CS 371 – First Project

Deadline: 04/15/2019

Traffic Analysis based on Machine Learning

Maximum points 100

General information

Maximum group size 3.

The project report and the code should be submitted on Canvas as a single Zip file by 04/15.

The project should be discussed with the TA during the week of the 04/15. Contact the TA to set up an appointment.

Packet sniffer

A packet sniffer is a piece of software designed to "sniff" the packets that are sent/received by your computer. Such software poses obvious security concerns, since the user may not be aware of its presence, and the traffic may be spied for malicious purposes.

In this project we will play the role of the bad guys and design a packet sniffer to analyze the data that is transmitted over the network while you use your computer. The goal is to extract features from the flow of packets sent and received. These flow features are used to train several machine learning models. Finally, we want to verify that these trained models can understand what applications are running in your computer just looking at the ongoing *flows* of packets.

The project will be developed in Python using a library named Scapy, which can be downloaded here: https://scapy.net/.

Scapy is a Python library that allows you to manipulate network packets. To use Scapy you should have some knowledge of Python and some background on network protocols. Once you install Scapy, you can either use it interactively from your terminal or through Python by just importing the library. However, for this project, using Python scripts is necessary.

The power of Scapy lies behind the ease of use. For example, in order to sniff traffic in a network all it takes is a single function called `sniff()`. This function is the only thing that we actually need from this library. All the rest of the programming is not going to depend on specific libraries, so feel free to use the Python tools, libraries or data objects that you are most comfortable with. However, every program will need to end with the creation of a .csv (comma separated values) file that will be used as a training set for the machine learning model. This file will then be used by another Python script where the machine learning algorithm is implemented.

Project design

A very basic skeleton file (scapy-skeleton.py) for the packet sniffer has been uploaded to Canvas. Also, you can use it to test if Scapy has been set up properly. This file uses the sniff function and has a basic callback implemented that shows how to extract some of the fields in the header of the packet. You will most likely need to modify such function for your own purposes by extracting the fields that you believe are necessary to extract the features you want to calculate. This file will be used to generate the training set that has been mentioned before. A more detailed step-by-step guide is given below:

- Use sniff function to start sniffing packets. The information you can get from each sniffed packet are:
    - Ethernet header: dst, src, type;
    - IP header: version, ihl, tos, len, if, flags, frag, ttl, proto, chksum, src, dest;
    - TCP header: sport, dport, seq, ack, dataofs, reserved, flags, window, chksum, urgptr, options;
    - UDP header: sport, dport, len, chksum;
    - Occasionally some other protocol might show up, such as ARP, which is a transport layer protocol.
- Make wise use of the pnr callback to capture and store useful information from the sniffed packets.
    - To store such information you can use any data structure, data type, or even write it into a file.
- Detect network "flows" among the sniffed packets. Flows are defined by the tuple (src IP address, src port, dest IP address, dest port, transport protocol). The transport protocol is either TCP or UDP. You can assign to each flow an ID, which is shared among all packets of that flow.
- Come up with features that you can use to extract some useful attributes/characteristics of the flow. This is the "creative" part of the project. You can think about your own features that make sense in this context. Different features may provide different (better or worse) results.
- Create a csv file with the following characteristics:
    - Each row represents a flow
    - Each column represents a feature
    - The last column is dedicated to the label (i.e., the activity that generated that flow). You will have a label for each one of the scenarios explained in the **Evaluation** section of this document. Intuitively, each label represents a different traffic type (e.g., web browsing, video streaming, etc.)
    - An example of how the csv file should look like is given below. You should not include the first line in your output file. It's been added for the purpose of understanding.

flow_id	feature_1	feature_2	feature_3	feature_4	label
0	0.47	33.4	401	0.012	1
1	0.11	19.2	258	0.008	2
2	1.2	81.2	521	0.043	3

Keep in mind, that this is what you would see if you open a csv file with excel. But if you looked at it as plain text, a single one of these rows would look like this:

```
0,0.47,33.4,401,0.012,1
1,0.11,19.2,258,0.008,2
2,1.2,81.2,521,0.043,3
```

You will also see another skeleton file (ml-skeleton.py) which implements the machine learning algorithm that you will use. This is just something to start with. It is ok to just leave it as it is, or add/remove some machine learning algorithms, as long as you compare at least 3 different ones. Feel free to make any modification or change the structure of the program. These are the key-points that you will need to take care of:

- Import of the csv file into a data structure known as pandas DataFrame that is well suited to be used with Python Machine Learning library (scikit-learn).
    - In “columns_list” you will need to insert a set of strings. These are names that will be assigned to the fields of your csv document.
    - In “features” you will need to insert another set of strings. These are the actual features that will be used to feed the machine learning model. Play around with the features you want to use and see how accuracy scores change.
- The data objects “X” and “y” represents, respectively, features and labels. These objects will then be used by the “train_test_split” function to split the dataset into training set and testing set.
- Finally, the actual machine learning model is used and fed with the objects created in the previous step. These last two steps will be repeated 10 times. This is a technique known as “cross validation” which is used for evaluation of the score. The idea is that, since the training and testing set are always randomly created, some samples that bring noise might alter the final score. Therefore, by running the algorithm more times, the effect of such noise will be reduced.

Evaluation

The packet sniffer should be used to analyze the packets exchanged by the computer in which it is running. These scenarios should be considered:

1. Web browsing
2. Video streaming (e.g., YouTube)
3. Video conference (e.g., Skype)
4. File download

The minimum requirements for the training set are the following:

- 100 samples (each sample is a flow with its calculated features). More specifically, 25 samples for each scenario.
- Each sample must be generated from the analysis of at least 1000 packets.

You should clearly describe what are the flow features that you want to consider. This is the **main** creative contribution of the project.

The first part of the evaluation will consist in comparing the values of the considered features in the different scenarios. A graph for each feature should be shown, and meaningful comments should be provided.

The second part of the evaluation will consist in comparing the ability of different machine learning tools, trained using your features, in detecting what applications are running on your computer. Graphs should be produced to compare the different machine learning tools in terms of accuracy, precision & recall and F1 score.

The results should be discussed, and meaningful explanations should be given. Do not just say in words what the graph shows but discuss "why" the results look that way.

Report

A report should be prepared. It must include at least the following sections:

- Abstract: Summary the overall report
- Introduction and motivation: Overview of packet sniffers and machine learning
- Proposed features: Pseudo-code (or formula), description and rationale of the proposed features
- Implementation: Description of the methodology used for implementing the proposed solutions (e.g. relevant functions and data structures, structure of the program, etc.).
- Experiments: Description of the experiments performed and obtained results. Results should be represented in a graphical form as well as discussed in the writing.
- Conclusions: Summary of the work and final considerations.
- Provide the dataset collected for training and testing within the zip file on Canvas. This will be used for during discussion with the TA.

Optional part

The extensions to the project described below are optional. Some groups which want to extend the project can try to address the following problems. If an optional part is addressed, it should be discussed in the project report and it should be run during the meeting with the TA.

Each optional part is worth 10 points.

1) Multiple applications running concurrently. You can extend the project to be able to identify multiple applications running at the same time. In this case, you should train your machine learning models with traffic generated by more than one application running, e.g. watching a video while web browsing. The trained model should work both in the case of single and multiple applications, and should be able to tell what applications are actually running. In other words, in this case you should extend the scenarios to combination of individual application scenarios considered before.

2) Real-time identification. You can extend the project towards the realization of a tool that recognizes the application running in real time. In this case, you should train your machine learning model first, as done for the mandatory part of the project. Nevertheless, once trained, your code should continuously run while you use your computer. It should calculate the features in real time and pass them to the already trained machine learning model. Your program should then output on the shell, periodically, what type of application it thinks is currently happening on your computer.