

## Study Project: Adversarial Machine Learning

### Website Fingerprinting

#### Introduction

In the previous task you have implemented a script which automatically calls websites. In this part, you extend this script to call a path of random sub-pages and store there raw SSL/TLS traffic for corresponding fetches. Therefore you use the tool *tcpdump* from the *net-tools* package. The tool *tcpdump* prints out a description of the contents of packets on a given network interface. Although, in the case of Website Fingerprinting, the contents of the packets are encrypted, they contain useful meta-information in the context of traffic analysis attacks. For example, the description is preceded by a time stamp.

The tool now allows to store the packet information into a specified output file for later processing. The idea of this project part is to store this TCP/IP dump to a file and map the contents of the dump to the corresponding fetch. Thereby it is important to have a clean dump that only consists out of packets resulting from the automated website fetch.

#### Preliminaries

Please consider the following preliminaries before starting the actual implementation.

- a) If not already installed, install the *net-tools* package; especially *tcpdump*.



- b) Consider possible side effects in your TCP/IP dump and limit them (if not already done in part I)



- c) Which additionally information do you need to map the TCP/IP dump to a given fetch? If necessary extend your browser automation script from part I to provide this information.

#### Subtask 01: Collection of SSL/TLS Traces

Extend your browser automation script such that the following requirements are fulfilled. On a given URL, your script behaves in the following way:



- a) The website with the given URL is visited automatically without the need of human interaction.
- b) On the website the script parse it contains and selects randomly a link which yields to a sub-page of the main-site. From this sub-page we contain to next sub-page, still staying on sub-pages of the main page, until we reached a certain depth, controlled by a given parameter.
- c) Each website and there corresponding randomly selected sub-pages should be called/fetched multiple times which are specified by a parameter `num_instances`.
- d) The complete and raw SSL/TLS traffic occurring during the load of the website / sub-page is saved to a file called *identifier\_runnumber\_website.pcap*. The identifier should be unique. It is mandatory to use the given format for file naming.

- e) The saved file *identifier\_runnumber\_website.pcap* only contains traffic corresponding to the website; every side-effecting data before and after the page loading is truncated.
- f) On **error occurrence**, the script should report this and reject the collected traffic dump. (How do you notice an error?)



Now run your extended Website Fingerprinting script on the given list of URLs containing 100 different websites. Collect the SSL/TLS traces for every website at least 40 times crawling for a depth of 4 sub-pages. Save the resulting traffic corresponding to their run-number and target website each in a file. Make sure that there are no errors during the fetching process which may yield in negative side-effects in the SSL/TLS dump and thus the website fingerprinting attack executed later.

### Subtask 02: Feature Extraction & Visualization

This task concerns the extraction of useful meta-data from TLS packets for which the necessary features will be provided. Please visualize your extracted features to ensure that your fetching algorithm is working properly. Please consider the following preparations before starting with the actual implementation:

- a) The features to be extracted are the number of outgoing packages, the number of incoming packages, the sum (length) of all outgoing packages, the sum (length) of all incoming packages and the cumulative chronological packet sequence (normalised).
- b) Think about a reasonable way to calculate the sampling rate for each cumulative chronological packet sequence.
- c) Choose a library to read pcap-files and to extract the requested data.
- d) Choose a tool to visualize your extracted features.



) How can you identify **corrupted** data in your data set?

Now implement a piece of code that extracts all requested features from your collected data and fulfills the following requirements:



) All the cumulative chronological packet sequences are supposed to be of the same length (50 packages).

- b) The extracted features shall be stored in a file.

The second subtask is to visualize your extracted features by implementing a piece of code that fulfills the following requirements:




- a) For each website visualize all cumulative chronological packet sequences in a separate diagram.
- b) Save each diagram in a separate file. Use the same naming convention as for pcap-files.



) Check your diagrams for **corrupted** data. Could this task be handled automatically?

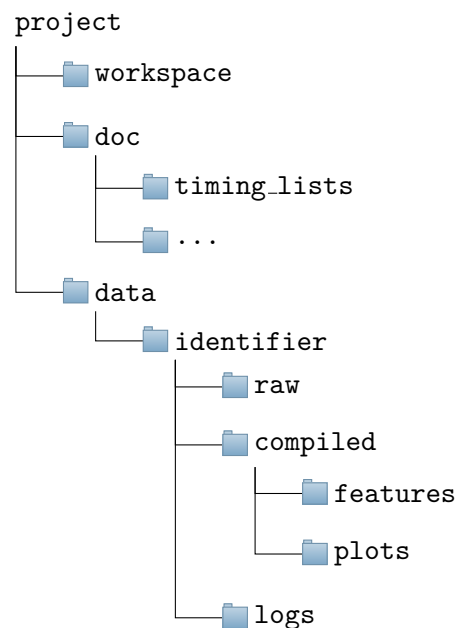
### Subtask 03: Framework implementation

The last task for this part of the project is to combine all your previous implemented scripts to a framework. Your **framework** has to be controlled via command line parameter. Therefore implement at least the following parameters:

Option	Parameter	Description
<code>-h</code>	-	Shows an overview with all implemented command line parameters.
<code>-l</code>	-	Shows the log-file. If more than one exists, list all existing logs.
<code>-v</code>	input file	Shows the diagram of extracted features of input file.
<code>-c</code>	input file	Calls all web pages containing in given input file.
<code>-n</code>	number	Defines how often a web pages shall be called. <b>Default</b> 
<code>-i</code> 	network interface	Defines network interface on which <i>tcpdump</i> shall listen.
<code>-o</code>	output folder	Defines folder, where pcap-files shall be stored.
<code>-e</code>	input folder	Extracts features from all pcap-files containing in input folder.
	output folder	Defines folder, where extracted features shall be stored.

### Directory tree

Use the following directory tree for your git repository:



### Hint

Please note that the effort for this project is calculated for three persons. Therefore it makes sense to distribute the workload.

### Preparation of consultation

Prepare yourself for a short consultation of about 15 minutes. You shall give a short overview of libraries, scripts or already existing tools you have used. Furthermore present and explain your implementations. Give a proof of work through a demonstration of your program/scripts.

*Good luck!*