Intrusion Detection System Using Machine Learning (Interim Report)

Vasisht Duddu
2015137
IIIT-Delhi, India
vasisht15137@iiitd.ac.in

Shubham Khanna
2015179
IIIT-Delhi, India
shubham15179@iiitd.ac.in

Anubhav Jain
2015129
IIIT-Delhi, India
anubhav15129@iiitd.ac.in

1. Introduction

With increasing complexity of systems, the threats to these systems are increasing exponentially. Attackers are resorting to more sophisticated malware to avoid detection mechanisms and exploit systems of critical importance. Attackers and malware writers keep coming up with new sophisticated malware like polymorphic and metamorphic malwares which are capable of changing their code dynamically. This creates a need to use advanced defence mechanisms to deal with unknown problems. Present malware detection schemes use heuristics and rely on hard coded features for detecting malware.

With the growth of machine learning, the detection can be improved by training the ML model to learn from past data an predict/classify files as malware or benign. This approach allows to protect against attacks which may not have been seen before.

We explored this idea and used machine learning to detect malware in Windows PE32 executable files. We trained multiple models on the data set and optimized it for best possible generalization. Accuracy in intrusion detection systems are not the only metric to be concerned about and we worked on exploring various other metric to improve the model.

2. Related Work

Machine Learning has been explored for malware and network anomaly detection and researchers have used various algorithms and feature extraction methods to improve the performance of the model.

The table shows the state of art work on using machine learning models for malware classification of PE32 executable file.

Source	Model	Accuracy	False Positives
Tek[1]	Random Forest	99.35	0.56
Adobe*[2]	Random Forest	98.21	6.7

* Data set used is very extensive and contains signatures and samples of polymorphic malware

Following are some of the work done by researchers on malware classification:

- Mining n-grams (Siddiqui et al.)[5] gave a 94% accuracy
- Multiple algorithms (Schultz et al.)[6] gave a 97.76% accuracy
- Multiple algorithms by extracting 189 features (Shafiq et al.)[7] gave a 99% accuracy
- Association mining (Ye et al.)[8] gave a 92% accuracy
- SVM on program strings (Ye et al.)[9] gave a 93.8% accuracy

3. Data set and Evaluation

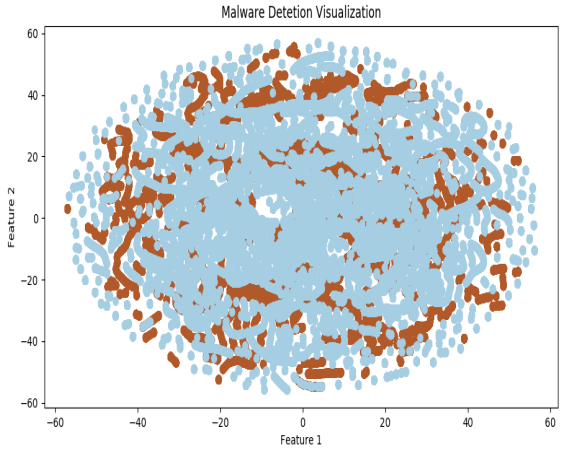

Figure 1. Visualization of data set using TSNE: Red showing malware samples and Blue showing benign samples

The total data has 138047 samples with 54 features. We divided the data into training and testing in the ratio 70:30. We further used randomization for better validation and uniformity of data samples used to train data. Figure 1 shows the data set visualized using TSNE and it is clearly non-separable. The table gives a summary about the information of the data.

Total Size	138047
Training Size	96632
Testing Size	41415
Features	54
After Extraction	14

3.1. Feature Extraction

We used feature selection to reduce the 54 features to a smaller set of feature to get the most relevant features for differentiating benign binaries from malware.

For extracting the features, we can manually plot histograms for each feature and check if it can help to classify the result. Another method is to remove each feature and see the effect on the outcome.

For this dataset, we used the Tree-based feature selection for extracting the relevant features.

3.2. Evaluation Metrics

In intrusion detection systems, the accuracy is not a sufficient measure of the performance of the model. The false positives and false negatives also play an important role. It is important to reduce the false positive values to as low as possible (ideally 0). This requires to compute the confusion matrix for finding the false positive and false negatives.

The false positives and false negatives are measured using specificity and sensitivity. We need to train the model in a way that we get a balance between the false positive and false negatives. This balance can be obtained by adjusting the classification threshold which can be selected from the ROC curve.

For a large number of malicious classified values, even 1% false positives will result in large number of false alarms to the network administrator which we want to minimize. This will defeat the purpose of the intrusion detection system.

4. Analysis and Progress

The main objective was to get a good accuracy with a very low false positive rate. Reducing the false positives for malware detection is of critical importance for an efficient IDS.

We initially used svm, logistic regression, random forest, gradient boosted trees and adaboost classifiers. Svm did not give result and required a lot of computation. Adaboost did not perform well and hence we did not consider it. Even though adaboost performed better than logistic regression we tried to tune logistic regression to get better results. Based on results, we observed that ensemble based approaches gave better results than other classifiers. Random forest, Adaboost and Gradient boosted trees are all based on decision tree classifiers which act as weak classifiers. Hence, using decision tree classifiers may also yield good results but not comparable to the ensemble methods.

Cross validation was computationally very heavy and hence we preferred to use randomization of data set instead. For tuning we used grid search to get the best parameters and used the best parameters to further train the model. We then evaluated various metrics and compared them before and after the tuning. We calculated false positives from confusion matrix and AUC score from ROC curve for comparing the improvement of the models after tuning and get an idea of how good the model is. We tried to maximize accuracy while trying to keep the above in mind. For logistic regression, we plotted the probabilities of output classes and tried to manually select the threshold for classification and find a balance between sensitivity and specificity. However, this did not yield any major improvement in the model and we preferred ensemble approaches.

We plotted the learning curves for further analysis and they have been shown in Figure 2 and Figure 3.

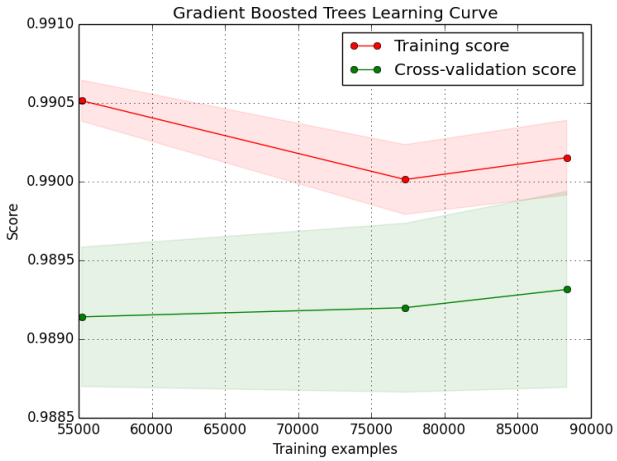

Figure 2. Learning Curve of Gradient Boosted Classifier

5. Results

The table gives a summary of the results obtained.

Model	Accuracy	False Positives	AUC Score
Logistic Regression	30.06	1	0.5
Random Forest	98.22	0.91	0.987
Gradient Boosted Trees	98.45	0.71	0.986

We obtained results very close to Teks results as shown above. We have a lower classification accuracy and a higher

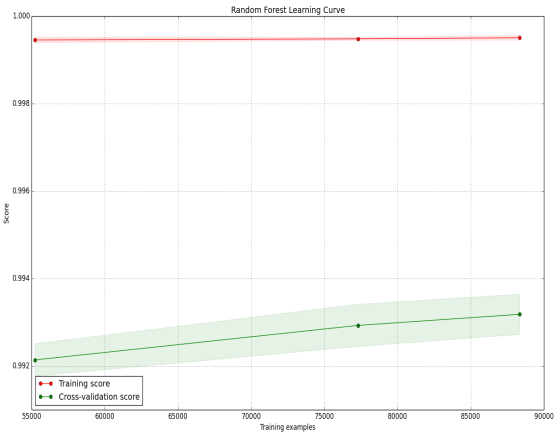

Figure 3. Learning Curve of Gradient Boosted Classifier

false positive rate than the state of the art results. One possible reason is that the features extracted are different. We used randomization of data set and parameter tuning which to improve the performance. Tek claimed random forest to be more optimal. However, according to our analysis, gradient boosted trees seemed to work better with a lower false positives. Hence, we prefer to use gradient boosted trees over random forest for the given data.

We used logistic regression and due to its bad performance, we ruled out other linear classifiers as they may not be optimal for this data set. All the ensemble approaches used use decision trees as the weak classifier. Hence, decision trees may give fairly descent results as compared to logistic regression. AUC scores of random forest and gradient boosted trees tell that both the trained models are equally good but the false positive rate is lower for gradient boosted trees and accuracy is higher.

6. Future Work

We will explore machine learning aldorithms specifically neural networks on a new data set for network anomaly detection. The data set is UNSW-NB15 data set[4]. The data set is very large with 175,341 records and the testing set is 82,332 records from the different types, attack and normal and 49 features.

We will use neural networks and optimize them to classify the network traffic as being malicious or benign and corresponding to different types of attacks.

Since it is part of IDS, we propose to use confusion matrix to calculate the specificity and sensitivity, and try to find the right balance between them. We will use ROC curve to compute AUC score to determine how good different models are in comparison to each other. We will also consider classification accuracy to determine the optimal model.

For analysis, we will use grid search to tune the model and obtain the best possible parameters for optimal performance. We will explore cross-validation and data set randomization for further improving the performance. For neural network we will evaluate the performance for different sizes and architectures and compare them.

Following are the roles each member have followed and will follow for the rest of the project duration and evaluation:

- Anubhav Jain: Data visualization , pre-processing and outlier removal
- Vasisht Duddu: Model and parameter selection, feature Extraction, training and analysis
- Shubham Khanna: Parameter tuning, model improvement and analysis

References

[1] Machine learning for malware detection: https://www.randhome.io/blog/2016/07/16/machine-learning-for-malware-detection/

[2] Towards Classification of Polymorphic Malware: https://www.blackhat.com/docs/webcast/TowardsClassificationofPolymorphicMalware-Final.pdf

[3] Malware data set: https://github.com/Te-k/malware-classification/blob/master/data.csv

[4] UNSW-NB15 data set: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-data sets/

[5] M. Siddiqui, M. C. Wang, and J. Lee. Detecting trojans using data mining techniques. In D. M. A. Hussain, A. Q. K. Rajput, B. S. Chowdhry, and Q. Gee

[6] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables.

[7] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq. Pe-miner: Mining structural information to detect malicious executables in realtime.

[8] Y. Ye, L. Chen, D. Wang, T. Li, Q. Jiang, and M. Zhao. Sbmds: an interpretable string based malware detection system using svm ensemble with bagging.

[9] Y. Ye, D. Wang, T. Li, and Ye. Imds: Intelligent malware detection system.