Wireless traffic profiler

Wireless Internet course project

Master's Degree in Computer Science and Engineering
Politecnico di Milano

Michele Scuttari
920679
michele.scuttari@mail.polimi.it

April 2020

1 Introduction

1.1 The problem to solve

The goal of this project is the live capture of encrypted wireless traffic and its subsequent classification into three possible types of activities: browsing, video streaming and file downloading. A fourth possible activity is indeed the idle one.
Unfortunately, encrypted traffic does not offer any clear information about level 3 (and upper) properties, such as the source IP or the service port. The only exploitable data are the MAC addresses and the ones related to the traffic itself, such as the packets lengths and their arrival times.

1.2 Why machine learning?

Each activity is characterized by a specific pattern in its traffic. For example, browsing is characterized by bursty traffic, while downloads lead to persistently high bandwidth usage; but if this simple discrimination can be done by just setting a threshold on the packets number, parameters become more difficult to identify and set as the number possible activities increases. In this sense, a machine learning algorithm allows to automatically tune the thresholds at the best possible values.

2 Design and implementation

2.1 Requirements

The main hardware requirement is a Wi-Fi network card capable of working in **monitor mode**.
As for the software, the whole project is developed using **Python 3**. The live capture of the wireless traffic is done by leveraging **Tshark**.
Furthermore, also the following library are used and thus required for the project to work:

- **pyshark**: interface for Tshark used to start the live caputre or analyze the previously captured packets to be used to train the model.
- **scikit-learn**: library that features various classification, regression and clustering algorithms.
- **joblib**: library used to save the trained model and load it later.

2.2 Modules

The project consists in three modules:

- **Classifier**: it's the core of the of the project; it computes the traffic statistics and uses machine learning algorithms to perform their classification.
- **Trainer**: it allows to load the pre-captured packets, perform a supervised learning and save the model for later use.
- **Main**: it performs the live capture and classification of wireless traffic using a pre-trained model.

2.3 Classifier

The classification is done using a **Support Vector Machine** (SVM) using **RBF** kernel, because it has been demonstrated as very well performing by other researches.[1] The features used for the classification are the **average packet size**, the **packet size variance**, the **average inter-arrival time**, the **inter-arrival time variance** and the **packet rate**.
The received packets are stored in a sliding window of pre-determined size and, with the passing of time, the packets exceeding the window time frame are removed.
When a packet is added or removed, the features are recomputed. While this is feasible in case of low traffic, high bandwidth activities such as downloads can create up to some thousands packets per second, and the computation becomes unfeasible for a real time environment. To overcome this issue, the mean and the variance, for both packet sizes and inter-arrival times, are computed incrementally[2] when the number of packets exceeds a certain fixed threshold, thus allowing to reduce the computation complexity from *O(n)* to *O(1)* (where n is the number of stored packets).
Furthermore, when on live mode, a prediction history is kept and its size is equal to window size (e.g. for a window size of 10 seconds, 10 predictions are kept). The most frequent prediction in then considered as the correct one. Although increasing a bit the recognition delay, this technique allows to also

[1] Fan Zhang et al. "Inferring users' online activities through traffic analysis". In: Jan. 2011, pp. 59–70. DOI: 10.1145/1998412.1998425; Enrico Testi, Elia Favarelli, and Andrea Giorgetti. "Machine Learning for User Traffic Classification in Wireless Systems". In: Sept. 2018, pp. 2040–2044. DOI: 10.23919/EUSIPCO.2018.8553196.

[2] Katharina Tschumitschew and Frank Klawonn. "Incremental Statistical Measures". In: Mar. 2012, pp. 21–55. ISBN: 978-1-4419-8019-9. DOI: 10.1007/978-1-4419-8020-5_2.

increase the resilience to temporary prediction errors; some traffic patterns (e.g. download and video streaming) are in fact very similar in the short period (i.e. couple of seconds) and can only be distinguished in a longer run.

2.4 Trainer

To train the SVM, packets are pre-captured using Wireshark. The trainer module allows to load and use them to perform supervised learning. The packets having the MAC address of the profiled device are used by the classifier to compute traffic statistics, but to keep the statistics updated also in low traffic scenarios, also beacon frames are passed to the classifier and used just to check if the stored packets exceed the time window. In fact, beacon frames are transmitted from 1 to 100 times per second and per access point, and thus allow to accurately track the passing of time when the data rate drops below 1 packet/s.

2.5 Main

The main module is similar to the trainer one, except that performs live prediction instead of training. The predictions are done just one time per second, in order to periodically fill the aforementioned predictions history.

3 Experimental results

The whole available data is separated between train (80%) and test data (20%). After training the SVM, the test accuracy gives a result of 97.69%. Although it may be considered as an overfit warning, the live classification is quite good: idle and browsing states are perfectly identified, while video streaming may be initially recognized as a download activity, probably due to its variability according to the video quality. Anyway, video streaming recognition tends to stabilize thanks to the predictions history. The download activity is instead perfectly recognized.

4 Conclusions

Being the SVM very efficient, the classification can be extended also to other type of activities such as music streaming. Anyway, more classes implies more training data, and this dramatically increases the training time. To give an idea, just for this tests training usually took up to 45 minutes.
Initially, also more shorter activities such as email sending were considered,

but their recognition is mostly impossible to their short duration; in fact, with just these features available at the MAC layer, an email can be easily confused with other activities, such as a small file network transfer or simple noise.