Project 5: ML for Security–
Constructing & Evading network traffic based model of IDS

1. Introduction:

The goal of this project is to introduce students to machine learning techniques and methodologies, that help to differentiate between malicious and legitimate network traffic. In summary, the students are introduced to:

- Use a machine learning based approach to create a model that learns normal network traffic.
- Learn how to blend attack traffic, so that it resembles normal network traffic, and by-pass the learned model.

2. Readings & Resources:

This assignment relies on the following readings:

- "Anomalous Payload-based Worm Detection and Signature Generation", Ke Wang and Salvatore J.Stolfo, RAID2004
- "Polymorphic Blending Attacks", Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, Wenke Lee, Usenix Security 2006
- "True positive (true detections) and False positive (false alarms)"

3. Task A:

- **Preliminary reading:** Please refer to the above readings to learn about how the PAYL model works: a) how to extract byte frequency from the data, b) how to train the model, and c) the definition of the parameters; threshold and smoothing factor.
- **Code and data provided:** Please look at the PAYL directory, where we provide the PAYL code and data to train the model.
- **Install packages needed:** Please read the file SETUP to install packages that are needed for the code to run.
- **PAYL Code workflow:** Here is the workflow of the provided PAYL code:
  - Read in the parameters: threshold for the mahalanobis distance and smoothing factor. The parameters need to be provided by the user.
  - Read in the normal data and separate it into training and testing. 75% of the provided normal data is for training and 25% of the normal data is for testing.
  - Read in the payloads of the training data.
  - Sort the payload strings by length and generate a model for each length.
  - Each model per length is based on [`mean frequency of each ascii, standard deviation of frequencies for each ascii`]
  - Read in the payloads of the test data.
  - Test the testing data against the trained model: 1. Compute the mahalanobis distance between each test payload and the model (of the same length), and 2. Label the payload: If the mahalanobis distance is below the threshold, then accept the payload as normal traffic. Otherwise, reject it as attack traffic.
- **Select parameters and set them in the PAYL code.**
- **Run the PAYL code:** `python wrapper.py`
- **Observe the output of the PAYL code:** The code reports the true positive.

```
$ python wrapper.py
Attack data not provided, training and testing model based on pcap files in data/folder alone.
To provide attack data, run as: python wrapper.py <attack-data-filename>
-----------------------------------------------
Training
Testing
Total Number of testing samples: 7616
Percentage of True positives: XX.XX
Exiting now
```

- **Main Task: Perform experiments to select proper parameters.** Provide different parameters as input to the code, and observe the True Positive rates. Please select parameters that give you a True Positive rate of 95% or above but try to get as close to 100 as possible. Please note that it is entirely up to the student, to write her/his own wrappers around the code provided, as needed. e.g. a script to evaluate multiple parameters in parallel. Also please note that you may find multiple pairs of parameters that can achieve a TP of 95% and above.
- **Deliverable.** Please provide the threshold, the smoothing factor and the true positive rate. See Deliverables section for the format.

4. Task B:

- **Attack payload:** To download your specific attack payload, visit the following url and append your GTID at the end with pcap file extension: e.g. if your GTID is `einstein7`: http://www.cc.gatech.edu/~rgiri8/6262_P5/einstein7.pcap
- Train the model on normal data, using the parameters that you found from Task A.
- Test the attack trace (`einstein.pcap`) against the model. Verify that it gets rejected.
- You should run as follows and observe the following output:

```
$ python wrapper.py attack-trace-test
Attack data provided, as command line argument attack-trace-test
-----------------------------------------------
Training
Testing
Total Number of testing samples: 7616
Percentage of True positives: XX.XX
-----------------------------------------
Analysing attack data, of length1
No, calculated distance of ZZZ is greater than the threshold of YYY. It doesn't fit the model.
Total number of True Negatives: 100.0
Total number of False Positives: 0.0
Number of samples with same length as attack payload: 1
```

5. Task C:

(a) **Preliminary reading.** Please refer to the "Polymorphic Blending Attacks" paper. In particular, section 4.2 that describes how to evade 1-gram and the model implementation. More specifically we are focusing on the case where $m <= n$ and the substitution is one-to-many.

(b) We assume that the attacker has a specific payload (attack payload) that he would like to blend in with the normal traffic. Also, we assume that the attacker has access to one packet (artificial profile payload) that is normal and is accepted as normal by the PAYL model.

(c) The attacker's goal is to transform the byte frequency of the attack traffic so that is matches the byte frequency of the normal traffic, and thus by-pass the PAYL model.

- **Code provided:** Please look at the Polymorphic blend directory. All files (including attack payload) for this task should be in this directory.

- **How to run the code:** Run *task1.py*
- **Main function:** *task1.py* contains all the functions that are called.
- **Output:** The code should generate a new payload that can successfully by-pass the PAYL model that you have found above (using your selected parameters). The new payload (`output`) is `shellcode.bin + encrypted attack body + XOR table + padding`. Please refer to the paper for full descriptions and definitions of Shellcode, attack body, XOR table and padding. The Shellcode is provided.
- **Substitution table:** We provide the skeleton for the code needed to generate a substitution table, based on the byte frequency of attack payload and artificial profile payload. According to the paper the substitution table has to be an array of length 256. For the purpose of implementation, the substitution table can be e.g.a python dictionary table. We ask that you complete the code for the substitution function. You are free to create this table with one-to-one or one-to-many mapping as per your choice.
- **Padding:** Similarly we have provided a skeleton for the padding function and we are asking you to complete the rest.
- **Main tasks:** Please complete the code for the *substitution.py* and *padding.py*, to generate the new payload.
- **Deliverables:** Please deliver your code for the substitution and the padding, and the `ouput` of your code. Please see section deliverables.

(d) **Test your output.**

Test your output (below noted as `output`) against the PAYL model and verify that it is accepted. FP should be 100% indicating that the payload got accepted as legit, even though is malicious. You should run as follows and observe the following output:

```
$ python wrapper.py output
Attack data provided, as command line argument Output
-----------------------------------------------
Training
Testing
Total Number of testing samples: 7616
Percentage of True positives: XX.XX
-----------------------------------------
Analysing attack data, of length1
Yes, calculated distance of YYYY is lesser than the threshold of XXXX. It fits
the model.
Total number of True Negatives: 0.0
Total number of False Positives: 100.0
```

6. Deliverables & Rubric:

- **Task A: 35 points** Please report the parameters that you found in a file named `parameters`. Please report a decimal with 2 digit accuracy for each parameter.
  Format:
  |`Threshold:1.23`|
  |`SmoothingFactor:1.24`|
  |`TruePositiveRate:80.95`|
- **Task B: 5 points** Please append a new line in `parameters` with the score of the payload after completing Task B.
  Format:
  |`Distance:2000`|
- **Task C: 60 points**
  - **Codes: 40 points** Please submit the code for `substitution.py` and `padding.py`.
  - **Output: 20 points** Please submit your `output` of Task C.