Can't you hear me knocking: Identification of user actions on Android apps via traffic analysis

Mid-Sem Progress Report for CSE534

Name: Pranjal Sahu, Sushant Ojal

April 25, 2017

1 Project Progress

Our project plan can be summarized in terms of two steps:

- Network traffic pre-processing and labeling procedure. (Figure 1) and,
- Clustering and user action activity detection. (Figure 2)

All of our code is available on github.

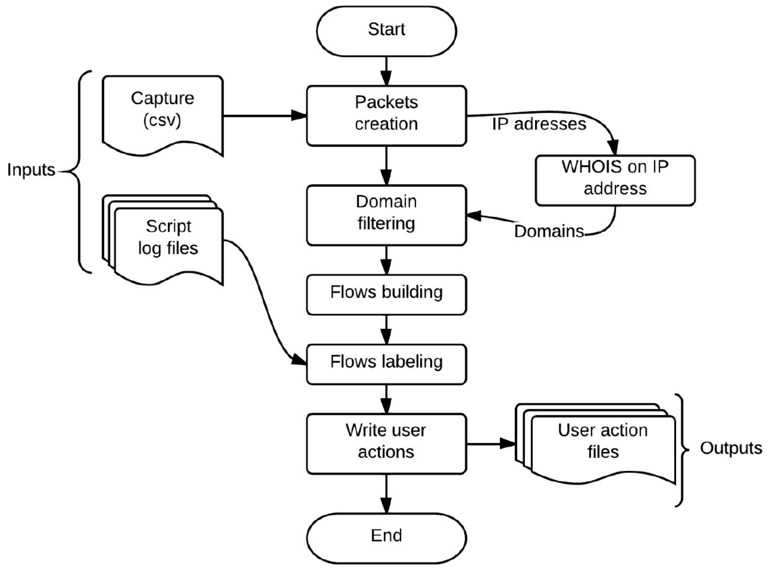


Figure 1: network traffic pre-processing and labeling procedure

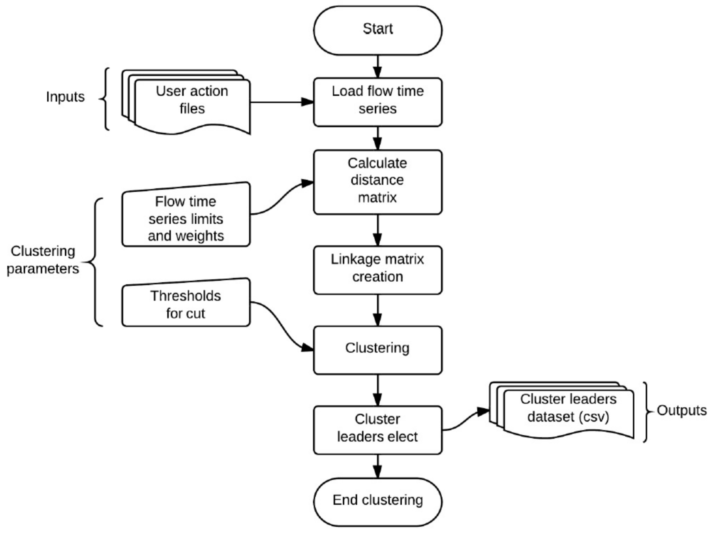


Figure 2: Flowchart for clustering procedure

1.1 Network traffic pre-processing and labeling

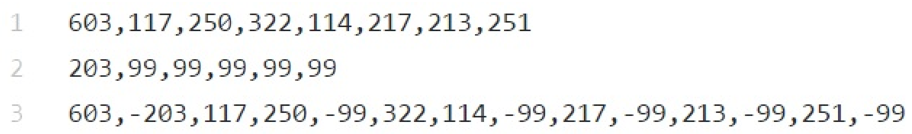
As a first step, we needed to create feature vectors out of the network traffic which was generated on using a particular mobile application. We used a MotoX with Android version 5.1 (Lollipop) to capture the network traffic generated by the Facebook mobile application. Using ADB shell commands, we emulated basic user interactions of touch ,type and swipe. Using these we simulated 20 iterations of three user interactions on Facebook:

- Posting on a user wall
- Sending a message
- Open a user profile

Code for event generation is available in **data_collection_script.py**. We captured the packets generated by this through Wireshark. Using the Wireshark scripting functionality, we extracted each flow and created a separate pcap file for every flow [**extract_flow.py**].

We removed the packets responsible for ACKs, retransmissions, SYN, SYN-ACK, DNS and IGMP so that only the relevant packets are remaining [**filter_flow.py**]. We also filtered out the flows based on IP which are related only to Facebook [**label_flow.py**].

We then labelled the flows in terms of the user actions by labelling all flows whose start time stamp or end time stamp was lying between the duration of event end plus 10 seconds [**label_flow.py**]. As shown in Figure 3 then created a feature vector per flow comprising of three arrays containing packet sizes of the outgoing, ingoing and combined incoming and outgoing packets [**write_feature.py**].



```
603,117,250,322,114,217,213,251
203,99,99,99,99,99
603,-203,117,250,-99,322,114,-99,217,-99,213,-99,251,-99
```

Figure 3: A feature vector. The first array shows the packet size of the outgoing packets, the second for the incoming packets and the third for both.

We are only considering packets which were transmitted upto 10 seconds after the user interaction was completed for a event (post on wall, send message etc.). Also we filtered out features which had length smaller than 8. At the end of this process, we obtained 93 feature vectors.

1.2 Clustering

For clustering, we needed a distance metric. We have used the dynamic time warping algorithm to align any two feature vectors. Specifically to find the distance between any two arrays containing packet lengths, we are using the **fast-dtw** python package. We then used **scipy.cluster.hierarchy** in SciPy python library [**clustering.py**]. We tried clustering the feature vectors with different weights assigned to each of the incoming, outgoing and incoming+outgoing feature vectors.

We were able to cluster the feature vectors with varying packet distributions for different weights. The following dendogram for instance, shows the packet distribution in which the weights for the outgoing distribution have been assigned as 1, and 0 for both the incoming and incoming+outgoing feature vectors. As the figure 4 shows the clusters are sufficiently separated.

2 Remaining work

At present we have only considered unsupervised classification by using hierarchical clustering. It remains to see how the Precision, recall matrices perform using this approach. We also want to use a supervised classification approach. Finally we want to do this whole procedure for another application like gmail etc.

3 References

Can't you hear me knocking: Identification of user actions on Android apps via traffic analysis. *Mauro Conti, Luigi V. Mancini, Riccardo Spolaor, Nino V. Verde*

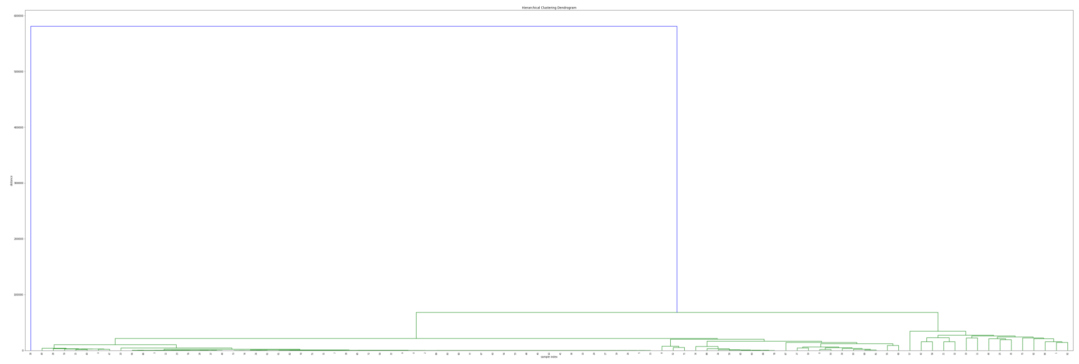


Figure 4: Dendogram