# Project phase 3

WPI CS4516 Spring 2019 D term

*Instructor: Lorenzo De Carli (ldecarli@wpi.edu)*

# Phase overview

- Implement a classifier for encrypted traffic
  - Capture traffic samples generated by various actions performed in the Android VM
  - Split dataset in training/evaluation
  - Use training traces to train model (Random forest, SVM, or another method of your choice) to distinguish between actions
  - Verify accuracy using evaluation traces

# Traffic capturing/processing

- Use tcpdump (or any tool of your choice) to capture traffic generated by performing a set of actions on the Android VM
- Convert network flows in traces into *feature vectors* suitable for a classifier

# Apps and actions of interest

- Start Android browser (homepage must be set to en.wikipedia.org)
- Start Youtube app
- Start Weather Channel app
- Start Google News app
- Start Fruit Ninja app
- (all apps above can be installed for free using the Google Play app, already pre-installed)

# How to capture traffic

- Start tcpdump, execute action, terminate tcpdump
  - Either by hand, or by using Android test automation commands (*start*, *monkey*) via adb shell
  - You should aim at having ~50 traces per action (although less may work too)
  - Label each trace w/ the action it captured

# How to process traffic/2

- **Data cleanup:** the Android VM is relatively quiet in terms of network chatter, but you may end up capturing flows unrelated to the action you are performing
- Suggestion for data cleanup:
  - Discard obvious noise (e.g., ARP)
  - Look at DNS requests to figure out the IPs of flows generate by the app
- You may also decide not to cleanup your data and hope the classifier can figure it out
- Looking at captures using Wireshark may help

# How to process traffic/3

- Once you cleaned up the traces, divide them in bursts as explained last week
- All traffic in each burst must be partitioned into flows
- A flow is a set of packets sent between the same pair of addresses/ports and carrying the same protocol (TCP/UDP) (note, traffic flows in both directions)

# How to process traffic /4

- Once you have a set of flows, you must convert them in feature vectors
- Vectorization: the process of representing an object with a vector of scalar features, suitable for classification algorithms
- Features you can’t use:
  - IP addresses
  - MAC addresses
  - Packet payloads
- Everything else is fair game

# How to process traffic/5

- Examples of vectorization:
  - Convert each flow in a vector including the lengths of the first 10 packets
  - Convert each flow in a vector containing statistical features of the sequence of packet lengths
  - …

# I have the vectors, now what?

- Train a classifier to distinguish between vectors generated by different apps
- Zhuoqun will give a brief demo later for those of you not familiar with scikit and machine learning in general

# Phase 3 deliverables

- A python script named classifyFlows that, given a pcap trace, must print out a list of bursts, flows in every bursts, and label of the action that generated a certain flow (if any)
- Output format:

```
➢ classifyFlows mytrace PCAP
Burst 1:
<timestamp> <src addr> <dst addr> <src port> <dst port> <proto>\
<#packets sent> <#packets rcvd> <#bytes send> <#bytes rcvd> <label>
```

<label> must be either the name of an app, or unknown if the classifier is unable to determine which app was detected

# Phase 3 deliverables/2

- Internally, your code must:
  - Partition the traffic in bursts
  - Partition each burst into flows
  - Generate feature vectors from flows
  - Attempt to classify each vector using the model you trained
- We will evaluate the accuracy of your code in classifying traces generated using the Android VM

# Phase 3 deliverables/3

- By the phase 3 deadline (4/15, 10:45am) you must upload to Canvas a .zip file containing:
  - Your Python script
  - A README file specifying any Python package on which your code depends, and any information we need to be aware of when testing your code
  - If your code has known limitations or issues, also briefly document them in the file.