Classification of Firewall Logs at University Network

Dalya Manatova
Security Informatics, Luddy School of Informatics, Computing, and Engineering
Indiana University
Bloomington, USA
dmanato@iu.edu

***Abstract*—This is a report for Applied Machine Learning Class (CSCI 556) at Indiana University. The project goal is to implement machine learning algorithms for real-world tasks. Firewall is the most critical part of the network which should not be conflicting with the security policies used which should not trigger security vulnerabilities. However, managing firewall rules has been difficult, challenging and error-prone. In this experiment, we have suggested a multi-classifier approach using nine models, using prepossessing techniques to handle imbalance and high variety of categories.**

I. Introduction

Network traffic generates excessive amounts of data in every corporate network, and tracking such traffic can be a hectic task. Although many companies develop numerous products to help manage traffic, some specifics are still overlooked when a product is designed to help manage broader and generalized network traffic. Since the amount and insights of data produced by users in the network highly depended on the type of corporation and size of a network, it is essential to look at network traffic management in the corporation context.

Firewall devices monitor traffic (in and out) in a network and can either allow or prevent traffic according to the pre-specified rules and policies. These rules and policies can be tuned according to the generated data in an ongoing process. Since some firewalls also provide visibility into the source and type of traffic coming, it is possible to alter rule set and understand the traffic through logs of a firewall. Such logs provide organizations with information about the source and destination IP addresses, protocols, and port numbers and are usually used to investigate attacks on the network. Correct tuning and configuration for such rules prevent future attacks on the system and significantly influence the organization's communication network. [1]
Many operating systems and network equipment, such as firewalls, generate large volumes of network data in the form of logs and alarms. These log files can be used for network control and debugging purposes. Logging protection-related or debug material, such as logging error logging and failed authentication, is an essential log file feature. BY expecting and analyzing them, administering and managing the firewall becomes less costly.

This project aims to predict if the traffic malicious/suspicious, based on firewall log. Identify essential attributes that should be tracked cautiously to catch suspicious traffic.

II. Related Work

Finding data on actual organizational firewall logs is difficult, since most companies don't feel comfortable sharing it due to security and privacy issues. This project is focusing on firewalls logs in academic settings, ie. University network traffic. The data that is used in this project is published by Firat University on the UCI Machine Learning Repository. [1] F. Ertam and M. Kaya from Firat University in 2018 published a paper associated with this data, where they used SVM for classification of traffic in 4 classes of firewall decision actions, and the F1 score for their method didn't appear above 0.76. [2] In this paper I was able to achieve better result with specific data preprocessing that improved the result of prediction. It's difficult to compare the methods since authors hardly explained their procedures and pipelines for data transformation.

In another study by S.R.T.M. University, 500 000 instances created by the firewall were examined using six features (I.P., Protocols, Port numbers). Firewall log dataset is evaluated and features are implemented into machine learning classifiers like Naive Bayes, kNN, One R, and J48 using the Spark in Weka platform. Authors, we able to reach F1 0.998 on their dataset. [3] This means that the potentially resulting dataset used for this paper could be improved from the originally published results.

III. Methodology

A. Dataset

The data set is 65 532 records with 12 attributes. The amount of records was produced with 30 seconds of firewall work.
In order to classify the firewall log data, only 4 of the attributes in the dataset were selected. The original date included seven more attributes, such as data size and session time, that are presumably affected by the label outcome. Such features won't help in prediction, as if the firewall denies the connection, the data size and session time won't have a considerable value.

[1]UCI Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data

Table I shows the descriptions of attributes in the dataset, and Table II shows the selected features for classifier. Table III gives explanations for four classes that are targets for prediction.

TABLE I
ORIGINAL DATASET

Feature	Description
Source Port	Client Source Port
Destination Port	Client Destination Port
NAT Source	Network Address Translation Source
NAT Destination	Network Address Translation Destination
Bytes	Total Bytes
Bytes Sent	Bytes Sent
Bytes Received	Bytes Received
Elapsed Time (sec)	Elapsed Time for flow
Packets	Total number of Packets
pkts sent	Packets Sent
pkts received	Packets Received
Allow	Label Classes (allow, deny, drop, reset-both)

TABLE II
FEATURES

Feature	Description
Source Port	Client Source Port
Destination Port	Client Destination Port
NAT Source	Network Address Translation Source
NAT Destination	Network Address Translation Destination

TABLE III
TARGET LABEL

Class	Description
Allow	Allows the internet traffic
Deny	Client Destination Port
Drop	Silently drops the traffic
Reset	Occurs as an error and send response to two ends devices.

B. Preprocessing

All our features are categorical features with numerous unique values. Presumably, there are almost 65 thousand port numbers and thousands of addresses for sender and receiver. Thus all features are converted to the binary matrix (dummy variable matrix) with only 1 and 0 for each unique value in the dataset. This, however, leads to overload for processing the data and OneHotEncoding is used with sparse storage.

Typically firewalls allow most of the traffic, and in some rare, suspicious cases can deny/drop the packet of data to go through. This leads to an imbalance of class representation in the logs. As this can then lead to the model overfitting and predicting the 'allow' class much better than others, we have to make some transformations. It is crucial since predicting the traffic that shouldn't be going through the firewall is explicitly preferred over 'allowed traffic' since preventing the malicious and suspicious connection is significantly important. One of the approach that could be used is over-sampling. Oversampling creates copies of existing samples or adds more samples to the minority class. Though it increases the training set size, thus consume extra time. The popular method to increase the size of the minority class without a high risk of overfitting is the Synthetic Minority Over-sampling technique (SMOTE), which creates synthetic samples from the minority class. The SMOTE samples are linear combinations of two similar samples from the minority class. [4] Fig 1 shows the distribution of classes before and after applying the SMOTE technique.

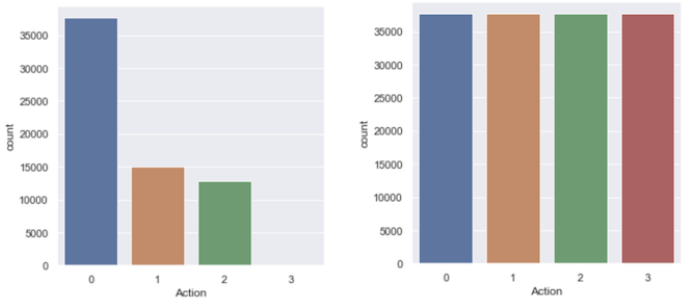

Fig. 1. Classes distribution before and after SMOTE

Note that classes also encode from strings to number labels using Label Encoding, since for some classification models it's required.
Dataset is split by for training and testing with the ration 0.33.

C. Classification Phase

Since data generated by firewalls is usually massive, running a machine learning algorithm on at least one-day data would be problematic and costly. Therefore, it is important to pick the algorithms that will provide results and converge in a reasonable amount of time with the increased dimensionality of using one-hot encoding and increased size of data using SMOTE.
Our problem is multiclass prediction, and this paper will compare the performance of three models: Support Vector Machine (SVM), K-Nearest Classifier (KNN), and Random Forest. The combination of using transformed data and over-sampled data for different models for performance comparison is also computed.

A justification for using SVM is that it usually performs well in a high-dimensional space. Additionally, since our features are basically binary values, we assume the data is linearly separable. However, it requires more time to converge as data size grows since it computes the distances for each point (training complexity $O(n^2)$)).
KNN is robust to noisy training data and is useful in the case of a large number of training examples, so it's also assumed to fit our problem. Although the time also increases as data size grows, better than SVM (training complexity $O(nd)$).
Random Forest is expected to show a good performance regardless of any transformation since categorical features are not a problem for decision trees. This algorithm can accommodate high-dimensional spaces as well as extensive training samples.

IV. Results and Discussion

A. Results

The obtained Precision, Recall and F1 Score values of the classifiers are shown in Table . Note that our case's preferred metric would be Precision ($= \frac{TP}{TP+FP}$) since we aim to minimize False Positives (classified as 'allowed' traffic but shouldn't be). We also would consider macro metrics instead of micro for each specific class since we aim to compute the metric independently for each class and then take the average (treating all classes equally). Table IV on the next page shows all test results for all model combinations.

The linear kernel was used for all SVM models, except for Basic Classifier (without preprocessing). For the basic classifier, we used RBF, assuming that data won't be linearly separable.

As we can see from the results, the best model turns out to be Random Forest (with 10 decision trees), and F1 doesn't change much with preprocessing since Random forest is robust to categorical data complication and imbalance. Although Random Forest shows a good result, SVM improved significantly with SMOTE and OneHotEncoding. Confusion Matrix is shown on Fig 2 and 3.

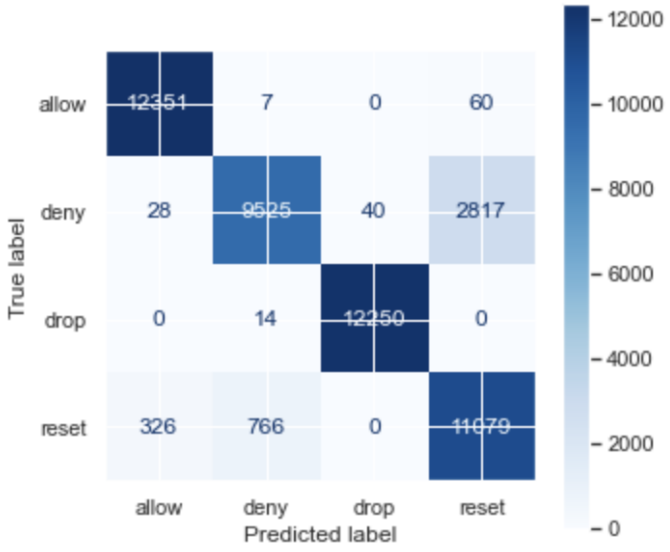

Fig. 2. Confusion Matrix for SVM (SMOTE and OneHotEncoding)

B. Future work

Firewall is the most critical part of the network which should not be conflicting with the security policies used which should not trigger security vulnerabilities. However, managing firewall rules has been difficult, challenging and error-prone. In this experiment, we have suggested a multiclassifier approach using nine models.

We performed our experiment on the firewall dataset generated at University Environment. We considered only the four primary attributes: Source port, Destination port, Source NAT, Destination NAT. The action attribute with "Allow", "Drop",

Fig. 3. Confusion Matrix for SVM (Without prepossessing)

"Deny" and "Reset" has been selected as a class attribute. The experiment results showed that the proposed method is more accurate as compare to currently used method on this dataset. It was observed that the highest Precision value was obtained in the SVM (linear) classifier using OneHotCoding with 0.934 and Random Forest (10 trees) with 0.999. F1-score values observed that the best result was for the same classifiers, but an additional step of preprocessing (SMOTE) with 0.917 and 0.989, respectively.

Future research can be conducted on the networks of Indiana University and on the bigger dataset. Also, the unseen test data should be utilized to check the real performance of the model.

References

[1] E. Ucar and E. Ozhan, *The Analysis of Firewall Policy Through Machine Learning and Data Mining,* Wirel. Pers. Commun., vol. 96, no. 2, pp. 2891–2909, Sep. 2017.

[2] F. Ertam and M. Kaya, *Classification of firewall log files with multiclass support vector machine,* 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, 2018, pp. 1-4, doi: 10.1109/ISDFS.2018.8355382.

[3] H. As-Suhbani, S.D. Khamitkar, *Classification of Firewall Logs Using Supervised Machine Learning Algorithms,* International Journal of Computer Sciences and Engineering, Vol.7, Issue.8, pp.301-304, 2019.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *SMOTE: synthetic minority over-sampling technique,* J. Artif. Int. Res. 16, 1 (January 2002), 321–357, 2002.

[5] N. Singh, D. Tomar, and B.N. Roy, *An Approach to Understand the End User Behavior through Log Analysis,* International Journal of Computer Applications, 5, 27-34, 2010

TABLE IV
RESULTS

		SMOTE and One-hot encoding	One-hot encoding only	Basic Classifier
SVM (liner, rbf for basic)	**Recall**	0.918	0.859	0.732
	Precision	*0.922*	*0.934*	*0.724*
	F1 Score	0.917	0.887	0.728
KNN (n_neighb = 5)	**Recall**	0.855	0.821	0.742
	Precision	*0.876*	*0.931*	*0.736*
	F1 Score	0.848	0.856	0.739
Random Forest (n_samples = 10)	**Recall**	0.989	0.95	0.925
	Precision	*0.989*	*0.999*	*0.999*
	F1 Score	0.989	0.972	0.955