vmrc.aspx?ReadOnly=0&VirtualMachineID=6ca27d21-dbd3-4594-b39e-8b...

CleKali264

Connected to VM

English | SEND CTRL+ALT+DEL | SEND CTRL+C | TOGGLE FULL SCREEN

Press Ctrl-Alt to release the cursor from the guest.

1 2 3 4 | 1:38

(myenv)root@kali: ~/website-fingerprinting

File Actions Edit View Help

(myenv)root@kali: ~/website-fingerprinting | root@kali: ~/website-fingerprinting

```
OUT: 26,IN: 31,TOTAL: 57,SIZE: 34685,RATIO: 1.1923076923076923
OUT: 25,IN: 33,TOTAL: 58,SIZE: 34819,RATIO: 1.32
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 26,IN: 34,TOTAL: 60,SIZE: 34884,RATIO: 1.3076923076923077
OUT: 26,IN: 31,TOTAL: 57,SIZE: 34685,RATIO: 1.1923076923076923
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 25,IN: 29,TOTAL: 54,SIZE: 34553,RATIO: 1.16
OUT: 25,IN: 30,TOTAL: 55,SIZE: 34622,RATIO: 1.2
OUT: 26,IN: 31,TOTAL: 57,SIZE: 34687,RATIO: 1.1923076923076923
OUT: 25,IN: 33,TOTAL: 58,SIZE: 34817,RATIO: 1.32
    22 pcap files
 - csuohio.edu
OUT: 15,IN: 22,TOTAL: 37,SIZE: 32242,RATIO: 1.4666666666666666
OUT: 14,IN: 23,TOTAL: 37,SIZE: 32286,RATIO: 1.6428571428571428
OUT: 15,IN: 19,TOTAL: 34,SIZE: 32022,RATIO: 1.2666666666666666
OUT: 12,IN: 18,TOTAL: 30,SIZE: 31978,RATIO: 1.5
OUT: 13,IN: 20,TOTAL: 33,SIZE: 32110,RATIO: 1.5384615384615385
OUT: 14,IN: 22,TOTAL: 36,SIZE: 32242,RATIO: 1.5714285714285714
OUT: 15,IN: 22,TOTAL: 37,SIZE: 32220,RATIO: 1.4666666666666666
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 15,IN: 23,TOTAL: 38,SIZE: 32286,RATIO: 1.5333333333333334
OUT: 13,IN: 19,TOTAL: 32,SIZE: 32044,RATIO: 1.4615384615384615
OUT: 15,IN: 16,TOTAL: 31,SIZE: 31846,RATIO: 1.0666666666666667
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 18,IN: 24,TOTAL: 42,SIZE: 32352,RATIO: 1.3333333333333333
OUT: 13,IN: 20,TOTAL: 33,SIZE: 32110,RATIO: 1.5384615384615385
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 16,IN: 23,TOTAL: 39,SIZE: 32286,RATIO: 1.4375
OUT: 14,IN: 19,TOTAL: 33,SIZE: 32044,RATIO: 1.3571428571428572
OUT: 16,IN: 22,TOTAL: 38,SIZE: 32220,RATIO: 1.375
    19 pcap files
Training size: 135
Testing size:  16
Accuracy: 75.0%

┌──(myenv)─(root㉿kali)-[~/website-fingerprinting]
└─#
```

7. Part Four - Task 7. What does it mean by the prediction accuracy of 90.9%?

The gather_and_train.py script is used to train a classifier. This classifier's job is to identify which of the seven websites a user is visiting by analyzing the network traffic generated by that visit.

To test how well the classifier does its job, the data collected is split into two parts:

- **Training data (70%):** This portion is used to "teach" the classifier what the network traffic patterns of each website look like.
- **Testing data (30%):** This is a separate set of data that the classifier has never seen before. It's used to evaluate how accurately the classifier can identify the websites based on the traffic.

So, a prediction accuracy of 90.9% means that when the classifier was given the testing data, it correctly identified the website being visited in 90.9% of the cases.

8. Part Five - Task 8. Please provide the screenshot of the prediction obtained at step 20.

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.

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

Website content changes over time, and this can impact the accuracy of the website fingerprinting classifier

- **Website Fingerprinting Basics**: Website fingerprinting relies on analyzing the unique patterns of network traffic associated with a website. These patterns include the size and timing of data packets.
- **Content Dynamics**: Websites frequently change their content, layout, and functionality. This results in alterations to the underlying network traffic patterns. For example, a website might add new images, scripts, or interactive elements, all of which affect the size and timing of data packets.
- **Classifier Training**: The classifier is trained on a specific set of network traffic data captured at a particular time. If the website changes significantly, the traffic patterns used to train the classifier may no longer accurately represent the website's current traffic.
- **Accuracy Degradation**: As a result, the classifier's accuracy in correctly identifying the website can decrease over time. The classifier might misidentify the website because the current traffic patterns deviate from the patterns it was trained on.

In conclusion, while the classifier might still have some level of accuracy, its ability to predict correctly can be compromised a month later due to changes in website content and the corresponding changes in network traffic patterns.