Project

Class Machine Mining For Networks

2023-2024

Different types of attacks exist in a network, e.g. port scan, denial of services, botnets. There are two main ways to prevent them. The first one is to try to detect explicitly a virus in the payload of packets. The second one is to detect anomalous patterns of communication.

The goal of the project is to develop a method to carry out anomaly detection in IP traffic. In few words, the principle of the method is to build a profile of each IP address under the form of a small graph, called *graphlet*. We then build a model using Support Vector Machine to distinguish normal from malicious end hosts from an annotated trace. The last step will be to try to detect attack in a not annotated trace.

You have to write a **report** presenting your results and provide the code in a Python notebook.

0. Pre-requisites: Install *Scikit-Learn* `https://scikit-learn.org/` and *NetworkX* (`https://networkx.github.io/`).

1. Read the paper

   Karagiannis, T., Papagiannaki, K., Taft, N., and Faloutsos, M. (2007, April). **Profiling the end host.** In *International Conference on Passive and Active Network Measurement (PAM)* (pp. 186-196). Springer, Berlin, Heidelberg.

2. Understand the concept of *end host profile* using *graphlet*.

3. Build (by hand) the graphlets corresponding to the following network flows.

```
format: srcIP protocol dstIP sPort dPort
14.112.37.26 17 14.112.37.29 138 138
14.112.37.26 17 14.112.37.33 80 80
14.112.37.27 6 14.112.37.34 167 80
14.112.37.28 6 14.112.37.29 443 443
14.112.37.26 17 14.112.37.33 443 80
```

4. A traffic trace with annotated flows (normal or malicious) is provided in the file `annotated-trace.txt`. You will be using it to build your model of end hosts.
   Note that the srcIP and dstIP are given as simple integers and not with the traditional format with prefixes.

5. Build all the end host graphlets corresponding to flows in `annotated-trace.txt`.

We will now build the end host models using the random walk kernel, but in two ways. First, we do not use the kernel trick to avoid the mapping of the graphlets in the high dimensional space. Second, using the kernel trick.

6. Build a function to transform the graphlets into the high dimensional space of the random walk kernel (consider only walks of length 4).

7. Use a Support Vector Machine algorithm to build a model separating the normal from the malicious end hosts in the high dimensional space.

8. Use now the kernel trick to avoid having to map your graphlet in the high dimensional space. Compare the computation time.

We will now moving to the detection phase of potential attacks.

9. Apply your model to the traffic trace provided in the file `not-annotated-trace.txt`. Do you detect anomalous traffic? Plot them. To which kind of attacks, could they correspond?

10. Discuss potential false positives and false negatives.

11. Redo the modeling and detection phases using now the **ideal kernel** for subgraphs of sizes 5. Compare the two methods (resolution time, efficiency).

12. Additional question: Compare the use of the Random Walk Kernel with another learning method using Graph Neural Networks. You may use the GraphSAGE proposition described in the paper:

    Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In *Advances in neural information processing systems (NIPS)* 30 (2017).