PROJECT PROPOSAL REPORT

Development of a framework to gather the data from mobile devices for the purpose of user activity / behavior classification

December 12, 2017

Department of CSIS
BITS Pilani

Team:

Devamalya Hazra [2016H1120169P]
Mohit Agarwal [2016H1120161P]
Rishabh Sharma [2016H1120154P]
Shubham Bansal [2016H1120157P]

Introduction

With the increasing use of mobile devices, a huge amount of network traffic is generated on a daily basis. Had the data been unencrypted, it would be easy for eavesdroppers to tap into the data. However, packets generated from mobile applications are generally encrypted to prevent such intrusions.

Our aim is to gain valuable insight into the encrypted data by analyzing the network flow thereby, creating a user profile using various machine learning techniques.

Phases

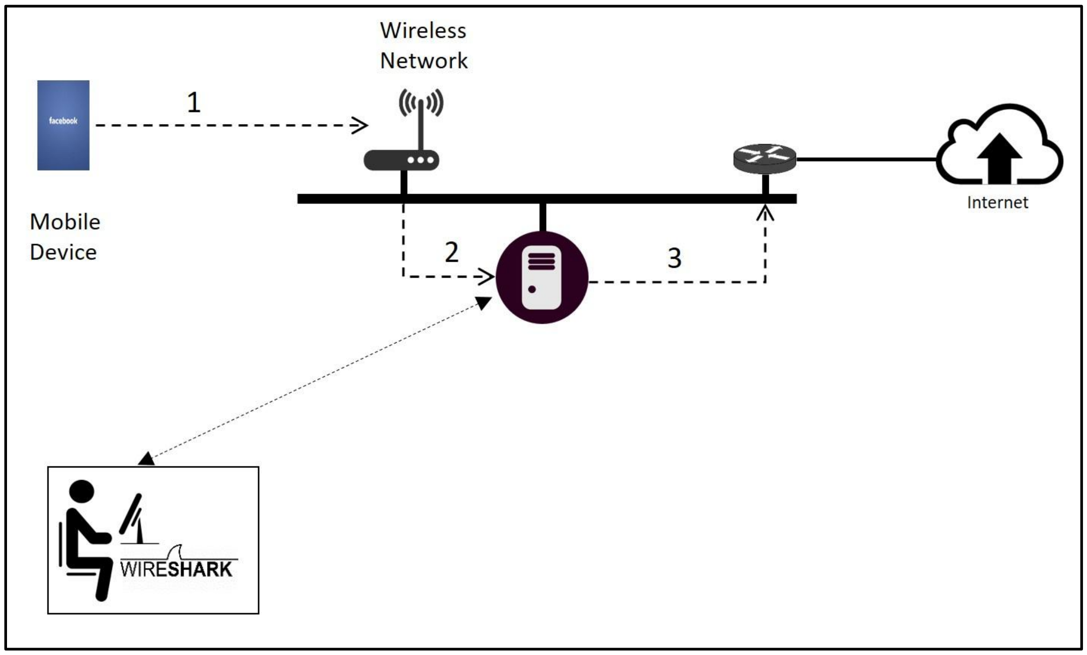
The development of the framework involves the following four phases:

I. **Packet Capture**: The packets generated from the mobile data are not limited to a specific application, but are generated from multiple sources (apps and services running in the background). The first step in capturing app specific data is to restrict the background data generated from other apps and services. A blocking service is configured to block data from all other apps other than the one under consideration. This way we are able to generate packet capture (.pcap) files targeting specific mobile applications.

   The encrypted packets generated by the mobile applications can be captured using the following techniques:

   i. **tcpDump with Android Debug Bridge (ADB):** tcpDump[1] is a packet analyzer that allows users to display packet information received over a network. Android Debug Bridge (ADB) is a command-line tool that lets us communicate with an Android device. We write an adb script to establish a connection with the mobile device. Further, we install tcpDump on the target device and invoke it via the adb shell. The packets intercepted by tcpDump can be accessed using the adb shell. The problem with this technique is that tcpDump requires root access of the mobile device to be able to intercept the outgoing app traffic. But, it is infeasible to grant root permissions to tcpDump on every device.

   ii. **Tinyproxy:** Tinyproxy[2] is a HTTP proxy server daemon for POSIX operating systems. It can be used to intercept network data at a centralized server, much like VPN servers. The advantage of using tinyproxy is that the packets that it intercepts are accessible through sniffing utilities like WireShark. Moreover, it is scalable as it can be used to capture packets from hundreds of users, by just making them connect to the proxy server.



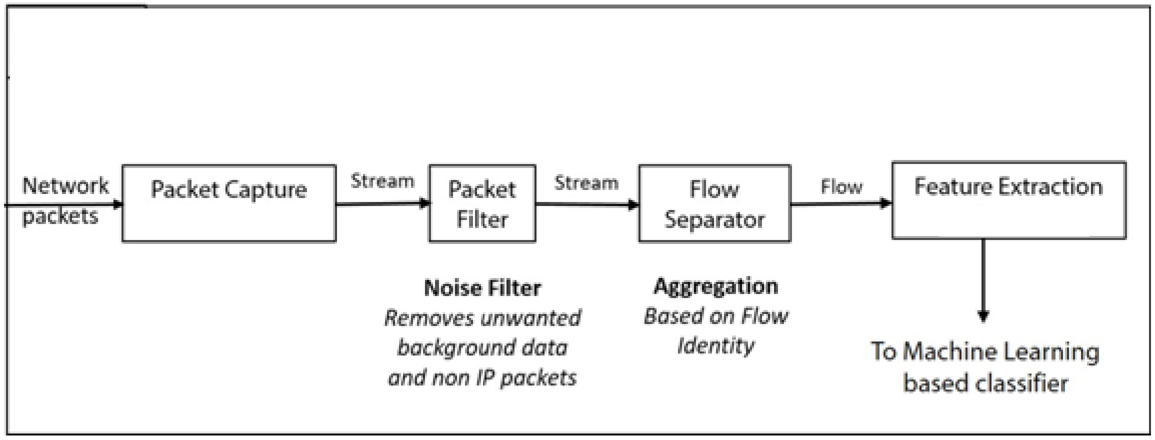
Packet Capture using Proxy Server (TinyProxy)

We have used tinyproxy for the purpose of capturing app data from various Android devices because it resolves the problems posed by the other techniques.

II. **Packet Filter**: The .pcap files generated in the previous phase contain unnecessary data, like the background data of the proxy server, non IP packets (SSDP, ARP), etc. This unnecessary data has to be filtered out to get pure app-specific .pcap files. In order to achieve this, we have used Python scripts(scapy) that can parse the .pcap files and separate the app-specific data from the rest.

III. **Flow Separator**: This phase involves identifying unique flows from the packet stream. A network flow is defined as time ordered sequence of packets exchanged between two peers during a session. The choice of initial handshake as a part of flow; amount of data to be included in each flow; time duration using which a flow can be separated from another flow are few of the challenges which needs to be addressed. Data exchange in mobile applications like WhatsApp usually happens over a long course of time with fewer data packets exchanged in each flow so the definition of flow needs to be kept appropriately keeping these considerations in mind.
.

IV. **Feature Extraction**: In this phase, features are identified and extracted from each aggregated flow. The features can be like: total bytes, total packet count and direction based features: minimum packet length, maximum packet length, average packet length, standard deviation of packet length, Other protocol features can be TLS specific features (TLS version, cipher suite, client public-key length), initial data packets information (URL, DNS addresses/hostname), sequence of packet length and timestamps, byte distribution, etc.



Phases of framework development

Implementation

In this section, the implementation details of the ideas mentioned in the Phases section is discussed.

Phase I. Packet Capture: We have created a blocking service that can be configured on the Android devices so that all the background data of the other applications can be restricted and we can obtain pure app-specific .pcap files. We have also used tinyproxy for the purpose of capturing app data from various Android devices because it resolves the problems posed by the other techniques.

Phase II. Packet Filter: We have used Python scripts(scapy) that can parse the .pcap files and separate the app-specific data from the rest. The non-IP packets like ARP, SSDP, etc. have been removed. The data packets corresponding to the proxy server have been removed by filtering on the basis of the IP address of the proxy server.

Phase III. Flow Separator: The filtered .pcap files have been used to identify flows of data from the mobile application. A flow is defined as a five tuple (SIP,DIP,SPORT,DPORT), where SIP represents the IP address of the source; DIP represents the IP address of the destination; SPORT represents the source port; DPORT represents the destination port. In simpler terms, any data exchange between a client and server is considered as a flow as long as a client port that communicates a server port is open. So, for a collection of packets to be defined as a flow, they must have the same source and destination sockets.

Phase IV. Feature Extraction: Various features have been extracted from the identified flows. These features serve as the basis for user-activity classification. The features extracted have been classified into two categories:

a. **Size-based features**- These features are concerned with directly, or indirectly the size of the flow. Two size-based features have been identified.
    - *Number of packets*- It denotes the total number of packets contained in the flow.
    - *Average packet size*- It denotes the mean of the size of the packets in a flow.

b. **Direction-based features**- These features deal with the direction of communication between the client and the server. The identified direction-based features are summarized in Table 1.

Table 1: Identified direction-based features

Sl. No.	**Feature**	**Description**
1	Minimum packet length (forward)	Size of the smallest packet among all the packets that originate at the client and ends at the server
2	Maximum packet length (forward)	Size of the largest packet among all the packets that originate at the client and ends at the server
3	Mean packet length (forward)	Average size of all the packets that originate at the client and ends at the server
4	SD packet length (forward)	Standard Deviation size of all the packets that originate at the client and ends at the server
5	No. of packets (forward)	Total number of packets that originate at the client and ends at the server
6	Flow size (forward)	Total size of packets that originate at the client and ends at the server
7	Minimum packet inter-arrival time (forward)	Smallest inter-arrival time among all the packets that originate at the client and ends at the server
8	Maximum packet inter-arrival time (forward)	Largest inter-arrival time among all the packets that originate at the client and ends at the server
9	Mean packet inter-arrival time (forward)	Average inter-arrival time of all the packets that originate at the client and ends at the server
10	SD packet inter-arrival time (forward)	Standard-deviation of the inter-arrival time of all the packets that originate at the client and ends at the server
11	Minimum packet length (backward)	Size of the smallest packet among all the packets that originate at the server and ends at the client

12	Maximum packet length (backward)	Size of the largest packet among all the packets that originate at the server and ends at the client
13	Mean packet length (backward)	Average size of all the packets that originate at the server and ends at the client
14	SD packet length (backward)	Standard Deviation size of all the packets that originate at the server and ends at the client
15	No. of packets (backward)	Total number of packets that originate at the server and ends at the client
16	Flow size (backward)	Total size of packets that originate at the server and ends at the client
17	Minimum packet inter-arrival time (backward)	Smallest inter-arrival time among all the packets that originate at the server and ends at the client
18	Maximum packet inter-arrival time (backward)	Largest inter-arrival time among all the packets that originate at the server and ends at the client
19	Mean packet inter-arrival time (backward)	Average inter-arrival time of all the packets that originate at the server and ends at the client
20	SD packet inter-arrival time (backward)	Standard-deviation of the inter-arrival time of all the packets that originate at the server and ends at the client

Future Work

So far we have captured the outgoing packets and tagged them with the applications they belong to. Our next step is to take up specific apps, and tag their packets with the activities performed within the app. For example, we should be able to distinguish between the packets sent while uploading a picture, and while updating ones status within the Facebook app. For this we need a detailed understanding of the underlying architecture and encryption protocols used by the app. After reading multiple research papers[3] we encountered multiple definitions of network flow so we plan to define network flow which helps us in solving user activity identification problem.

Further steps involves identifying features (statistical and other network related features) and converting the .pcap file into the corresponding feature vector and we plan to use python script for converting the .pcap file to feature vector with appropriate user activities as classes/tags. Once the network flows are tagged successfully, we can use these feature vectors to train a model using machine-learning algorithms, and classify unknown encrypted network flows.

References

[1] Analyzing Android Network Traffic. [Online]. Available: https://code.tutsplus.com/tutorials/analyzing-android-network-traffic--mobile-10663

[2] How to install and configure Tinyproxy on Ubuntu 14.04. [Online]. Available: https://www.rosehosting.com/blog/install-and-configure-tinyproxy/

[3] Conti M, Mancini LV, Spolaor R, Verde NV. Analyzing android encrypted network traffic to identify user actions. IEEE Transactions on Information Forensics and Security. 2016 Jan;11(1):114-25.

[4] Introduction to Cisco IOS NetFlow - A Technical Overview. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html