Anomaly Detection - Nucleon

Danielle Ben Bashat (ID. 204212757) Adiel Matuf (ID. 307895268)

Submitted as final project report for Projects with the industry workshop, IDC, 2021

1 Introduction

1.1 Motivation and problem scope

Identification of malicious events in the network traffic data is required in order to protect the computer networks that are often vulnerable to cyberattacks. A reliable Network Intrusion Detection System (NIDS) should be robust enough so that the novel threads could be identified effectively and also detect the malicious events efficiently so it can be applied to large-scale live streaming data.

Identification of potential intrusions includes Signature-based IDS solutions and Anomaly-based IDS solutions. Signature Detection uses a library of known threats to identify them, enable achieving a high threat detection rate with low false positives because all alerts are based on the detection of known malicious content. Nevertheless, this method is limited to detecting known threats and can't address zero-day vulnerabilities. Anomaly Detection methods learn a model of 'normal' behavior and report on any deviations. These solutions can detect novel or zero-day threats, however, at the expense of accuracy.

Anomalies in network traffic are the patterns in data that deviate significantly from the expected normal behavior. In order to develop Anomaly-based IDS, various approaches are utilized which include: Rule-Based - the system detects anomalies based on a manually given set of rules; Supervised Learning - machine learning algorithms are trained on a labeled dataset to learn a prediction model; Unsupervised Learning - machine learning algorithms learn to identify anomalies from an unlabeled dataset. This project will address malware detection within the wide network, delivering a passive monitoring solution, an intrusion detection system (IDS). Our goal is to use the data collected through Nucleon's polymorphic sensors (constantly changing environments, so the attackers wouldn't detect it) and develop a new unsupervised anomaly detection algorithm. We hope the algorithm will detect suspicious network behavior, identify and will able to stop hidden and massive fast replicating cyber attacks on the network.

1.2 Related Works

The model Isolation-Forest(IF) first in introduced by Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou [4] a model-based method that explicitly isolates anomalies instead of profiles normal points. Based on their model developed new model that aim to use for intrusion detection systems to protect the computer networks from cyberattacks which is unsupervised machine learning approach that combines the K-Means algorithm with the Isolation Forest for anomaly detection [3]. Despite the former's there are also work in a field of Deep Neural Networks (DNNs). Therefore Gabriel C. Fernandez and Shouhuai Xu [2] elevate the power of Auto-encoders to build such anomaly detection system which can be an effective approach.

2 Solution

2.1 General approach

We build an unsupervised anomaly detection model for malware attacks which will be integrated in the Nucleon detection product. The model will be the first one in the Nucleon product, hence, we will try to build a baseline model which will be improved over time by the company. The model should generalize a normal behaviour of clients packets and detect abnormal behaviour. We transform the raw data to help the model to generalize a normal behavior efficiently and effectively. We develop several models and deliver the one with best performance. In addition, we implement a pipeline that Nucleon can reconstruct for a large scale.

2.2 Data set

The dataset we used was collected from Nucleon's sensors which consist of packets data from the Internet. The instances are logged with timestamps and labeled as normal and malicious (packets that are related to cyberattacks). The data labeling was done by Nucleon's cyber researchers. The dataset contains 1,648,904 instances with 14 fields (described in Table 1.). The time range is from 05/07/2021 16:34:03 to 02/08/2021 14:43:48. Most of the data were labeled as normal packets when only 1,075 instances labeled as attacks - 0.065%.
The main challenges we have dealt with include the following:

1. Due to the massive amount of the data it's impossible to label each packet by experts. Hence, we tackle the problem in an unsupervised manner.
2. Handling imbalanced distribution of normal and malicious data and the sparse occurrence of the malicious data. When choosing the metric to evaluate the models we need to take it into account.
3. The data consists of many valuable features which are nominal variables with high cardinality.
4. Time series data: to prevent data leakage, the train and test subsets are split by date and not in a random order and we implemented nested cross validation. Moreover, we extracted feature aggregations for enriching with more structured and informative features.

Table 1: Description of the dataset attributes

No.	Attribute	Description	Type	Uniques
1	_id	global unique id. (MongoDB _id)	Categorical	1,648,904
2	ts	timestamps of the packet in the sensor.	Timestamp	1,648,904
3	session_id	a unique number that a Web site's server assigns a specific user for the duration of that user's visit (session).	Categorical	1,604,008
4	sensor_id	each sensor of Nucleon has a specific id which is static and not dynamic (as IPs).	Categorical	867
5	protocol	a protocol from the transport layer, tcp or udp.	Categorical	1
6	port	a destination port. (range 0 to 65535)	Categorical	10
7	ip	a source ip, where the packet came from.	Categorical	6517
8	raw_data	the raw data that was sent.	Categorical	141,706
9	data	the data that was sent encrypted.	Categorical	141,706
10	country	a country code of the source ip. (US, IT, CN etc.)	Categorical	107
11	public_proxy	whether the ip came from public proxy or not.	Categorical - Boolean	2
12	tor	whether the ip came from Tor or not.	Categorical - Boolean	2
13	packet_size	the size of the packet.	Continous	N/A
14	identify	normal/ malicious labels	Categorical - Boolean	2

2.3 Design

2.3.1 Data Preparation: Exploration, Transformation and Feature-Engineering

2.3.1.1 Handling Null Values

We handling null values as follow- 'session_id', 'port', 'ip', 'country': all null values get encoded by there frequency of null in the train set (according to the features); 'sensor_id': shouldn't be null because it's represent the sensor of Nucleon and always should be filled (which sensor record the packet) hence if null we drop it; 'ts': is a timestamp when the packet is recorded hence if null then we drop it. 'packet_size': null packet size will filled with 0; 'tor': will be filled with 0 (not from tor); 'identify': will be filled as not identify as malicious.

2.3.1.2 Feature Engineering

For extracting more valuable information and ease the model learning we had done some feature aggregations to capture additional information from the unstructured data. We calculated several statistics based on the historical data. For each session id, we aggregated on the packet size of the current instance and the previous instances (based on the timestamp). We calculated the total packet size sum, the average, the standard deviation, the minimum value of packet size and maximum value of packet size. In addition, we calculated an accumulated count of the previous instances for the same session id, and for each IP, an accumulated count of the previous instances for the same IP and on the same day.

In addition, we pre-processed the value of the timestamp 'ts' as the following equation from the paper [3]: $T_v = D * 24 + H$.

2.3.1.3 Feature Encoding

Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical values of the relevant features into numerical representation. We experimented different encoding techniques. The different encoding methods we implemented and for each nominal variable are described in Table 2 (below).

Table 2: Description encoding methods for categorical variable

Attribute	Encoding Method
country	1. Frequency based label encoding. 2. Dummy encoding more 10 most common labels and 'Other' group.
sensor_id	Frequency based label encoding
ip	1. Dummy bits encoding (16 bit). 2. As done by the paper [3].
port	1. Dummy bits encoding - Ipv4 (32 bits). 2. Frequency-based label encoding.
tor	Dummy encoding
identify	Dummy encoding

* Frequency based label encoding: numerical conversion based on the labels frequency in the train (past).

2.3.1.4 Feature Scaling

For the Auto-Encoder, we normalized and scaled the continuous variables. We used the standard min-max scaling, this normalization method scales the data to [0,1] range as follows: for feature X, we define: $X_{norm} = (X - X_{min})/(X_{max} - X_{min})$

2.3.2 Research Design

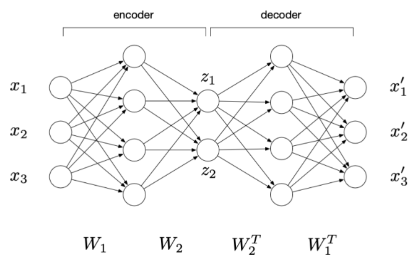
The research design flow included: Feature Selection process, Nested Cross Validation for hyper-parameters tuning, and the final models comparisons. The model algorithms we used were Isolation Forest, Isolation Forest-KMeans and AutoEncoder. The models were trained on clean train subset without malicious data and were evaluated on test subset included the malicious and non-malicious data.

2.3.2.1 Autoencoder

As in [2], we use Auto-Encoder for unsupervised network anomaly detection.

Auto-Encoder learns a compressed representation of the input data. The model outputs a reconstruction of the input data. The assumption is that with a malicious data flow the reconstruction error will be high compared to normal data flow.

Figure 1: Auto-Encoder NN architecture [2]



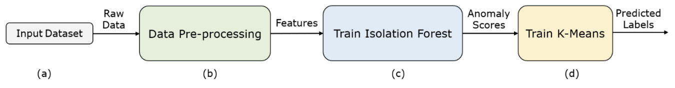
2.3.2.2 Isolation Forest, Isolation Forest-KMeans, and IF-using Robust Z-score Models

Isolation Forest (IF) [4] is an unsupervised algorithm for detecting anomalous data. Isolation Forest assigns an anomaly score based on distance from the root node. At the basis of the Isolation Forest algorithm, there is the tendency of anomalous instances in a dataset to be easier to separate from the rest of the sample (isolate), compared to normal points.

As mentioned in [3], IF have the following fundamental issues: it requires information about the ratio of anomalies (i.e., contamination ratio) in the training data which may require the construction of a manually labeled training dataset - a very time-consuming process in big data scenarios. In addition, it needs to find out a threshold to convert the anomaly scores predicted by the model to different labels - a tedious process that becomes more expensive in large datasets. We want to deliver a scalable approach that fits industry large network traffic data in real-time both effectively and efficiently and not only within the research academic framework and that will easily suits for Nucleon's deployment. Therefore, we try two approaches to automate and optimize the classification threshold: K-Means and using robust Z-score method.

Isolation Forest-KMeans (IF-KMeans) [3] cluster the IF model output score to two clusters, normal and anomaly.

Figure 2: IF-KMeans Full Pipeline [3]



Robust Z-Score - a method which is not dependent on the number of observations. Instead of using the mean and standard deviation, we use the median and the deviation from the median. When instead of using standard deviation, it uses the MAD (Median Absolute Deviation). $MAD = median\{\|x_i - \widetilde{x}\|\}$

Then, we calculate the modified Z-score like this: $M_i = \frac{0.6745(x_i - \widetilde{x})}{MAD}$.

2.3.2.3 Feature Selection

For removing redundant and irrelevant features, we started with Exploratory Data Analysis (EDA) to analyze the variable distributions and their correlations and associations (for the categorical variables). The features we intentionally excluded were: 'Protocol' - it had only one value in the dataset (tcp); 'Public Proxy' - it was found to be identical to 'Tor' feature; 'agg_session_id_packet_size_min' and 'agg_session_id_packet_size_mean' - were strongly high correlated to other features; 'data' and the 'raw_data' - due to their very high cardinality, to support more effective and efficient learning.

After the initial exclusion, we experimented with different feature combination sets as well as comparing the different encoding methods for the categorical features: 'country', 'ip' and 'port' as described above in Table 2.

We ran the following trials:

1. Experiments 1-3: Country, IP, Port
   For each experiment, we compared the performance without the feature vs. the two different encoding methods per feature. We had 3X3 total running trials.

2. Inclusion Features
   Here, we experimented if the features we originally had from the dataset, and the new extracted features are improving the performance or not. (Excluding 'packet_size' since it was already selected to use for the learning.)
   For each feature*, we compared the performance with the feature vs. without it. We added the feature to the best-selected features from experiments 1-3 and with 'packet_size'.
   * ['sensor_id', 'time_val', 'tor', 'agg_session_id_row_count', 'agg_session_id_packet_size_sum', 'agg_session_id_packet_size_max', 'agg_session_id_packet_size_std', 'agg_ip_row_count']

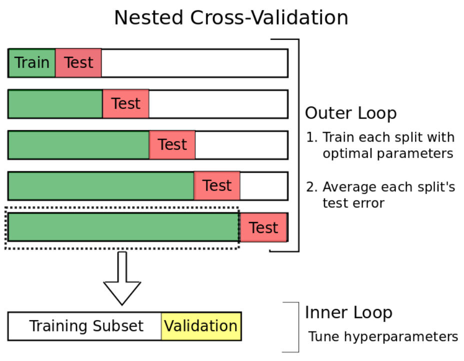
Finally, we outputted the features that increased the performance metric when added. We ran the Feature Selection process for the Isolation Forest and for Auto-Encoder separately. The final features that were selected for Isolation Forest: 'packet_size', 'country', 'ip', 'sensor_id', 'time_val', 'tor'; and for Auto-Encoders: 'packet_size', 'country', 'ip', 'sensor_id', 'time_val', 'tor', 'agg_session_id_row_count', 'agg_session_id_packet_size_sum', 'agg_session_id_packet_size_max'.

An interesting finding was that although 'time_val', 'tor', 'agg_session_id_row_count', 'agg_session_id_packet_size_sum', 'agg_session_id_packet_size_max' were found to improve the AUC score as well as the number of identified anomalies when each of the features was added separately to the base features, when the model was trained on this features all together - although the AUC score increased, the amount of TP significantly decreased for Isolation Forest. Therefore, we decided to exclude the aggregation features for IF. For the Auto-Encoder, those features are significantly improved the performance.

2.3.2.4 Model Tuning: Hyper-Parameters Optimization and Nested Cross-Validation

Data Segregation:
We split subsets of data to train the models and to validate the performance on unseen data using time indices for the split as we were dealing with time-series data. Therefore, in order to prevent data leakage, the test subset should come after the train subset in the timeline. While our problem is unsupervised we tested the models on a labeled dataset. We did so for measuring the performance and the quality of the pattern prediction. For reducing training time we implemented a random search on the hyper-parameters space. For each parameter, we sampled with replacement. For Isolation Forest tuning, we iterated through 8 different parameters combinations and for each combination we trained on 4 different Train-Val datasets and averaged the results. As per the Auto-Encoder, we iterated through 36 different parameters combinations.

Figure 3: Nasted Cross Validation [1]



Isolation Forest (IF), IF-KMeans: The hyper-parameters we tuned were: 'n_estimators', 'max_samples', 'contamination', 'max_features', and 'bootstrap'. The best hyper-parametrs results were: {'n_estimators': 300, 'max_samples': 'auto', 'contamination': 'auto', 'max_features': 8, 'bootstrap': False, 'n_jobs': -1, 'random_state': 100}

Auto Encoder (AE): The hyper-parameters we tuned were: 'optimizer', 'loss', 'architecture' (size and #neurons) and 'learning_rate'. The best hyper-parametrs results were: {'optimizer': 'rmsprop', 'loss': 'mae', 'metrics': ['acc'], 'architecture': [#features, 256, 36, 16, 8, 16, 36, 256, #features], 'learning_rate': 0.001 }

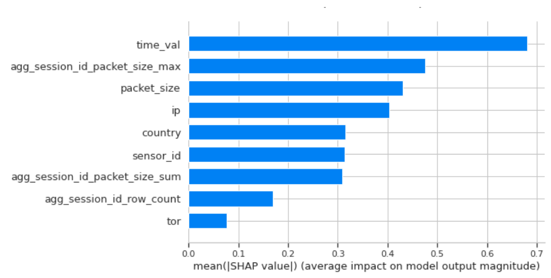
Evaluation Metrics: since the anomalous instance quantity is way lower than normal instances, Accuracy metric won't reflect the actual performance we care to measure. Therefore, we computed a confusion matrix which gave more insight into the types of the models' errors being made, i.e. FP, FN. Then, from the confusion matrix, we calculated precision, recall, F-score. We used the AUC metric score for the final selections during our trials (Feature Selection, Hyper-Parameters tuning, and Best Model Evaluation). Because we are evaluating an unsupervised method, it makes sense to use a classification metric that are not dependent on the prediction threshold to give an estimate of the quality of scoring. Moreover, we tuned the threshold using the K-Means layer in IF-KMeans and the Robust Z-score method, which isn't depend on the number of observations because it uses Mean Absolute Deviation (MAD method), as for IF and as for the AE in further steps. It should be pointed out that for Nucleon use case, predicting a non-anomalous example as anomalous will do almost no harm, especially as they are planning to develop an external API service to analyze the model results, but missing an anomalous can cause significant damage. Therefore, the amount of FN has a higher weight than FP.

3 Final Experimental Results

3.1 Models Output Interpretation

SHAP (SHapley Additive exPlanations) Values: based on Shapley values, a concept coming from game theory, allow to explain the output of machine learning models. We used the SHAP values to interpret the Isolation Forest's feature importance (figure 4) and for individuals prediction explanation (in figures 5-6). Figures 5-6, show how the values of each feature impact the model output from a base values. Features increasing the prediction value are shown in red, those decreasing the prediction value are in blue.

Figure 4: Feature Importance based on Shap Values



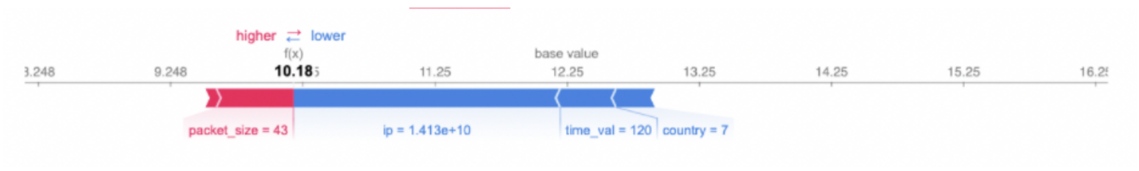


Figure 5: Instance predicted incorrectly as anomaly-(FP)

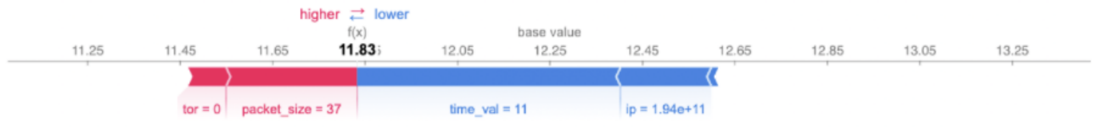


Figure 6: Instance predicted correctly as anomaly-(TP)

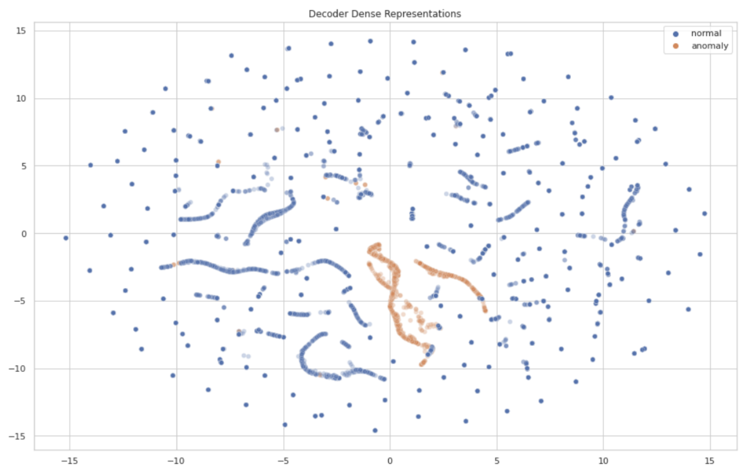


Figure 7: The compression representation of the data learned by the AE (encoder) using T-SNE

Figure 7 visualizes the compression representation of the data learned by the AutoEncoder. We used T-SNE (t-Distributed Stochastic Neighbor Embedding), a dataset decomposition technique to reduce the dimensions of the data.

3.2 Final Results

After the Feature Selection and the Nested Cross Validation processes, we trained the models with the best hyper-parameters and best features in respect to each model and compared the results which can be seen in Table 3.

Table 3: Best Models Results

Model	AUC	TPR (TP/TP+FN)	FPR (FP/FP+TN)	Accuracy	Average Time*
IF	0.89	0.96	0.144	0.856	± 2 Sec
AE	0.935	0.96	0.089	0.9111	± 3 min

*for around 1,400,000 samples

IF-KMeans and IF-Robust Z-score:

The IF-KMeans average training time was ± 1 min and its performance wasn't good comparing to the classic IF and therefore, we chose to exclude this model. In addition, we have tried the IF with the Robust Z-score method to better optimize the classification threshold. However, even iterating through different Z-score thresholds didn't converge to a better solution.

In conclusion, one can see that the Auto-Encoder is better than the Isolation-Forest. The AUC score and the FPR are better, when the TPR is equal for both. Yet we can see that Auto-Encoder training is longer comparing to IF.

4 Discussion

In this project, we have shown that both of the algorithms (IF and AE) indeed were able to detect almost all the anomalies. For True Positive Rate and AUC both of the algorithms had similar scores. However, as for False Positive Rate the AE was found to be much more accurate. Therefore, the AE is the best approach for our problem, in terms of accuracy performance. Nonetheless, for the IF, training times duration are lower. Therefore, for computational efficiency, the IF is a better approach.

5 What you have learned

While we were working on this project, we were faced a real challenging problem which gave us an experience of working with Time-Series datasets and how to approach to an Anomaly-Detection problem in general. Moreover, we have learnt to lead a research project from requirements stage to a fully developed solution. During the time we worked with Nucleon, we experienced how a project can start from some general big idea to real practice. Beside the knowledge that we gained during the project about machine learning and deep learning, we also got a much better sense of the Cyber-Security domain and its challenges. Last, we have learned some new techniques to encode categorical features, including based on their distribution in the population (frequency) and how to enrich the dataset with additional features.

6 Potential future work

1. Developing an AutoEncoder with adding Entity Embedding layers for the high cardinality features as done in the paper[2]. Moreover, an additional interesting approach can be inputting the latent space representation into the Isolation Forest model and compare the results.

2. Try using clustering methods (like KMeans, ScanDB etc.) and analyze if malicious transactions could be clustered in specific clusters (one or more).

3. Improving our results with the same direction. By creating new features from the same data especially frequency based (as we saw that those kind of features were very helpful to anomaly detection models). IF focusing on Auto-Encoders, experiment with additional architectures and functions to detect outliers.

7 Code

link: https://gitlab.com/Nuc1eonCyb3r/data-science/nabu
Note - the repository is private and the data is located in Nucleon MongoDB or in S3, therefore to access it please contact to Nucleon.

References

[1] C. Cochrane. Time series nested cross-validation, 2018. Medium, May 19, 2018.

[2] Gabriel C. Fernandez and Shouhuai Xu. A case study on using deep learning for network intrusion detection, 2019.

[3] Md Tahmid Rahman Laskar, Jimmy Huang, Vladan Smetana, Chris Stewart, Kees Pouw, Aijun An, Stephen Chan, and Lei Liu. Extending isolation forest for anomaly detection in big data via k-means, 2021.

[4] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.