Problem Solving Environments and Applications in Data Science ECE525

Report

Tuesday 17 January 2023

Fall Semester

Project 6
Comparison of Classification Techniques for Distributed Denial of Service Attack Detection

Anastasia Psarou 2860
Christos Arseniou 2730

Supervisor: Elias Houstis

1 Introduction

In this project we aim to classify DDos Attacks based on the information we are able to collect about them. Distributed Denial of Service (DDoS) attack is a menace to network security that aims at exhausting the target networks with malicious traffic. As we can see in the figure below, a DDos Attack can belong in many different classes. Designing a real-time detector with low computational overhead remains a significant concern in DDoS attack detection, despite the existence of many statistical methods. Additionally, the effectiveness of new detection algorithms and techniques depends heavily on the availability of well-designed datasets.

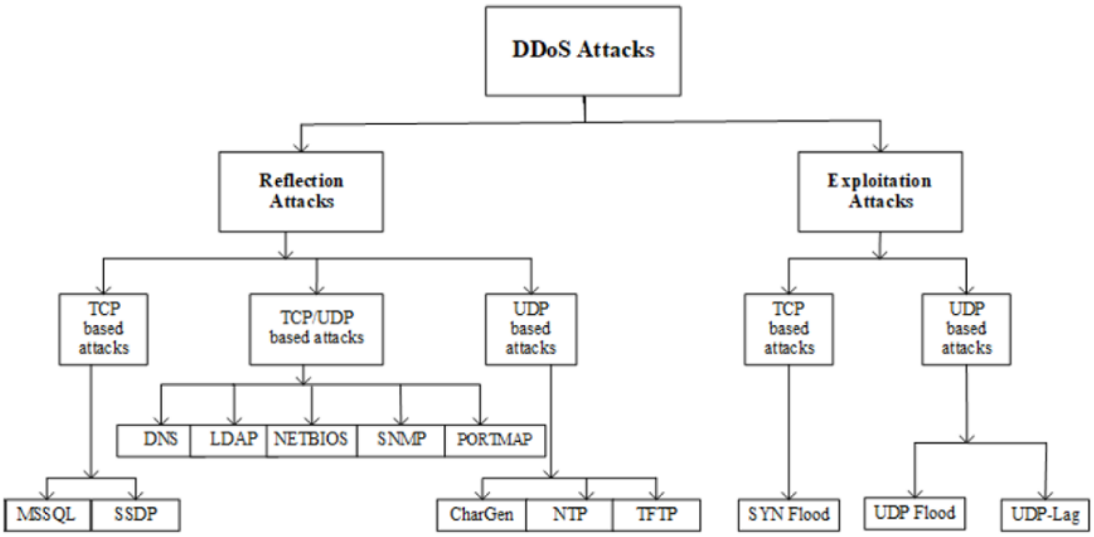

Figure 1: DDoS Attacks

2 Dataset

The dataset comprises of both benign and recent common DDoS attacks that closely resemble real-world data (PCAPs). Additionally, it includes the outcome of network traffic analysis carried out using CICFlowMeter-V3, with labeled flows based on timestamp, IPs of source and destination, ports of source and destination, protocols, and attack type. It was structured based on the architecture seen below.

The main focus when creating this dataset was to produce realistic background traffic. To achieve this, they utilized the B-Profile system (Sharafaldin et al. 2016) to analyze human interactions and generate

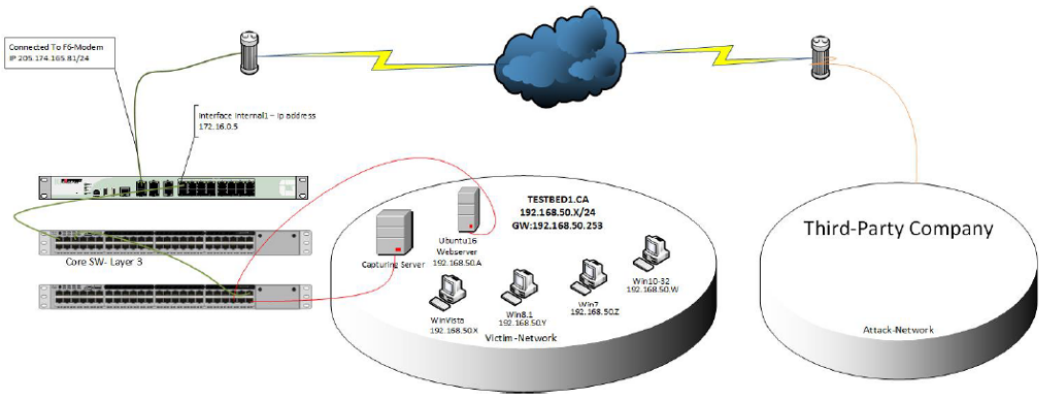

Figure 2: Testbed Architecture

natural background traffic in our test environment (as shown in Figure 2). The dataset includes the simulated behavior of 25 individuals using HTTP, HTTPS, FTP, SSH, and email protocols.

3 Preprocessing

Due to the error shown in the figure below we had to reduce the size of the datasets to execute the analysis in our systems.

```
MemoryError: Unable to allocate 1.59 GiB for an array with shape (37, 5775786) and data type int64
```

Figure 3: Memory Allocation Error

Therefore, we reduced each dataset to 20000 rows and we deleted the columns that had zero values in every cell. Lastly, we combined each dataset that referred to one attack in order to create a complete attack dataset that contain every entry. With that being done, we became ready to implete our classification methods.

4 Multi-Layer Perceptron

Multi-Layer Perceptron is a fully connected class of feedforward artificial neural network, in which the mapping between inputs and outputs is non-linear. It consists of one input and one output layer and one or more hidden layers with many neurons stacked together. Furthermore, an arbitrary activation function should be used.

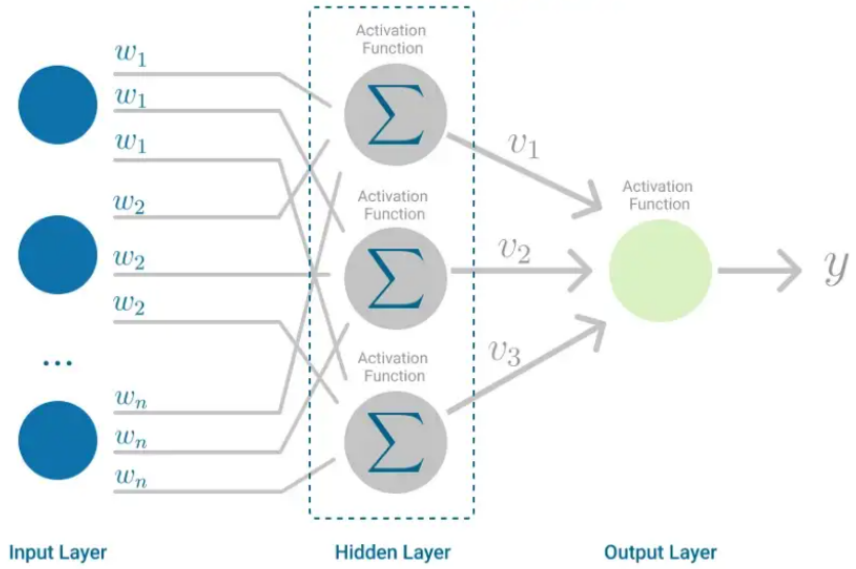

Figure 4: Multi-Layer Perceptron

The way that the algorithm works is that each layer is feeding the next one with the result of their computation and this goes all the way through the hidden layers to the output layer.

Afterwards, the backpropagation technique is being executed. That is the learning mechanism that allows the Multilayer Perceptron to iteratively adjust the weights in the network, with the goal of minimizing the cost function. In order for this to work properly, the activation function, must be differentiable.

In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient.

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn't changed more than a specified convergence threshold, compared to the previous iteration. We can notice that also in our execution, when the loss becomes

stable, the algorithm stops its execution.

```
Iteration 1, loss = 0.70526487
Iteration 2, loss = 0.47030785
Iteration 3, loss = 0.44832991
Iteration 4, loss = 0.44159180
Iteration 5, loss = 0.43732316
Iteration 6, loss = 0.43352801
Iteration 7, loss = 0.43101352
Iteration 8, loss = 0.42879998
Iteration 9, loss = 0.42824198
Iteration 10, loss = 0.42804285
Iteration 11, loss = 0.42712268
Iteration 12, loss = 0.42447760
Iteration 13, loss = 0.42473194
Iteration 14, loss = 0.42450797
Iteration 15, loss = 0.42431925
Iteration 16, loss = 0.42350683
Iteration 17, loss = 0.42348885
Iteration 18, loss = 0.42207848
Iteration 19, loss = 0.42293056
Iteration 20, loss = 0.42228115
```

Figure 5: Multi-Layer Perceptron Iterations

We decided to implement the Multi-Layer Perceptron algorithm with the use of two different resampling methods, k-fold cross validation and train-test split.

The train-test split is implemented with the use of the train_test_split() function and we notice that the algorithm is completed after 128 iterations and 0.32 loss. The accuracy achieved for this case is 0.88.

Afterwards, the k-fold cross validation method is used with k=5. That means that our data sample will be split into 5 groups. For each different group we notice different number of iterations ranging from 82 to 175. As far as the accuracy is concerned we get 0.81.

For the given data, the train-split method is better, as it achieves better accuracy and it's faster compared to the k-fold on for k=5.

5 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It is particularly effective in high-dimensional spaces and when the number of features is greater than the number of samples.

In our project, we used SVM to classify DDos attacks by training the algorithm on a labeled dataset of network traffic data. The SVM algorithm finds the best boundary, or hyperplane, that separates the different classes of data in the training set. Once the boundary is established, the algorithm can then use it to classify new, unseen data. The boundary is

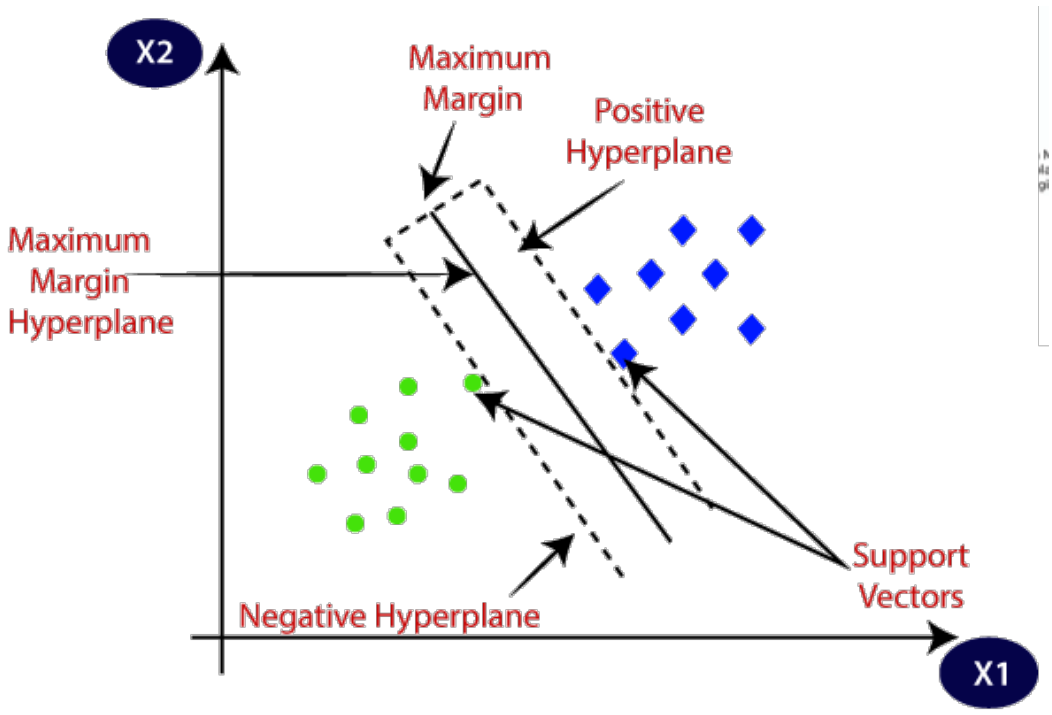

Figure 6: Support Vector Machine Implementation

chosen to maximize the margin, or distance, between the boundary and the closest data points of each class, which helps to improve the accuracy of the model. The figure below, showcases the way the hyperlane is designed.

The accuracy obtained from the SVM implementation turns out to be similar to the one we got from the MLP.