Final Project Report for ECE417
Network Intrusion Detection System with Machine Learning

Evangelou Sotirios SEVANGELOU@UTH.GR
Kalais Konstantinos KKALAIS@UTH.GR
Chatziefremidis Eleutherios ECHATZIEF@UTH.GR

Abstract

As a final Project for the ECE417 Machine Learning Course, we chose to implement a Cyber Security oriented system. Specifically, the outgrowth of this idea is a Network IDS system that analyses network packet metadata and decides on the normality of the network traffic and the type of an attack when the traffic is abnormal.

1. Introduction

1.1. The problem

The so-called fourth industrial revolution has changed the modern life as we knew it. Besides personal computers, the huge network we call Internet now includes, mobile phones, tablets, IOT devices and many more people have network access and knowledge that was considered arcane 20 years ago.
The possibilities and opportunities are limitless; unfortunately, so too are the risks and chances of malicious intrusions by people or other systems.
Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks, and the subprocess of the identification of these attacks is called Intrusion Detection.

1.2. The motivation

Intrusion Detection Systems (IDS) that monitor network traffic and anomalies are called NIDS (network intrusion detection systems) and often use an approach called anomaly-based detection which detects deviations from normal traffic behaviours.
In the past years this has been achieved by 'rules' introduced by individual engineers or 'smart' systems.
Thus, we decided to leverage the power of Machine Learning and Neural Networks in order to implement a 'smart' system that will analyse packet metadata and take decisions on the security of the network it resides in. The motivation is to contribute in the field of security and especially autonomous security, and create a system that will effectively handle the network security issues for an individual user or a whole business.

1.3. Previous Research on the issue

The use of Machine Learning techniques in developing intrusion detection systems is not a new idea.Knowing that, we decided to search for papers and research on the particular issue.
We focused on the work of two particular papers, both from research conducted by the University of New Mexico.
The first, by Mahdi Zamani and Mahnush Movahedi [1] delves into the use of Machine Learning techniques in the Intrusion Detection field of study, taking a more general approach, which familiarised us with the problem we wanted to solve.
The second paper we studied, by Mukkamala, Janoski and Sung[2] takes a closer perspective on the issue, trying to implement solutions using Support Vector Machines and Neural Network on a DARPA dataset, and comparing the two methods' results.
Based on these respected professors' work, we decided to use some of these methods in our project.

2. The Dataset

2.1. General Description

The dataset we used to train our Machine Learning models and test it, belongs to the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS)[3].
The creators of the dataset used the IXIA PerfectStorm tool for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours.
The main reason we chose the dataset was because it uses as features the fields contained in PCAP files. This means, that the system can be applied in real situations, using as input packets captured in the network with Tcpdump or

University of Thessaly, ECE 417 Course Final Project.

Wireshark, turned into PCAP files and consecutively into csv format by a simple parsing script.

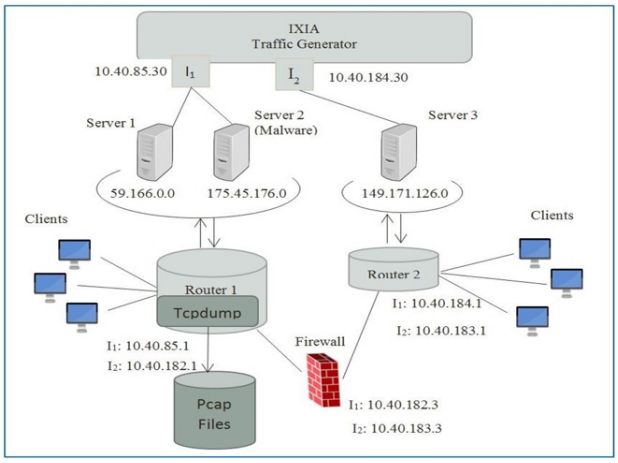

Figure 1. Testbed used to create the Dataset

2.2. Specifics of the Data

The dataset we downloaded from the UNSW website, was already split into a training set of 175341 instances and a testing set of 82332 instances. The dataset had some minor inconsistencies such as missing features that we removed during the dataset cleaning phase. The dataset consists of 43 features- fields of PCAP files including IP sources and destinations, ports, service types etc. The attack types provided as labels by the dataset (instead of normal packets) are:

- Fuzzers
- Analysis
- Backdoors
- DoS
- Exploits
- Generic
- Reconnaissance
- Shellcode
- Worms

2.3. Preprocessing

One of the main parts of Machine Learning is preprocessing, and data preparation.
In order to ensure the best performance of the algorithms we will discuss in the next section, we decided at first to find highly correlated features and drop them from the dataset. Using the FeatureSelector library by Will Koehrsen [4], we were able to drop features with at least 85% correlation, in order to make our algorithm more cost and time efficient.
Next, we use the LabelEncoder class provided by Scikit Learn library, in order to transform the nominal values to numeric and enable the algorithm to perform arithmetic operations on these values.
Lastly, we scaled our data using the MinMaxScaler class of Scikit Learn in the range of (0,1) for easier convergence. This procedure was not implemented for the SVM methodology, as SVMs are scale invariant.

3. Algorithms and Body of Work

3.1. General Approach and Algorithms used

The dataset provides two columns as targets one with normal-not normal values, and one with attack type values. We found that classifying by attack type and aggregating the results to normal and abnormal traffic gave us better results than classifying on normal/abnormal traffic. Thus, our problem is categorised as a supervised multiclass classification problem.

Based on the papers we mentioned we decided on the use of three algorithms to focus on the issue:

- Multilayer Perceptron
- Random Forest
- Support Vector Machine

We will analyse the work implemented in each one, rated by increasing performance.

3.2. Support Vector Machine

SVMs were proposed by the above-mentioned papers as the best performing Machine Learning technique for this kind of problem.
The best parameters we used, after testing several variants with trial-and-error approach, were 100 for the C value, and 0.00001 for the gamma value. We also used an RBF kernel, which is considered the most effective in general problems. Though, our dataset was very large, and comparing to the other two algorithms the time coefficient was a factor that made it not suitable enough in our opinion. Specifically, using only about 12% of the training set, the training phase lasted about 100 seconds, and the time would increase exponentially with more data as SVM tends to cross-calculate each point with the rest of the data. Comparing to the other two algorithms the time was in a much higher scale.

The experiments we conducted with SVMs used about 50.000 training points (more points would take too much time), and produced accuracy rates of 76.101% on deciding on the normality of the network traffic (Normal/Abnormal) and 68.087% accuracy in deciding the type of attack happening in the network.

```
C=100
gamma =0.00001
classifier = SVC(kernel="rbf",C=C,gamma=gamma,verbose=0)
classifier.fit(train_x, train_y)
yhat = classifier.predict(test_x)
```

Figure 2. SVM using Sklearn's SVC classifier class

3.3. Random Forest

The random forest technique was not mentioned in the researched papers, but was a technique we decided to try based on the speed of execution even on large datasets.
The parameters we used after trying several variants, were 1000 estimators and 5 as the depth of the tree. The algorithm provided us with a high accuracy result in a short time, specifically 81.336% on the binary classification problem of traffic normality, and 68.087% on the attack type classification problem. Generally our opinion was that the algorithm provided us with very accurate results in regard to how simple it was to develop and fast to execute.

```
clf = RandomForestClassifier(n_estimators=1000, max_depth=5,random_state=None)
clf.fit(train_x,train_y)
yhat = clf.predict(test_x)
```

Figure 3. Random Forest using Sklearn's RandomForestClassifier class

3.4. Multilayer Perceptron

Using Multilayer Perceptron provided us with the best results of the three algorithms. For the neural network's architecture we used:

- A Dense input layer of 300 neurons and rectifier linear unit activation function
- A Dense hidden layer of 150 neurons and rectifier linear unit activation function.
- A Dense output layer of 10 output classes with softmax activation function.

The model was compiled with Adam optimizer and a categorical cross-entropy loss function, and trained in batches of 150 instances for 150 iterations.

The results we got were a problem-high accuracy of 83.96% on the normality classification and 82.81% on the attack type classification.

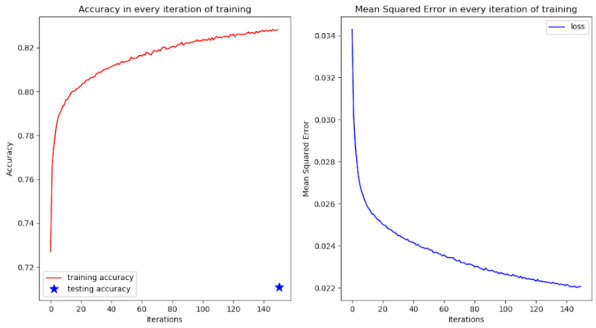

Figure 4. Plots of the Accuracy of atk type and the Mean Square Error by Iteration

For this problem it is significant that we have as less attacks classified as normal as we can. For this reason we also introduced a confusion matrix metric, that, as we can see provided as with only about 1.4% false positive classifications.

	T	F
T	24517	12483
F	1184	44148

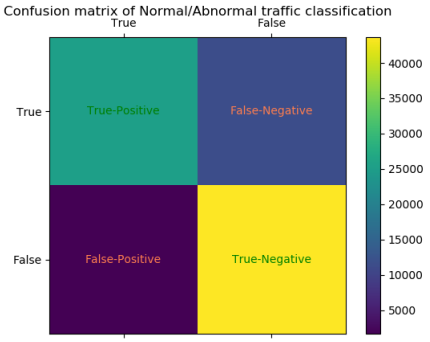

Figure 5. Confusion Matrix heat map

```
model = Sequential()
model.add(Dense(300, input_dim=trainx.shape[1], activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['mse','accuracy'])
```

Figure 6. MLP model using Keras, Tensorflow's high-level API

3.5. Methodology Comparisons

Below, an accuracy-based comparison between the three algorithms is provided, in order to give a better look into which algorithm is deemed more suitable.

Table 1. Accuracy of every algorithm used in the NIDS project

TESTS	NORMAL/ABNORMAL CLASSIFICATION	ATTACK TYPE CLASSIFICATION
MLP	83.96%	82.81%
RF	81.33%	68.08%
SVM	76.10%	68.08%

From the table, it is obvious that Multilayer Perceptron produces the better results of the three, and we also provide a bar plot to visually confirm that MLP performed better.

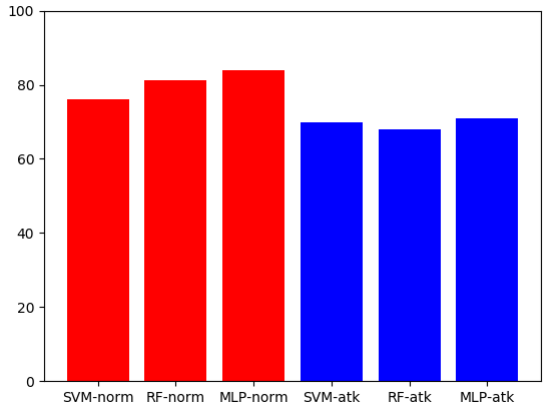

Figure 7. Bar plot for accuracy comparison between each algorithm

4. Downloading and Running the project

The NIDS project is hosted on the Github software development platform, and specifically in https://github.com/EvangelouSotiris/NIDS_Project_CE417. Instructions to download dependencies, clone and run successfully the project are included in the README.md file.

5. Thanks to

The project was implemented in the spectrum of the Machine Learning ECE417 course of the university of Thessaly, Electronic and Computer Engineering Department.
The professor and supervisor of the project, prof. E.N. Houstis guided us to the majority of the knowledge, tools and skills needed for this purpose. We would like to thank him.

6. References

References

[1] Mahdi Zamani, Mahnush Movahedi *Machine Learning Techniques for Intrusion Detection*. University of New Mexico, 2013.

[2] Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung. *Intrusion Detection: Support Vector Machines and Neural Networks*. New Mexico Institute of Mining and Technology, 2002.

[3] UNSW-NB15 Dataset, ACCS https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[4] Feature Selector library : Will Koehrsen https://github.com/WillKoehrsen/feature-selector