Wi-Fi Encrypted Traffic Classification.

Wireless Internet Project

Giulia Forasassi, Person code: 10770107,
giulia.forasassi@mail.polimi.it

***Abstract*—This paper presents a project about sniffing and traffic classification on the network. The dataset was created by applying the sniffing technique in monitor mode, identifying 4 possible classes: emails, web pages, video, and videocalls. Then, some supervised machine learning algorithms were evaluated and the experiments show that the Multilayer Perceptron reaches the highest accuracy. Moreover, from the results obtained, it can be seen that the classes that are more easily confused are emails and webpages.**

***Index Terms*—Packet Sniffing, Traffic Analysis, Traffic Classification, Machine Learning Techniques, Monitor Mode**

I. Introduction

Traffic analysis is the process of intercepting and examining traffic data in order to find relationships, patterns, anomalies and misconfigurations in the Internet Network. [1] In particular, network traffic classification is a subfield of traffic analysis and nowadays is one of the most interesting fields of computer science. It consists in classifying the traffic network into categories such as the type of applications (emails, video, videocalls, web browsing, etc.) or the traffic types (normal, abnormal). Through this technique, internet service providers or network operators can manage the overall performance of a network. There are many reasons why network traffic classification is very important, including troubleshooting tasks, provide a high Quality of Service (QoS) for network users, intrusion detection, and network security management.

In the past, many techniques have been proposed for the classification of data traffic on the network, for example: [2]

1) *Port Based Approach*, where each application was identified by its registered port, defined by the Internet Assigned Numbers Authority. Historical development has revealed significant inaccuracy and unreliability of this technique because the number of applications that flow over the network using random or non-standard ports has increased exponentially.
2) *Deep Packet Inspection (DPI)*: performs a matching between the packet payload and a set of stored signatures to classify network traffic. However, it was no longer used since it doesn't detect traffic encryption and protocol encapsulation.

Nowadays, the most common technique used to do traffic classification is *Machine Learning Technique*, which is used by many researchers and got very effective accuracy results.

In this project, we have dealt with traffic analysis by first sniffing the network in monitor mode and then classifying the traffic by comparing four supervised machine learning algorithms: Decision Tree, Random Forest, Gradient Boosting, and Multilayer Perceptron.

II. Packets Sniffing

Packet Sniffing is the practice of collecting some or all packets that pass through a computer network, regardless of how the packet is addressed. In this way, every packet, or a defined subset of packets, may be gathered for further analysis. The collected data can be used for a wide variety of purposes like monitoring bandwidth and traffic. [3]

In order to sniff packets on the network, we need to have a computer connected to the WiFi network, with an installed tool that allows you to intercept the traffic. Various tools permit to do this. In this case was used Wireshark, an open-source software that has a very intuitive graphical interface with many functions for analyzing traffic.

Exist two ways in which sniffing can be done:

1) *Promiscuous mode*: the computer used to sniff packets is associated with an AP, so its traffic will also appear;
2) *Monitor mode*: the computer used for sniffing allows packets to be captured without having to associate with an AP.

Monitor mode only works with wireless networks, while Promiscuous mode can also be applied to wired networks.

A. Setting up the Monitor Mode

Below are the step-by-step instructions used to set the WiFi network card in monitor mode on a Linux OS. [4]

The first step permits to know the wireless interfaces, as we can see in *Figure 1*:

- `iwconfig`

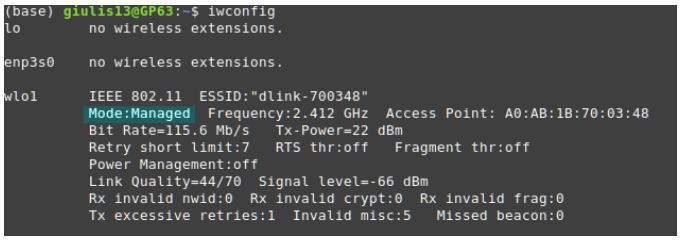

Fig. 1. Wi-Fi Card state: Managed Mode

The second step is to kill any process that could interfere with using the interface in monitor mode, with the following instruction:

- `sudo airmon-ng check kill`

Then it is necessary to disable the wlo1 wireless interface, activate the monitor mode, and finally re-enable the interface, using the following commands:

- `sudo ifconfig wlo1 down`

- `sudo iwconfig wlo1 mode monitor`
- `sudo ifconfig wlo1 up`

It is important to check that the wifi network card is set to the channel used by the AP to transmit data. Otherwise, once known, it can be manually set using the following command:

- `sudo iwconfig wlo1 channel 10`

Where 10 is an example of a possible channel. Now that the wifi network card is set to monitor mode, as can be seen in *Figure 2*, we can start to sniff packets.

```
(base) giulis13@GP63:~$ sudo airmon-ng check
[sudo] password di giulis13:

Found 4 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
   1000 avahi-daemon
   1011 NetworkManager
   1057 wpa_supplicant
   1081 avahi-daemon

(base) giulis13@GP63:~$ sudo airmon-ng check kill

Killing these processes:

    PID Name
   1057 wpa_supplicant

(base) giulis13@GP63:~$ sudo ifconfig wlo1 down
(base) giulis13@GP63:~$ sudo iwconfig wlo1 mode monitor
(base) giulis13@GP63:~$ sudo ifconfig wlo1 up
(base) giulis13@GP63:~$ iwconfig
lo        no wireless extensions.

enp3s0    no wireless extensions.

wlo1      IEEE 802.11  Mode:Monitor  Frequency:2.412 GHz  Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

Fig. 2. Wi-Fi Card state: Monitor Mode

III. Dataset

A. Organization

The created dataset contains 80 examples that were made by sniffing packets over the network using Wireshark. We divided the examples into 4 classes: emails (20), video (20), videocalls (20), and webpages (20). Each example is saved as a .pcapng file and contains a various number of packets, from which it is possible to extract the information necessary to classify it. The example's recording time is a random number between 50 seconds and 2 minutes.

B. Features

We choose 12 features to describe the examples, some of which are upload and download specific. [5]

The features for which we have distinguished between the upload and the download value are:

- total packet length;
- average inter arrival time;
- maximum inter arrival time;
- minimum inter arrival time.

Where the length is the size of the frame expressed in bytes and the inter arrival time is the amount of time that elapses after the receipt of a packet until the next packet arrives. Furthermore, are considered the following features:

- flow time;
- percentage of data packets;
- percentage of control packets;
- percentage of management packets.

Where the flow time is time passed between the first packet and the last one.

IV. Models

This section illustrates the process involved in network traffic classification, as shown in *Figure 3*. As we can see from the graph, there are four sequential steps to follow for analyzing network traffic using ML techniques:

1) *Data Collection*: this is the first step, dealing with capturing real-time traffic from the network;
2) *Feature Extraction*: this step deals with extracting features from network flow. Inter-packet arrival time, packet length, and packet type are examples of features that will be used to train the classifier;
3) *Training Process*: the training phase involves data sampling and classifier training based on the samples. This step uses supervised algorithms to create classification rules and build the model;
4) *Classification Process*: this is the final stage where dataset examples are classified.

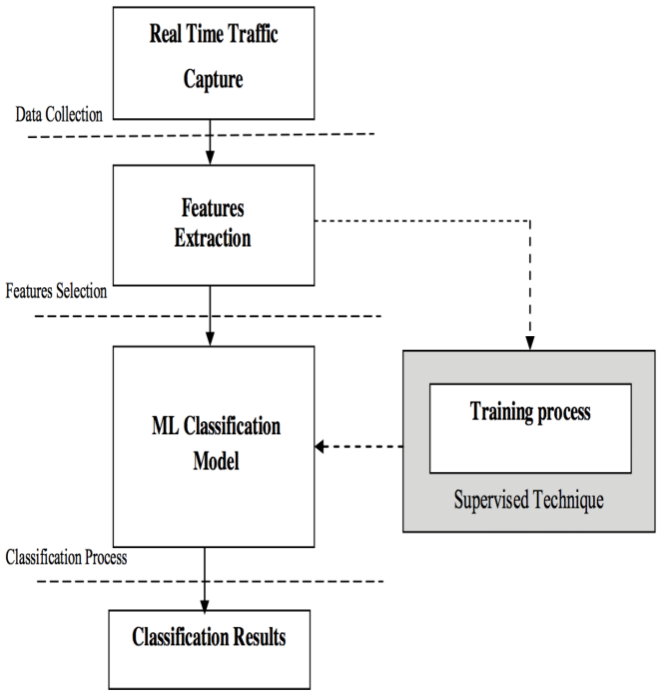

Fig. 3. Network Traffic Classification Model

We have chosen to compare the following supervised machine learning algorithms in terms of accuracy:

- Decision Tree;
- Random Forest;
- Gradient Boosting;
- MultiLayer Perceptron.

V. Experiments

This section shows the performed experiments with their results. First, we analyze the dataset using TSNE to get a 2D representation of the examples. Secondly, we compare the machine learning models: since there are few examples in the dataset, we use a Leave One Out Cross Validation. Lastly, we plot the Confusion Matrix of the best performing algorithm to understand the strengths and the weaknesses of the classifier.

A. TSNE

To get an idea of how similar or different the examples of the various classes are, we have chosen to make a graphical display using the TSNE algorithm shown in *Figure 4*.

From the graph shown, we can see that the video and videocall classes are easily distinguishable, while email and webpage are less so.

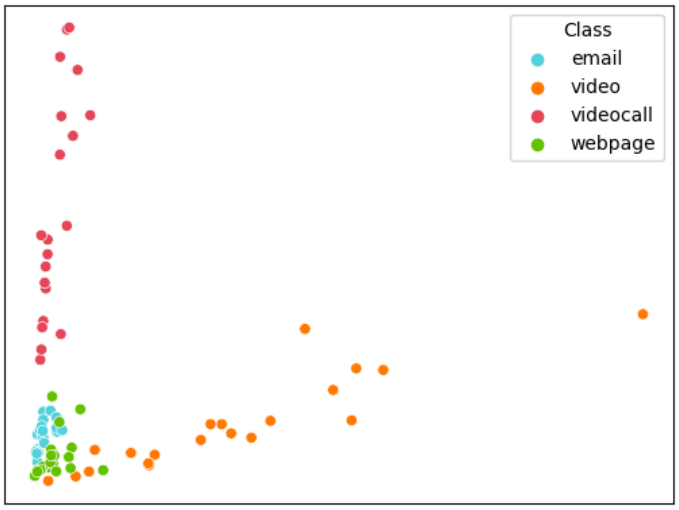

Fig. 4. TSNE Plot of the dataset

B. Leave One Out Cross Validation

Leave One Out Cross Validation consists of forming as many splits as the number of examples in the dataset. At each split, the test set consists of a single example, while the training set contains the remaining examples. Each example appears in the test set once and only once. At each iteration, the model is trained on the training set and evaluated on the single test example. The final accuracy is obtained as the average of the individual accuracies.

Table I shows the accuracy values for the chosen algorithms.

C. Confusion Matrix

To understand the nature of classification errors, we have chosen to show the confusion matrix (*Figure 5*) for the best performing model, namely MLP.

The matrix confirms what was observed from the TSNE graph: the classes that MLP most easily confuses are emails and web pages.

Model	Accuracy
Decision Tree	85.0%
Random Forest	88.8%
Gradient Boosting	87.5%
Multilayer Perceptron	**92.5%**

TABLE I
Accuracies of the models

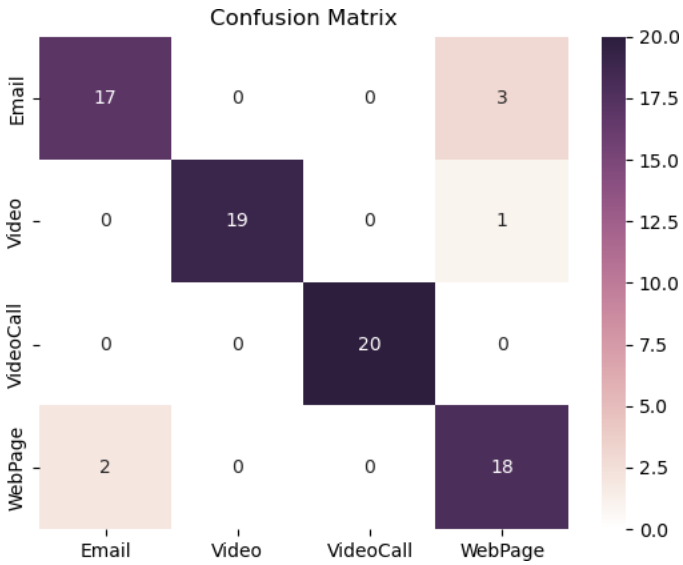

Fig. 5. Confusion Matrix Plot for MLP model

VI. Conclusions

In this work, we sniffed packets over the network using monitor mode to create the sample dataset. After that, supervised machine learning algorithms (Decision Tree, Random Forest, Gradient Boosting, and Multilayer Perceptron) were used to build models for predicting the classes of the examples (emails, video, videocalls, and webpages). The experiments showed that the MLP model achieves the best results in terms of accuracy, and that the classes that are the most difficult to distinguish are emails and webpages.

Appendix
Tools

Below are the main tools that were used to create the following project:

- *Python language*, for the project developing;
- *Scikit-Learn*, free software machine learning library to classify the models; [6]
- *Wireshark*, a free and open source packet analyzer; [7]
- *Pyshark*, a Python library that allows to read the data exported from Wireshark. [8]

REFERENCES

[1] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey."

[2] Traffic classification. [Online]. Available: https://www.wikiwand.com/en/Traffic_classification

[3] Traffic sniffing. [Online]. Available: https://www.wikiwand.com/en/Packet_analyzer

[4] Monitor mode guide. [Online]. Available: https://linuxhint.com/capture_wi-fi_traffic_using_wireshark/

[5] A. Moore, D. Zuev, and M. Crogan, "Descriminators for use in flow-based classification."

[6] Scikit-learn. [Online]. Available: https://scikit-learn.org/stable/index.html

[7] Wireshark. [Online]. Available: https://www.wireshark.org/

[8] Pyshark. [Online]. Available: https://github.com/KimiNewt/pyshark