Unsupervised Anomaly Detection of IoT DDoS Attacks using Deep Auto-encoders

Mack Preston, Mohammad Ariful Islam, Monzurul Amin, Faizan Ali
Department of Computer Science
Dalhousie University
{Mack.Preston,M.ArifulIslam,mh705769,Faizan.Ali}@dal.ca

Abstract

While the introduction of IoT (Internet of Things) devices promises greater automation and productivity, it also introduces new security risks to critical control systems composed of unsafe devices and machines. Host-centric IT security solutions such as networks, antivirus, and software patching are ineffective in preventing the security of IoT devices from getting compromised. This project presents a network-centric, behavior-learning-based, anomaly detection approach to secure such vulnerable environments. We detect different DDoS attacks in real-time using unsupervised machine learning (ML) and demonstrate that the predictability of benign TCP traffic, from IoT devices, can be used to detect anomalous attack traffic.

1 Introduction

We are on the verge of an industrial revolution. Often referred to as "Industry 4.0", it is the future of industrial manufacturing based on the Internet of Things or IoT. Greater control of physical devices, machines, manufacturing processes, and supply chain promises to usher in a new era of unprecedented levels of automation and productivity. Increasing the integration and distribution of devices also exposes critical resources such as industrial control systems. This could have dire consequences ranging from disruption of digital services to sabotaging nuclear reactors, oil rigs, and power grids.

Securing IoT devices with host-centric IT security solutions such as anti-virus, encryption, and software patching is a difficult task as these solutions are heavy burdens for restricted devices. Also device manufacturers often cannot be trusted to prioritize security over cost and functionality. Another problem is that unlike servers that are regularly updated to the latest version to keep up with the technology curve, IoT devices are rarely updated, making them easy targets for attackers. In light of these challenges, networks need to play a bigger role in IoT security.

The legacy anomaly detection solutions rely on the already present attack signatures. Recently, there is a growing interest in machine learning approaches to detect such attacks. Again, the supervised learning techniques utilized (for example [3]) require labeled data. This limits the ability of the detector to identify new types of attacks.

Unsupervised learning is a promising solution to IoT network security, as it learns from unlabeled data. Also as IoT traffic, although can be composed of diverse traffic between different types of devices, is highly predictable [14] and its signatures can be learned by unsupervised learning model. Our approach aims to take advantage of this by learning a representation of the normal/benign traffic in order to detect malicious traffic as anomalous.

2 Related Work

Traditional intrusion detection systems are rule-based and require traffic signatures that are well known [1]. Hence, there have been statistical and machine learning approaches that made important roads in this area as well [2, 4, 5]. Like other fields, such as computer vision and natural language processing, machine learning approaches have been effective for the two keys aspects of network security, automated and effective pattern learning (attack detection). Support model learning with deep learning auto-encoders [6] and its linear equivalents, principal component analysis (PCA), are some of the commonly used methods for the purpose [3]. Another important technique is classification in which there are many kinds of methods, including random forests, nearest neighbors to K, support vector machines, decision trees, artificial neural networks [8], and clustering [7]. Usually, feature engineering methods work on un-tagged data but developing machine learning models for classification often involves supervised learning from labeled data. Especially in the context of safety, classification task is often training on both normal and attack data to build models [8, 3].

This project deviates from the existing works in these ways:

1. Machine learning is used to learn the normal traffic and then attack traffic is detected.
2. Dimension reduction tools are used, not for feature engineering but attack detection.

This project is inspired by the unsupervised approach presented by Bhatia et al. [13]. The authors devise the similar pipeline but use smaller number of features that are extracted directly from packet headers. Instead, our project introduces the use of flow generators like Argus and Tranalyzer as a pre-processing step to encode useful sequence information into the features.

2.1 Synthetic IoT Traffic Dataset

For our experiments, we used a synthetic public data set for IoT traffic [9]. This data-set consists of packets captured from 34 Devices for 23 days in 2018, 4 days in May, 11 days in June, and 8 days in October. A variety of device types are used, including energy management controllers, cameras and health-monitoring devices. All devices are connected to a TPLink Archer C7 v2 WiFi access point, which acts as a gateway to the public internet. This gateway can capture packets, execute scripts, and transfer traffic traces to USB Storage. Devices generate 12 to 36K total TCP flows with remote hosts every day.

The data-set is broken into two halves, the first of which consists of purely benign normal traffic. The second half consists of benign traffic as well as a variety of artificial DDoS attacks. The authors of the data-set launched a number of different attacks at varying packet-per-second intensities. The attack types are as follows:

1. TCP attacks
2. ArpSpoof
3. TcpSyn
4. PingOfDeath
5. UdpDevice
6. TcpSynReflection

Due to poor initial performance, we reduced our attack detection scope to purely TCP flows and attacks.

2.2 Deep Auto-encoders

Auto-encoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. A deep auto-encoder is composed of two symmetrical deep-belief networks that usually have three or four shallow layers of neurons each representing the encoding and decoding part of the network structure. The encoder maps the input to the compressed vector and the decoder maps the compressed vector to reconstruct the input.

These models and variants like de-noising auto-encoders are often used for problems like image classification, but this work will instead use them for anomaly detection. After training the auto-encoder on normal traffic behaviour, we can detect when the traffic strays from the latent representation.

3 Methodology and Architecture

3.1 Overview

This section outlines our approach and describes some of our efforts to troubleshoot model performance and fine-tune hyper-parameters to fit our problem. Figure 1 provides a high-level overview of our proposed architecture. The implementation steps are as follows:

1. Traffic sequence information is extracted from raw packet captures using Flow Generators.
2. Standard pre-processing and normalization is applied prior to training.
3. Unsupervised models (PCA and Auto-encoder) are trained on benign traffic.
4. Models are used to compute re-construction loss for benign and attack flows from the test data-set.
5. An ROC curve is used to compute an optimal attack loss threshold to classify benign and attack flows.
6. Security alerts are raised if a specified number of attack flows occur within a sliding time window (this component is proposed but has not yet been implemented at time of writing).

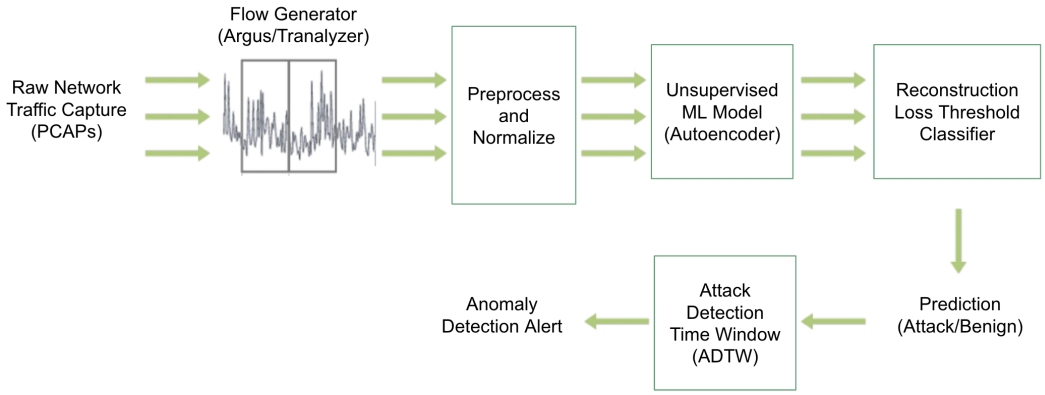

Figure 1: Proposed architecture

3.2 Preprocessing

3.2.1 Flow Generation

As mentioned in section 2.1, the traffic data-set consists of raw network traffic PCAP files. Although other works use information directly from the packet headers, our solution employs flow generators to extract meaningful sequence information as a pre-processing step. In network traffic, sequence information can provide useful insights into the behaviour of the communicating devices. In many DDoS attacks for example, a SYN packet is sent to establish a connection but the handshake is never completed. The SYN packet could be identical for benign and attack traffic, but the attack could be identified by adding the sequence context.

We duplicated our experiments using two different flow generators (Tranalyzer [10] and Argus [11]). Both tools use five tuples (protocol, src/dst IP, src/dst Port) to differentiate between the different flows. They also output different sets of features that can be used in our model. Tranalyzer was used to extract approximately 100 features from the PCAP files, while Argus yielded around 60. The commands used to generate the flows can be found in our GitHub repository [12]. All non-TCP traffic was filtered out so we could focus our model on detecting TCP attacks.

3.2.2 Flow Annotation

The public IoT data-set also includes information about the attacks that were conducted. The following masks were applied to the flows in order to label the malicious flows as attack flows:

1. Source and destination for IP and port match a known artificial attack
2. The flow falls within a known attack window for that IP/port

$$(attack.start\_time <= flow.start\_time)\&(flow.end\_time <= attack.end\_time)$$

According to the condition, any flow that falls within an attack window is labelled an attack flow. This is naive filter that could cause false positives in the test set. Previous works [3] did not encounter this problem since they operated on raw traffic that could be filtered at a finer-grained level. This could account for miss-classified flows discussed in section 4. Future works could investigate methods of annotating the raw traffic prior to flow generation to avoid this issue. Our labelling and filtering of the flows is available on GitHub in the t2_flow and argus_flow Jupyter notebooks [12].

3.2.3 Feature Selection

We used the following techniques to select appropriate features.

1. Filtered out low information features (features with a single value)
2. Filtered out non-numerical and categorical features (these should be re-visited in future iterations)
3. Used histograms to visualize feature distribution differences between attack and benign traffic (a subset of these visualizations are shown in the appendix, but the rest can be seen in the feature_hists notebook on GitHub).

Our initial implementation used 66 features from Tranalyzer, but resulted in poor performance. After reviewing the histograms of the attack and benign flows for each features, we identified that many features had distributions that looked identical between the two classes. After removing those and other uninformative features, we were left with 18 features that drastically improved the performance of the auto-encoders. Using a similar method, we are able to identify and fix an error in our attack labelling logic.

3.2.4 Normalization

We experimented with using scaling and normalization as pre-processing steps for the input of our auto-encoders. We also experimented using L1 and L2 normalization but found the resulting models to have a similar performance. Our final solution used L2 normalization on all features. One caveat of our approach was that the model normalizer was fit to only the benign traffic, meaning the normalizer would use only the max and min values from the benign data-set to scale the values. The pre-fitted normalizer was then applied to the mixed (attack and benign) flow data-set before feeding them into the auto-encoder for testing.

3.3 Model Architecture

Our auto-encoder implementation consists of three encoder layers and three decoder layers (including the input and output layers). This can be seen in figure 2. As described previously, our auto-encoder uses a bottleneck to force the network to generalize and create an abstract latent representation of the benign traffic is training on. In this model, The layers are chosen from 2 to the power -n $[n\ R]$ for the dimension of the input of each layer. We made the layer sizes dynamic in order to accommodate rapid experimentation on varying sizes of feature sets. We tried using more layers and different ratios, but found this configuration gave better results.

3.4 Loss and Activation Functions

We initially selected BCE (Binary Cross Entropy) as the loss function for our auto-encoder because its known to be good for anomaly detection. BCE loss has a requirement where the input must be

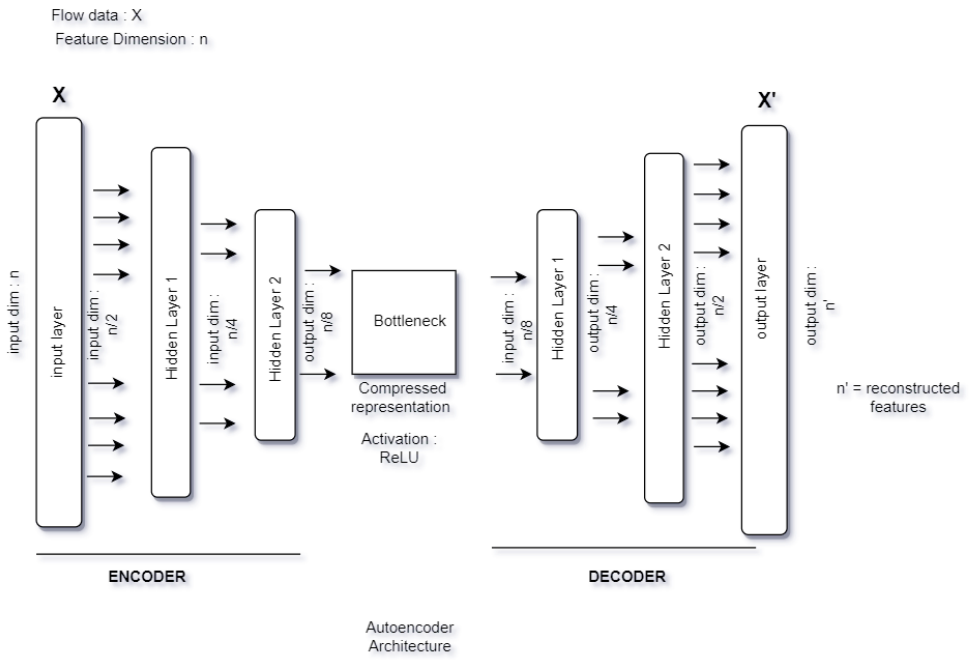

Figure 2: Auto-encoder network architecture

between zero and one. To support this requirement we were forced to use sigmoid activation functions, which ensured the output of each layer was in the desired range. In order to switch our activation function to the state-of-the-art Rectified Linear Unit (ReLU), we had to change our loss function to MSE. The ReLU activation function produces outputs from 0 to infinity, but MSE is capable of operating in that range. We also experimented with L1 loss, but got an inferior performance.

The Adams optimizer is used to optimize the model. It chooses the dynamic learning rate for our network parameters (layers, neurons). It also helps at the beginning of the training where loss is higher. It assigns learning rate per distribution value per epoch at that stage which ultimately speeds up the overall learning process.

3.5 Model Training

The auto-encoder learns the latent representation of the benign traffic by minimizing the Reconstruction Error RE(X, X'). This forces the model to generalize the representation it uses to encode and decode the input X to produce X'. We trained the model using a mini-batch approach, computing the gradients of the reconstruction error for 256 data points per batch. We experimented with smaller batches like 32 and 128, but found 256 was more efficient and had comparable results. For each training epoch, we did a full pass of the benign data-set. Future work may include training on a subset of this data or introducing noise into the input to create better generalization and prevent over-fitting. One concern is that the training high concentrations of identical flows that could bias the learning.

The training loss of the Tranalyzer auto-encoder can be seen in figure 3. The loss starts out fairly low already compared to a usual training curve. We think this can be attributed to normalization and the high concentration of zero values for some features. This can be seen in the pre-normalized histograms of the feature distributions in the appendix and GitHub. The auto-encoder consistently converges around epoch 10. To confirm the model is learning, we compared the re-construction loss distributions of the training data-set before and after training the auto-encoder. As shown in the appendix, the loss distribution moves towards zero after training. The final distribution post-training often showed a tight grouping around zero and another tight grouping around 0.06. This could mean

that the model is stuck in a local-minima and is unable to accurately reconstruct all of the benign traffic. A similar tight group appears in our test set loss distribution for benign flows.

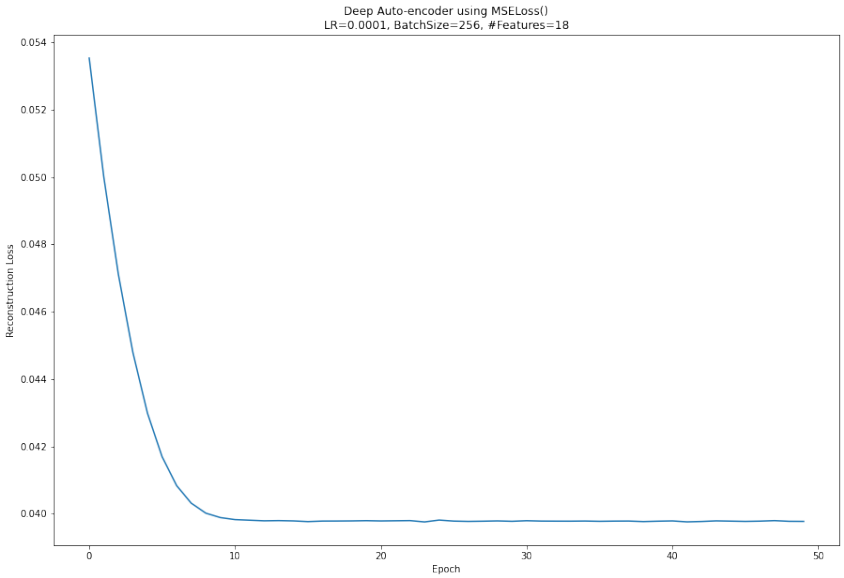

Figure 3: MSE reconstruction loss convergence

3.6 Attack Threshold Classifier

To convert our trained unsupervised auto-encoder into a classifier, we found a reconstruction loss threshold value using an ROC curve. This process is discuss further in section 4.1.

3.7 Attack Detection Time Windows (ADTW)

As the final step, the alert is generated if an anomaly is being detected. As we will see in the results section 4, there might be false positives when it comes to labeling the flow as an attack or a benign flow based on the output of the machine learning model. Hence, generating an alert for each attack flow would cause false alarms that are not desired. Therefore, to mitigate that, an attack detection time window is defined. In this, 4 flows are taken at a time and then analyzed. If some percentage (in our case 50%) of flows are attack flows, an alert is generated for attack detection.

4 Experimental Procedures and Results

4.1 Procedure

For our supervised learning baseline experiments, the models were trained on a stratified 67% train/ 33% test split of the labelled mixed flow data-set. The decision tree model was given a maximum depth of 3 due to the size of the data-set. For the same reason, a simple linear kernel was used for the SVM model. 10-fold cross-validation was employed for both models to evaluate the models performance on unseen data.

For our unsupervised experiments, our models were trained on the entire benign flow data-set, and tested against the entire mixed flow data-set. Each model was then used to compute the reconstruction loss for each of the labelled flows in the mixed data-set. To find the optimal attack threshold value, an ROC curve was constructed from the resulting loss values. Figure 4 shows a cross-hair of the selected threshold value based on optimal TP and FP rates for our Tranalyzer auto-encoder model. Figure 5 shows that threshold super-imposed over the attack and benign loss distributions of the mixed (test) data-set.

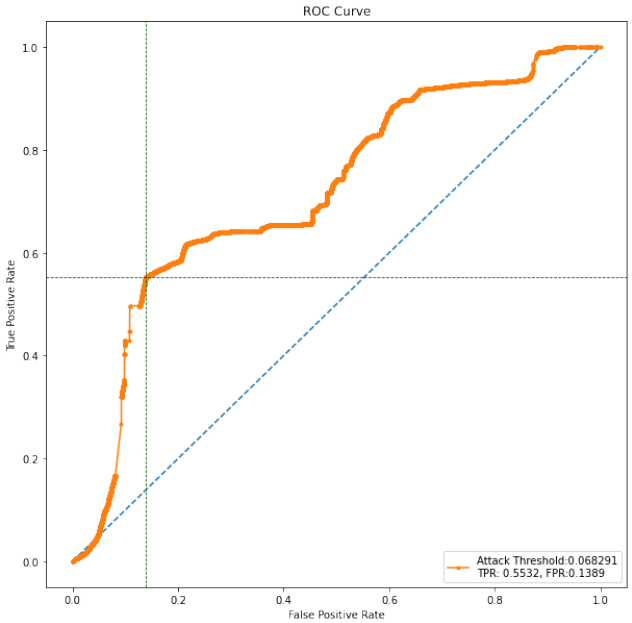

Figure 4: ROC Curve - Attack Threshold selection from Auto-encoder reconstruction loss on Tranalyzer flow data

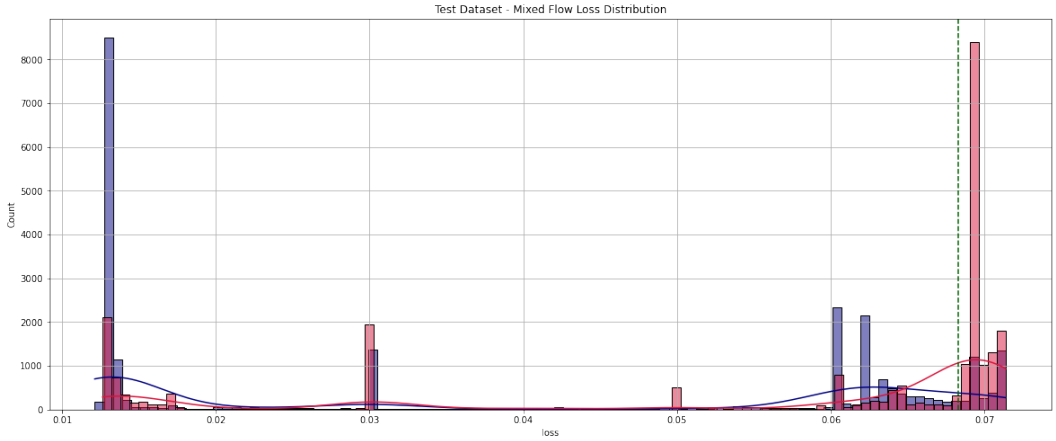

Figure 5: Auto-encoder reconstruction loss distribution of attack and benign Tranalyzer flows. Blue (benign) traffic is reconstructed with loss close to zero, while red (attack) traffic has a higher loss. Anything right of the green line (attack threshold) will be classified as an attack flow.

4.2 Result Analysis

Tables 1 and 2 show the results of the supervised and unsupervised approaches respectively. The decision tree performed similarly to the supervised baselines in other works [13], while SVM fell short on attack traffic prediction. The unsupervised PCA approach was also successful, yielding similar results but with a slightly lower recall for attack traffic. The deep auto-encoders did not work as well as our baselines, but still demonstrated that the technique was feasible. Interestingly, the Tranalyzer auto-encoder outperformed the Argus auto-encoder on overall detection. This could be because of better feature engineering, or Tranalyzer may just more informative features for our use-case.

From figure 5, you can see that the Tranalyzer auto-encoder was successful in learning a latent representation of benign traffic, resulting in a higher re-construction loss for anomalous attack traffic.

There are still groups of benign traffic with high loss and groups of attack traffic with low loss. The miss-classified flows could be a result of miss-labelling as described in section 3.2.2. It could also be a result of the auto-encoder learning a representation of TCP traffic that applies to both attack and benign flows.

Table 1: Attack Detection - Supervised Learning Baselines

Method	Data Type	Precision (%)	Recall (%)	F1 Score (%)
Decision Tree	Normal Traffic	99.0	99.0	99.0
Decision Tree	Attack Traffic	97.0	96.0	97.0
SVM	Normal Traffic	97.0	97.0	97.0
SVM	Attack Traffic	81.0	830.	82.0

Table 2: Attack Detection - Unsupervised Learning

Method	Data Type	Precision (%)	Recall (%)	F1 Score (%)
PCA	Normal Traffic	97.0	99.0	98.0
PCA	Attack Traffic	95.0	82.0	88.0
Auto-encoder (Tranalyzer)	Normal Traffic	71.0	87.0	78.0
Auto-encoder (Tranalyzer)	Attack Traffic	80.0	75.0	65.0
Auto-encoder (Argus)	Normal Traffic	80.0	58.0	67.0
Auto-encoder (Argus)	Attack Traffic	59.0	81.0	68.0

5 Conclusion and Future Work

The IoT traffic is prone to security threats and attacks. Machine learning is an effective technique that can learn from the normal traffic signatures and can differentiate between normal and attack traffic. From the results, the hypothesis can be proven. Although supervised learning techniques perform better than auto-encoder-based uncontrolled classifiers, when trained in benign traffic data, but that is mainly because it never sees new attacks (as we had limited dataset). Autoencoder is effective in modeling network behavior and detecting anomalies and attacks in IoT Networks and has potential to perform better for new attacks. Moreover, it can be noticed that Tranalyzer is a better tool for flow extraction as it provides better training results. We are trying to expand the scope of this initial study in the future. For that purpose, we can look into packet-based analysis for a better understanding of how that would be different from flow-based analysis. Also, different machine learning models, such as Variations Auto-encoder and One class SVM, can be utilized for comparison purposes. Another key direction is to use a more diverse data-set that includes newer types of attacks to see the effectiveness of unsupervised learning.

References

[1] M. Roesch Snort: Lightweight intrusion detection for networks Lisa. Vol. 99, 1999.

[2] P. Garcia-Teodoro et al. Anomaly-based network intrusion detection: Techniques, systems and challenges Computers and Security 28, 2009.

[3] R. Doshi, N. Apthorpe, N. Feamster Machine Learning DDoS Detection for Consumer Internet of Things Devices IEEE Deep Learning and Security Workshop 2018

[4] A. L. Buczak, E. Guven A survey of data mining and machine learning methods for cybersecurity intrusion detection IEEE Communications Surveys Tutorials 18.2 (2016)

[5] T. Zseby, et al. Nightlights: Entropy-based metrics for classifying darkspace traffic patterns Passive and Active Measurement, Cham:Springer International Publishing, pp. 275-277, 2014.

[6] P. Baldi Auto-encoders, Unsupervised Learning, and Deep Architectures Proceedings of ICML Workshop on unsupervised and transfer learning. 2012

[7] S. R. Gaddam, et al. K-Means+ ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and ID3 decision tree learning methods IEEE Transactions on Knowledge and Data Engineering, 2007

[8] R. Doshi, N. Apthorpe, N. Feamster Machine Learning DDoS Detection for Consumer Internet of Things Devices IEEE Deep Learning and Security Workshop 2018

[9] https://iotanalytics.unsw.edu.au/attack-data

[10] https://tranalyzer.com/

[11] https://openargus.org/

[12] https://github.com/MackPreston/iot-traffic-analysis

[13] Bhatia, Randeep, et al. "Unsupervised machine learning for network-centric anomaly detection in IoT." Proceedings of the 3rd acm conext workshop on big data, machine learning and artificial intelligence for data communication networks. 2019.

[14] Hamza, Ayyoob, et al. "Detecting Volumetric Attacks on IoT Devices via SDN-Based Monitoring of MUD Activity" SISR '19 April 3-4, 2019