

LESSON TITLE: **Lab – 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

Time Required: 60 minutes

☐ Advanced

Audience: ☒ Instructor-led

☐ Self-taught

Lesson Learning Outcomes: Upon completion of this lesson, 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

In this lab, we will be performing website fingerprinting using a k-NN attack on Kali Linux.

Systems and Tools Used:

- Kali Linux (*u: root, p: toor*)
- **Power down all other systems**

MAKE SURE YOUR KALI LINUX IS UPDATE TO LATEST VERSION
INORDER TO UPDATE KALI LINUX PLEASE FOLLOW THE STEPS IN TERMINAL
RUN THIS COMMAND IN TERMINAL >

gedit /etc/apt/sources.list

COPY 4 lines BELOW and delete everything which exist in previous file.

See <https://www.kali.org/docs/general-use/kali-linux-sources-list-repositories/>

deb <http://http.kali.org/kali> kali-rolling main contrib non-free

Additional line for source packages

deb-src <http://http.kali.org/kali> kali-rolling main contrib non-free

MAKE SURE TO SAVE IT AFTER PASTING IT

Update command

RUN THIS COMMAND IN TERMINAL >

wget -q -O - <https://archive.kali.org/archive-key.asc> | apt-key add

RUN THIS COMMAND IN TERMINAL >

sudo apt update

RUN THIS COMMAND IN TERMINAL >

sudo apt full-upgrade y

Website Fingerprinting is the process of monitoring encrypted network traffic to identify a website based on characteristics of its packet transfer sequence. Common characteristics used in website fingerprinting are packet ordering, packet sizes in each direction, total bandwidth used, and inter-packet timings. Once a website has been fingerprinted, an attacker can reliably identify which website a user visits by passively monitoring their network traffic.

Module Activity Description:

Part Zero: Download Required Software

- 1. Enter the following command into the Linux terminal to install the lynx browser and jq**
`sudo apt -y install lynx jq`
- 2. Enter the following command to install the necessary python3 modules**
`python3 -m pip install sklearn dpkt joblib`
- 3. Install the website fingerprinting code from github using the following command**
`git clone https://github.com/CSU-NSF-SaTC-EDU-Engg-Law/website-fingerprinting.git`

Description of provided code:

`./capture.sh` – shell script used to monitor and capture network traffic using tcpdump
`./mass-capture.sh` – shell script used to capture traffic of all domains in the config.json file.
`./gather_and_train.py` – python script used to train the classifier on the gathered .pcap files
`./predict.py` – python script used to predict the domain of a provided .pcap file

```

root@kali-linux-vm:~# git clone https://github.com/CSU-NSF-SaTC-EDU-Engg-Law/website-fingerprinting.git && cd website-fingerprinting
Cloning into 'website-fingerprinting'...
remote: Enumerating objects: 114, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 114 (delta 40), reused 13 (delta 7), pack-reused 34
Receiving objects: 100% (114/114), 121.08 KiB | 3.03 MiB/s, done.
Resolving deltas: 100% (47/47), done.
root@kali-linux-vm:~/website-fingerprinting# ls
capture.sh  CODE_OF_CONDUCT.md  CONTRIBUTING.md  LICENSE  packet.py  README.md
classifier  config.json          gather_and_train.py  mass-capture.sh  predict.py  utils.py
root@kali-linux-vm:~/website-fingerprinting#

```

Part One: Testing Network Monitoring

Next we will want to test the capture.sh script using lynx and one of the domains listed in config.json. We can see that capture.sh takes 2 arguments -- the domain being fingerprinted and the source. A fingerprinted website's .pcap files will be under the directory ./pcaps/<domain>

1. Open two separate terminals
2. In the first terminal, execute the command below (you will want to be in the website-fingerprinting directory).
./pcaps/capture.sh <domain> lynx
3. In the second terminal, use lynx to open one of the domains in the config.json file.
lynx -accept_all_cookies https://<domain>
4. Once the website has finished loading, quit the capture.sh script using ctrl+c and close lynx using qq.

Capturing Network Traffic

```

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 60

```

Launching Lynx

```

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

```

GitHub on Lynx

```
#                               GitHub: Where the world builds software · GitHub (p1 of 27)
#GitHub

Skip to content
GitHub no longer supports this web browser. Learn more about the browsers we support.

Sign up (BUTTON)
(BUTTON)

* Product
  + Features
  + Mobile
  + Actions
  + Codespaces
  + Packages
  + Security
  + Code review
  + Issues
  + Integrations
  + GitHub Sponsors
  + Customer stories
(NORMAL LINK) Use right-arrow or <return> to activate.
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

The current process of capturing each domain individually is tedious especially when considering that we would like to have more than one .pcap file per website to use as training data. In the next section we will use the mass-capture.sh script to automate the traffic capture for each domain in the config.json file.

Part Two: Optimizing Network Traffic Capture

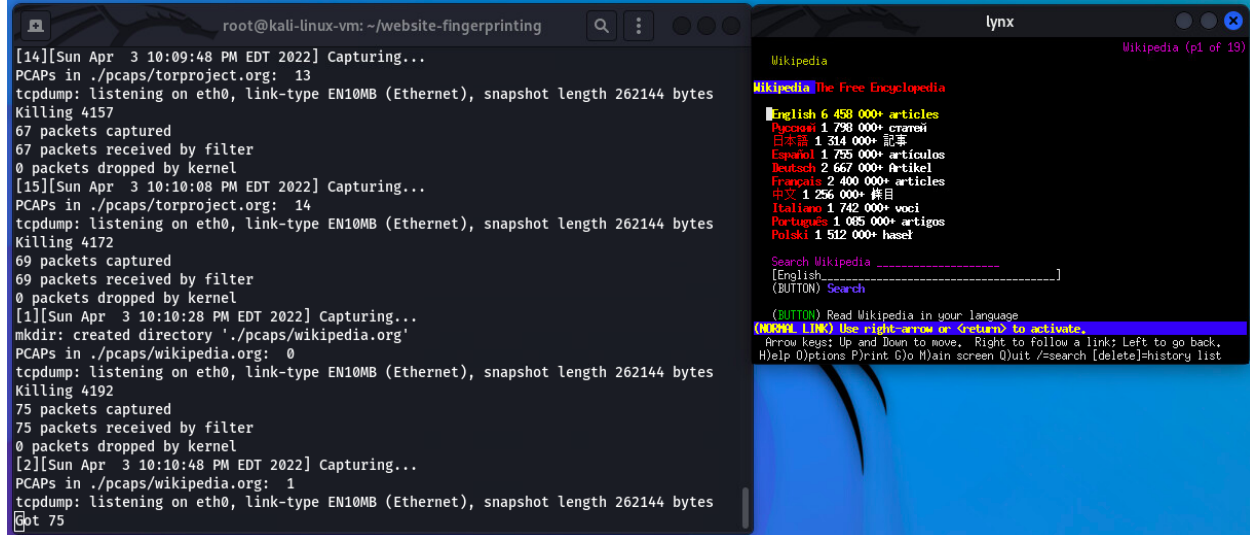
The list of domains the mass-capture.sh script will capture traffic for is given in the config.json file. It will place all captures into the pcaps directory. Mass-capture.sh takes one argument, the amount of captures we will use for each domain. For each capture, a new instance of xterm will be launched to browse to the domain – this allows you to watch and verify that the script is cycling through each domain.

Seven domains in config.json

```
root@kali-linux-vm:~/website-fingerprinting# cat config.json
{
  "pcaps": [
    "duckduckgo.com",
    "github.com",
    "google.com",
    "reddit.com",
    "torproject.org",
    "wikipedia.org",
    "csuohio.edu"
  ]
}
```

1. Clear the existing .pcap files using the following command
`rm -rf pcaps/`
2. Run `mass-capture.sh`. Argument 1 should be the number of captures you wish to do per domain. In our scenario, we will do 15 captures per domain to make sure our classifier is more accurate. This will take some time to run.
`./mass-capture.sh 15`

Mass-capture.sh running



The image shows two terminal windows. The left window is a Kali Linux terminal running the `mass-capture.sh` script. It shows the script capturing data for three domains: `torproject.org`, `wikipedia.org`, and `csuohio.edu`. The script uses `tcpdump` to capture packets on the `eth0` interface. The right window is a Lynx web browser displaying the Wikipedia homepage. The browser shows the Wikipedia logo and a list of articles in various languages, including English, Russian, Japanese, Spanish, German, French, Chinese, Italian, Portuguese, and Polish. The browser interface includes a search bar and navigation controls.

pcaps directory listing after capture

```
root@kali-linux-vm:~/website-fingerprinting# ls pcaps/
csuohio.edu    github.com    reddit.com    wikipedia.org
duckduckgo.com google.com    torproject.org
root@kali-linux-vm:~/website-fingerprinting#
```

Part Three: Training the Website Fingerprinting Classifier

This step involves using the `gather_and_train.py` script. The script will analyze the .pcap files belonging to each domain and train the classifier using k Nearest Neighbors (k-NN). 70% of captures are used for training while the remaining 30% are used for testing.

1. Enter the following command

```
python3 gather_and_train.py
```

```
OUT: 38,IN: 36,TOTAL: 74,SIZE: 36005,RATIO: 0.9473684210526315
OUT: 37,IN: 37,TOTAL: 74,SIZE: 3603,RATIO: 1.0
OUT: 38,IN: 37,TOTAL: 75,SIZE: 36064,RATIO: 0.9736842105263158
OUT: 39,IN: 37,TOTAL: 76,SIZE: 36065,RATIO: 0.9487179487179487
OUT: 39,IN: 37,TOTAL: 76,SIZE: 36066,RATIO: 0.9487179487179487
OUT: 37,IN: 36,TOTAL: 73,SIZE: 36007,RATIO: 0.972972972972973
OUT: 37,IN: 33,TOTAL: 70,SIZE: 3387,RATIO: 0.8918918918918919
OUT: 39,IN: 37,TOTAL: 76,SIZE: 36065,RATIO: 0.9487179487179487
OUT: 37,IN: 37,TOTAL: 74,SIZE: 36066,RATIO: 1.0
    15 pcap files
    - csuohio.edu
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30849,RATIO: 1.0294117647058822
OUT: 30,IN: 35,TOTAL: 65,SIZE: 30848,RATIO: 1.1666666666666667
OUT: 24,IN: 23,TOTAL: 47,SIZE: 24729,RATIO: 0.9583333333333334
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30848,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30847,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30847,RATIO: 1.0294117647058822
OUT: 0,IN: 0,TOTAL: 0,SIZE: 0,RATIO: 0.0
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30847,RATIO: 1.0294117647058822
OUT: 34,IN: 36,TOTAL: 70,SIZE: 30902,RATIO: 1.0588235294117647
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30847,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30848,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30848,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30846,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30848,RATIO: 1.0294117647058822
OUT: 34,IN: 35,TOTAL: 69,SIZE: 30848,RATIO: 1.0294117647058822
    15 pcap files
Training size: 94
Testing size: 11
Accuracy: 90.9090909090909%
root@kali-linux-vm:~/website-fingerprinting#
```

Part Four: Testing the Classifier

We can now test the classifier to see if it correctly identifies one of the domains specified in config.json. The scenario used is closed-word – i.e., the classifier can only tell which of the websites in config.json was visited and cannot be used to classify domains outside of that list as monitored or unmonitored.

Remove all existing .pcap files and create a new capture for your domain of choice (from the config.json file).

1. Remove existing .pcap files

```
rm -rf pcaps/
```

2. Create a new capture for a domain of your choice (remember you will need 2 terminals)

Terminal 0: `./pcaps/capture.sh <domain> lynx`

Terminal 1: `lynx -accept_all_cookies https://<domain>`

```
root@kali-linux-vm:~/website-fingerprinting# ./capture.sh csuohio.edu lynx
mkdir: created directory './pcaps'
mkdir: created directory './pcaps/csuohio.edu'
PCAPs in ./pcaps/csuohio.edu: 0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
Got 69
```


Jump to navigationIFRAME: <https://www.googletagmanager.com/ns.html?id=GTM-PV8NK2>IFRAME: <https://www.googletagmanager.com/ns.html?id=GTM-KVRPTRW>**Home**

- * CSU 2.0
- * President's Office
 - + Messages & Updates
 - + Social Media
 - + In the News
 - + Board of Trustees
 - + Senior Leadership Team
- * Academics
 - + Academic Calendar
 - + Academic Colleges
 - + Degree Programs
 - + Class Schedule
 - + eLearning
 - + Library
 - + Workforce Development
- * Admissions
 - + Undergraduate Admissions
 - + Transfer Admissions

-- press space for next page --

Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list

3. Test your new .pcap file against the classifier

python3 predict.py pcaps/<domain>/<new .pcap file>

```
root@kali-linux-vm:~/website-fingerprinting# python3 predict.py pcaps/csuohio.edu/
csuohio.edu-04-03-22_23\:49\:41-lynx.pcap
* Parsing configuration
Loading the classifier...
OUT: 34,IN: 35,TOTAL: 69,SIZE: 31806,RATIO: 1.0294117647058822
[[0. 0. 0. 0. 0. 0.2 0.8]]
[7] Prediction: csuohio.edu
root@kali-linux-vm:~/website-fingerprinting#
```