Digital Forensics - Project Report

Project 1 - Packet classification for mobile applications

Eugen Saraci - 1171697
Università degli Studi di Padova

eugen.saraci@studenti.unipd.it

January 13, 2019

In the last years the large diffusion qualche altro termine non sarebbe male of SSL/TLS has made it harder for attackers to collect clear text information through packet sniffing or, more in general, through network traffic analysis. The main reason for this is that SSL/TLS encrypts the traffic between two endpoints, which means that even though packets can still be easily captured, no useful information can be inferred from the packet's content without having the encryption keys.[1]

The authors of [1] and [2] showed that by training a machine learning algorithm with encrypted traffic data, one could correctly classify which actions a user actions performed on the most common Android applications such as Facebook, Gmail, or Twitter. This could easily lead, through a correlation attack, to the full deanonimization of fake, privacy preserving identities.

In this work I try to reproduce the results achieved in [1] and [2] by implementing the classification model as described in the papers.

1 Introduction and Notation

1.1 Actions and Flows

In order to easily understand this work, in the next two subsections I introduce two general concept used throughout this report.

[1] It is worth mentioning that the endpoints of the communication (i.e. source and destination IP addresses) are transmitted in clear text for routing purposes; by performing a DNS lookup of the addresses and attacker could easily infer what site a user is visiting.

1.1.1 Action and Action Label

An *action* is simply the action performed by a user while using one of the aforementioned Android apps. Examples of actions are: clicking on a profile page, tweeting a message, sending an email etc. Please note that "clicking on a profile page" is what I refer to as the *action label*, in many cases I use the words "action" or "action flow" to refer to the set of flows that represent that action. spiegare meglio sta roba

1.1.2 Flows

When a user performs an action some encrypted packets are exchanged with the destination server. A flow consists of the sequence of the byte sizes of the exchanged packets. If the packet is going from the user's phone to the server it is said to be *outgoing*; if the packet is coming from the server to the user's phone it is said to be *incoming* and it is marked with a "-" sign before the integer number representing its size.[2] An example of a 5 packet flow is: `[-12, 80, 90, -111, 30]`. Please note that a single action performed by the user usually generates multiple flows of different dimensions, by that follows that an action actually consists of multiple flows. The techniques used by the authors to determine which flows belong to which action, the ordering of the packets, the packet capturing system, the packet filtering system, and the statistical analysis on the flows will not be treated in this report since the starting point for this work comes when the dataset is already constructed.

1.2 Notation

- A: an action; it represents a sequence of flows;
- a: action label;
- F: a flow; it represents a sequence of packets;
- p: a single packet, it is an integer number representing the size in bytes of that packet.

Please note that all of the above can be subscripted by indexes; a subscripted element means that that element is the i-th element of a sequence, e.g. F_i is the i-th flow of a sequence of flows (possibly an action A).

By this follows that $A_i = [F_1, \ldots, F_n]_i$ and $F_i = [p_1, \ldots, p_m]_i$. Note that n and m are possibly (and probably) different for each flow F_i and for each action A_i, even for two actions A_i, A_j where $a_i == a_j$.

1.3 Dataset

The dataset consists of 252,151 rows (samples) and 12 columns (features), moreover, the data collected contains packets of different actions for 7 different Android applications: Facebook, Twitter, Gmail, Google Plus, Tumblr, Dropbox, and Evernote.

[2] The "-" sign is just notation, packets cannot have a negative size.

action_start	app	action_label	...	flow
1383129102.11	facebook	open facebook	...	[-15, 75, 144]
1383129102.11	facebook	open facebook	...	[-55, -255, -333, 122, -55]
⋮	⋮	⋮	⋮	⋮
1383129102.11	facebook	open facebook	...	[12, 12, 155, 155, -18, 255]
1383129244.01	facebook	click menu	...	[78, -206]
⋮	⋮	⋮	⋮	⋮

Table 1: Some rows of the dataset.

In Table 1.3 we can see the format of the dataset. I have purposively hidden some of the columns since I do not use them in this work. In the table we can see that each row contains the features of a single flow; to decide which flows belong to which action we compare the *sequence_start* field which is a fake but consistant timestamp of when the user started that action; When two rows have the same timestamp we can safely assume that they belong to the same action.

bisogna dire da qualche parte che lavoro con un dataset ridotto

2 Machine Learning

Goal: as in Figure 1 we want a machine learning algorithm $\mathcal{L}$ that predict the *action label* a_i given the *action's flows* $A_i = [F_1, \ldots, F_n]_i$.

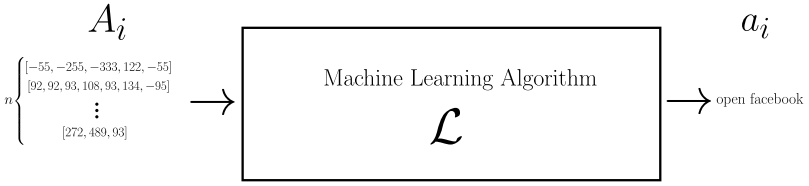

Figure 1: The ideal model predicti the label a_i for the a new input instance A_i.

Given the fact that each action generates multiple flows of different lengths, we know that standard supervised learning approaches are hard to apply. To see why standard approaches would not work we need to think about the structure of the input and output spaces. Our output space i.e. what we want to predict would be the *action label* a_i, while our input space, i.e. the predictors, would be the flows generated by a_i which we denote as $[F_1, \ldots, F_n]_i$. One way we could represent flows as features would be to have a feature for each flow F_j generated by the action a_i, and the value of the j-th feature would be the sequence of byte sizes of the flow $[p_1, \ldots, p_m]_j$. Because of the different number of flows for each action n we would immediately see that each row could possibly have a different number of features. The main problem of this approach,

other than the just mentioned diverse dimensionality, is that we are artificially defining features with no real justification; in other words, we have no reason to associate the first flow of an action A_i with the first flow of another action A_j by viewing them as the first feature of their respective samples.

To overcome this obstacle the original authors applied a two stage process. In the first stage, by using an unsupervised clustering algorithm, they addressed their missing knowledge about the number of flows and their features. The objective is to identify some kind of intermediate representation of the data $\Phi(A_i)$ that can be exploited by the second stage of the process, which will perform a canonic classification (Figure 2). In the remainder of this section we describe the algorithms used in the two stage process and my personal implementation.decidere come strutturare sta roba.

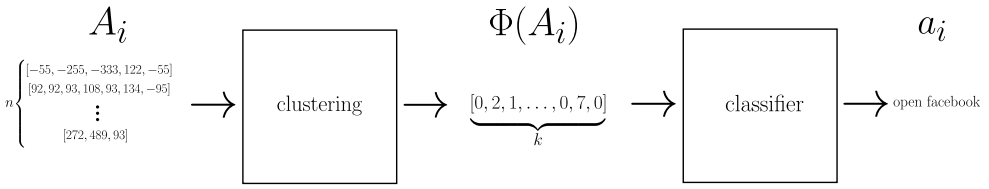

Figure 2: The two step classification of a new sample; first the new sample A_i goes through the clustering algorithm which computes the intermediate representation $\Phi(A_i)$, then $\Phi(A_i)$ is used by the classifier to predict the correct label a_i. k is the number of clusters, a more detailed description of k and Φ can be found in Section Sezione implementativa

2.1 Clustering

In the typical unsupervised clustering scenario the goal of the algorithm is to find k groups called clusters, where the *inter-cluster* similarity is very low while the *intra-cluster* similarity is maximized, meaning that samples belonging to one clusters are very similar to each other while still being very different from samples belonging to other clusters. Notice that the number of clusters k and the similarity function between samples have to be explicitly defined by the teacher. The input of the clustering algorithm is usually the whole datset while its output is a list of intergers $[c_i, \ldots, c_N]$ where N is the total number of samples and $c_i \in [1; k]$ is a an integer number that denotes the cluster to which the i-th sample has been assigned by the algorithm.

2.1.1 Hierarchical Agglomerative Clustering

In this specific instance I used (as suggeseted in the papers) a **Hierachical Agglomerative Clustering**, **HAC** for short. HAC starts by creating a cluster for each flow; at each step HAC reduces the number of clusters by merging (agglomerative) the closest clusters together based on the distance function and other parameters; HAC will stop whenever the desired number of clusters k is reached. The structure that HAC

generates can be viewed as a tree (hierachical), that structure is called dendrogram; in Figure 3 we can see the dendrogram for 20 samples (i.e. 20 flows). In my implementation the input consists in all the flows contained in the dataset[3], the number of clusters k is set to 50, and the distance function used to compute similarity between flows is the Dynamic Time Warping described in the following paragraph.

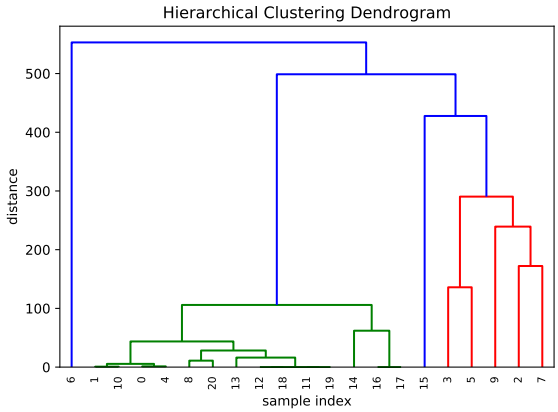

Figure 3: Dendrogram for 20 flows; the number of clusters can be seen by counting how many vertical lines get crossed by an imaginary horizontal line. Different horizontal lines will of course determine different clusters and different distance measures; the dendrogram allows the designer to decide at what level to draw the line i.e. which value to assign to k by looking at the distance value and at the hierarchy itself.

2.1.2 Dynamic Time Warping

DTW is an algorithm used generally to find the best match between two time series even when they have different lengths and/or have repetitions or deletions in them. If we view every single flow F_i as a series of packets $[p_1, \dots p_m]_i$, we can use **DTW** to measure how similar two flows are. This similarity measure is needed by the HAC algorithm to compute the distances between clusters.

2.1.3 Feature Extraction

The HAC output is, as stated before, a list of N integers $[c_1, \dots, c_N]$ where c_i is the cluster assigned to the i-th input sample. This is different from what is reported in Figure 1, in fact the clustering result needs to be rearranged before being given as input to the classifier. The aim of the rearrangement is to actually generate a new intermediate dataset where the target we want to predict are still the actions' labels but the features are now different. We generate k features for each action A_i; the

[3] We are just using flows now, the concept "actions" is being ignored for the moment since it is not really needed for the clustering stage.

action_label	C1	C2	...	C50
open facebook	0	11	...	6
writing search	1	1	...	5
back to news	0	0	...	0
open facebook	0	9	...	5
⋮	⋮		⋮	⋮

value of the j-th feature is the number of flows of A_i that ended in the j-th cluster during the clustering algorithm. This whole process composed by *clustering* and the final rearrangement is what we call feature extraction and denote with the symbol Φ, so $\Phi(A_i)$ is the intermediate representation of A_i and is considered to be a single input sample of the new intermediate dataset.

2.1.4 Classification and Random Forest

Random forest is a very good algorithm

3 Evaluation

3.1 Experimental Setup

3.2 Experimental Results

References

[1] Mauro Conti, Luigi V. Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2015. Can't You Hear Me Knocking: Identification of User Actions on Android Apps via Traffic Analysis. In Proceedings of the 5th ACM Conference on Data and Application Security and Privacy (CODASPY '15). ACM, New York, NY, USA, 297-304. DOI: https://doi.org/10.1145/2699026.2699119

[2] Conti, M., Mancini, L. V., Spolaor, R., & Verde, N. V. (2016). Analyzing android encrypted network traffic to identify user actions. IEEE Transactions on Information Forensics and Security, 11(1), 114-125.