Anomaly Detection Project

(Problem Statement 2)

Objective:

Utilizing data analysis and machine learning techniques to examine firewall logs and identity potential security incidents such as error, threats and suspicious activities.

Data:

For this project, I have used dataset from Kaggle:

<mailto:https://www.kaggle.com/datasets/tunguz/internet-firewall-data-set>

Data Description:

The data consists of 65532 observations and 12 features in total.

A screenshot of a computer

AI-generated content may be incorrect.

Step 1 – Exploratory Analysis (Light EDA)

This section consists of a light exploratory analysis done on the data, which includes looking at the type of data, any missing values or outliers and basic spread/distribution of all the features. The table above shows the data types for all columns. As can be seen from the non-null count column, none of the data columns have any missing values. The data consists of a mix of numerical and categorical variables. The source and destination ports have also been considered as categorical variables for the purpose of this project.

The table below shows a basic summary of the numerical features of the data.

A screenshot of a graph

AI-generated content may be incorrect.

Looking at the summary and comparing the IQR and maximum values, we can clearly see the presence of outliers in almost all of the features. These outliers might correspond to some known threat or a low-risk event, which has been looked at a later stage of this project. To visualize these features and see the patterns more clearly, it might help transforming the variables which will help treat the outliers. The features were transformed using the formula log(1+x), as there are features with zero value.

The charts below show the spread of bytes and packets for all the records in the data. (For detailed charts of all the features, please refer to the python script).

A graph of a log-transformed packet

AI-generated content may be incorrect.A graph of a log-transformed byte

AI-generated content may be incorrect.

Both the features seemed to have a skewed distribution, with most of the entries have low bytes or packets with very few having extremely high values.

Before fitting any machine learning model, it is important to check if there is any

##### multicollinearity present between the features in the data which might affect the results. The chart below shows the correlation matrix for the three features – Bytes, packets and elapsed time. The variable bytes is equivalent to the sum of bytes sent and bytes received, which guarantees a perfect or nearly perfect correlation between bytes and both bytes sent/bytes received. This is similar for the variable packets. Thus, it is better to check for multicollinearity by including either the components or the totals for both these variable types.

A red and blue squares with white text

AI-generated content may be incorrect.

##### Here we can see that the bytes and packets variables are highly correlated to each other. The variable for time elapsed does not seem to be correlated to any of the two variables. We can check for feature importance and try to improve the model fit by using either of the bytes or packets variable as well.

The charts below show the top 10 ports for each of the categorical features in the data.

A group of blue and white bars

AI-generated content may be incorrect.

The most interesting one is the plot for NAT Source port, which is 0 for most of the entries in the data.

#### The last feature is ‘Action’, which tells us what decision was made by a network security device in response to the traffic flow or connection attempt. So, if the Action column has a value - allow, the traffic was permitted and if the value is deny, then it was blocked. But can we surely say that the traffic which was blocked by the network security was actually malicious or suspicious? We need to validate this with some quick analysis first.

A screenshot of a computer screen

AI-generated content may be incorrect.

#### From the table we can see:

* Denied, dropped or reset traffic is extremely minimal. Only 1 packet, minimal data exchange and nearly zero duration.
* Allowed traffic is dramatically different with high bytes transferred, many packets and long session durations.

Thus, we can use 'Action' as a target variable here to fit a supervised machine learning model.

#### The problem involves detecting threats or blocked traffic, hence it makes sense to group together deny, drop and reset-both as they all indicate that the system rejected or blocked a connection. This would help make the model training easier, evaluation more meaningful and results more interpretable.

Step 2 – Train/Test Split

Before fitting the model, the data was split into training and testing datasets. 70% of the data was treated as training data and the rest 30% as the test set. The target variable was the feature – ‘Action’.

Step 3 – Deeper EDA (using training data set only)

#### This stage involved looking at the relationship of the target variable with all the other features in the data.

The charts below show the spread of the features ‘Bytes’ and ‘Packets’ for each class of the target variable. It looks like most of the entries that were denied had very small bytes or packet sizes, whereas the allowed entries do have some outliers as well.

A graph of a person with histogram

AI-generated content may be incorrect.A graph of a person with a beard

AI-generated content may be incorrect.

Looking at the spread of the top 20 source ports, it seems like they are almost evenly spread across both the classes of the target variable.

A graph of a number and a number

AI-generated content may be incorrect.

Step 4 – Modelling

To build a reliable threat detection system, I have developed both supervised and unsupervised machine learning models.

* The supervised machine learning model uses labeled data – with ‘Action’ as the target/response variable to learn patterns of known threats like intrusions or malware communications. It is precise and efficient at catching threats we have already seen.
* The unsupervised model however does not rely on labels. It flags anomalies based on unusual behaviour, even if they have never been seen before.

I looked at the current problem this way – the supervised model acts as a trained security guard here, who knows what a suspicious intruder looks like. The unsupervised model is analogous to motion sensors, they don’t know who is good or bad but alert you whenever something’s out of the ordinary. Relying on just might make us miss out on some threats or get too many false alarms. Together, they are much stronger.

Step 4.1 – Supervised Modelling

To create a supervised machine learning model with ‘Action’ as the target variable, I have chosen to use both random forest and gradient boosting algorithms. These algorithms work well with classification problems like this one.

Below are the results of the random forest model:

A blue squares with white text

AI-generated content may be incorrect.

Model accuracy is 99.98%, with only 4 misclassified points. From our exploratory data analysis, we did conclude that packets and bytes variables are correlated and one of them can be removed based on the feature importance given by the random forest model.

A graph with blue squares

AI-generated content may be incorrect.

From the plot above, we can see that the elapsed time variable is the most significant one here, followed by the NAT Source and destination ports. Out of the bytes and packets variables, packets clearly seems to be more significant. The model could be improved further by considering just one out of the two variables. The source and destination ports along with the packets and bytes sent seem to be the most insignificant ones.

Gradient boosting model was also fit to this data. Below are the results of that model:

A screenshot of a graph

AI-generated content may be incorrect.

Accuracy here is 99.97% with 5 points misclassified as an anomaly. This is a little surprising for me, as I have mostly seen the gradient boosting model performing better than random forest model, even if slightly. This might be because there are some redundant and uninformative variables in the data, as can be seen from the feature importance plot. There’s a possibility of high noise in the data as well.

Step 4.2 – Unsupervised Modelling

To capture unknown threats, I have built Isolation Forest and Local Outlier Factor (LOF) model which are known to be good at anomaly detection. The target variable ‘Action’ was obviously excluded from the model. The contamination rate for these models was chosen to be equal to the ratio of the two classes ‘Allow’ and ‘Deny’ from the action variable in the training data (from the supervised model).

The accuracy of the isolation forest was 35% and LOF was a little higher, around 41%. I understand the accuracy rates are quite low here, but we need to keep in mind here that the ‘Deny’ class includes all the three different types – deny, drop and reset-both. All the three types might not be a threat. These models also predicted some points under the ‘Allow’ class as a threat. These are mostly because of the suspiciously large sizes of the bytes or packets or longer elapsed time. However, the model accuracy could have been improved again here by doing feature reduction and some hyperparameter tuning.

Step 5 – Combining the Results

I have created a risk score for each of the points in the data based on the predictions of the supervised and unsupervised models.

* High risk – predicted as anomalies by both model types
* Known Attack – predicted as an anomaly only by the supervised models
* Potential Threat – predicted as anomaly only by the unsupervised models
* Normal

The distribution of points based on risk label in the test data can be seen from the plot below:

A graph of a risk assessment

AI-generated content may be incorrect.

Step 6 – Creating a dashboard

I have used the plots from exploratory analysis and model results to create a dashboard that displays the incident summaries for different timeframes. The data used did not have a timestamp column, so to achieve the purpose I have simulated a time frame for the data points.

The dashboard has been created using the package ‘streamlit’ in python.

Further work

* Model performances can be improved using feature selection and hyperparameter tuning, especially the unsupervised models.
* Multi-class models using ‘Action’ variable can be fitted to see how results change, again specially for unsupervised models.
* Additional models like LightGBM can be fitted for supervised modelling and One-Class SVM or HDBSCAN can be used for unsupervised modelling.