Securing edge devices by monitoring their behavior

Bala Kesavan
Kyndryl
Chennai, India
balakrishna.kesavan@kyndryl.com

Ankit Bose
Kyndryl
Bengaluru, India
ankit.bose@kyndryl.com

***Abstract*—Edge devices provide critical services to enterprises and must be secured from costly attacks. We discuss monitoring edge device behavior and flagging anomalies using a multivariate, temporal, Convolutional Neural Network Auto-Encoder. It produces visual output that helps Subject Matter Experts focus attention and remedial actions on key events, thereby preventing further damage to the network.**

Keywords— anomaly detection, multivariate, T-CNN, Auto-Encoder

I. INTRODUCTION

Edge devices performing critical functions have often been attacked, leading to data theft, financial loss and reputational damage. The devices could even be hijacked for Distributed Denial of Service (DDoS) attacks or crypto currency mining.

DDoS attacks launched by edge devices have overwhelmed targets with over 1 Tbps of traffic [1], enough to bring down target websites. Edge devices are chosen for DDoS attacks because they have sufficient capacity to generate web traffic but are usually poorly secured. In Fig. 1. [1], a self-explanatory DDoS mechanism, based on the Mirai botnet is shown as an example.

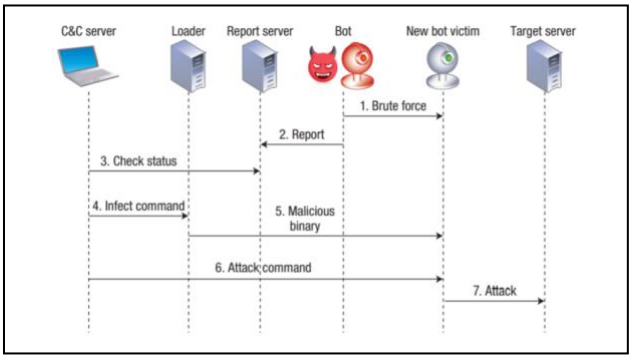

Fig. 1. DDoS generic mechanism.

There are several known approaches to securing edge devices, including secure firmware update mechanisms, encrypting all communication, and monitoring behavior of edge devices. Even during the process of being compromised and before participating in a DDoS attack, there is unusual traffic at the edge devices, like downloading binary code. When a pattern of abnormal behavior is known in advance, rules written into monitoring systems can alert administrators and Subject Matter Experts (SME). But bad actors innovate and try new tactics, techniques, and procedures (TTP) to attack edge devices, causing new forms of abnormal behavior.

Anomaly Detection (AD) can be used to flag such never-seen-before behavior, to prevent further damage to the network. A good AD solution is one that calls SME attention to nearly all truly problematic situations but does not throw up too many false alarms.

II. SOLUTIONS CONSIDERED

Monitoring data like Write_Response_Time, Read_Response_Time, Write_Data_Rate, Read_Data_Rate and Host_IO from the storage of edge devices were collected at hourly intervals for six months and used for AD model development. These features were treated as an interdependent and multivariate dataset. For example, a normally operating storage device is expected to show low Write_Response_Time when the Write_Data_Rate and Host_IO are set high. So, an unusually high Write_Response_Time reading, when the Write_Data_Rate and Host_IO are high could indicate anomalies.

Unsupervised AD techniques, that do not rely on previously known/ labelled examples for separating out anomalies were used. The development environment was Python 3 and used open-source libraries including scikit-learn, seaborn and TensorFlow. The key algorithms used are explained next.

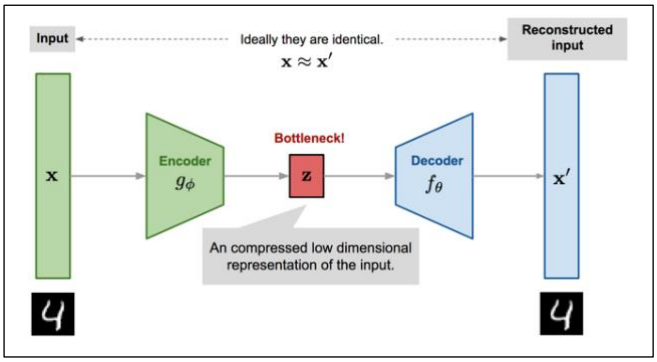

Fig. 2. Neural network Auto-Encoder.

The first option tried was the Auto-Encoder, which uses neural networks to learn to reconstruct the original input. Fig. 2. [5] shows a neural network Auto-Encoder trying to reconstruct the image of the handwritten digit, 4. The encoder half of the Auto-Encoder must compress the input to fit the bottleneck, which is a low dimensional representation of the input. Then, the decoder part of the Auto-Encoder must reconstruct the original input from this compressed representation. The difference between the input and the reconstruction is the reconstruction loss. The Auto-Encoder is trained by minimizing reconstruction loss.

The bottleneck layer is the key to this unsupervised learning approach. The Auto-Encoder must learn just enough so that it can reconstruct the input, without memorizing the specifics of the training data, since doing so would lead to

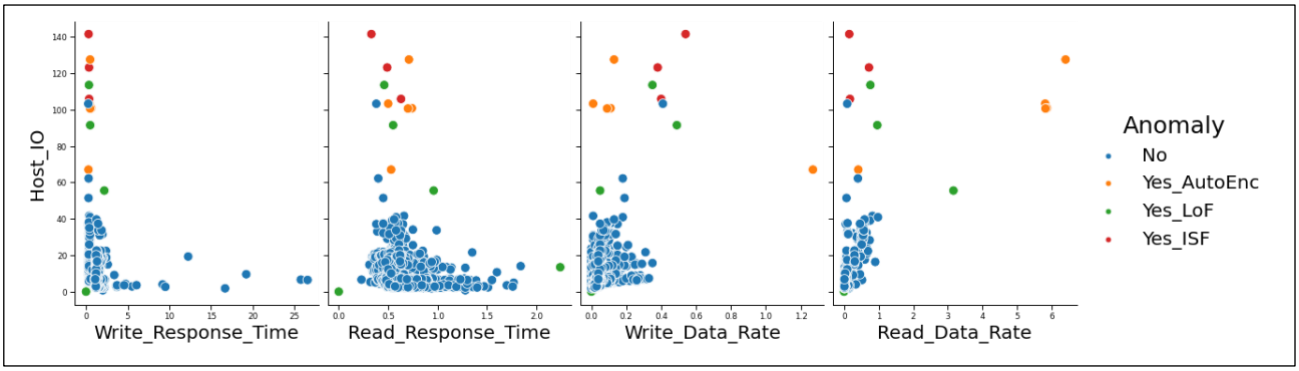

Fig. 3. Results from simple multivariate techique that treats each observation as independent of other observations.

overfitting and lack of generalization. The AD model must function as a generalized model that can handle similar but not identical inputs. One technique to help with generalization is to corrupt the input with Gaussian noise but still use the uncorrupted input as the target during training.

After the Auto-Encoder AD model is trained, predictions are generated (output) for new observations and reconstruction losses comparing the input and output are calculated. When a never or rarely seen input is presented, the Auto-Encoder is unable to reconstruct it, leading to large reconstruction loss which is flagged as an anomaly. Data is scaled for neural networks. Without it, the training process can be slow and unstable.

The second solution option tried, Isolation Forest (ISF) [2], is a technique in which the dataset is recursively partitioned by selecting a random feature and then splitting at a random value between the minimum and maximum value of the selected feature. Each such series of splits is thought of as an isolation tree and multiple trees, an Isolation Forest.

Anomalous observations require the least number of splits to be isolated, averaged across all trees. Hence, the number of splits is computed into an ISF score and the desired number of observations with the largest ISF scores are flagged as anomalies. This technique is insensitive to data scaling.

The third solution option, Local Outlier Factor (LOF) [3], is a technique in which each observations' average distance to its k-nearest neighbors is computed as its local density. Then, the local density of each observation is compared with that of its k-nearest neighbors to compute an LOF score. Then, the desired number observations with the largest LOF scores are flagged as anomalies. Since LOF uses distances to compute scores, this is sensitive to the range of values of each feature. To be reliable, the features have must be scaled.

The Auto-Encoder, ISF and LOF AD approaches explained above assume that each observation is independent of prior or later observations. Hence, these approaches focus only on unusual readings of individual features or unusual combinations of readings of all the features to identify

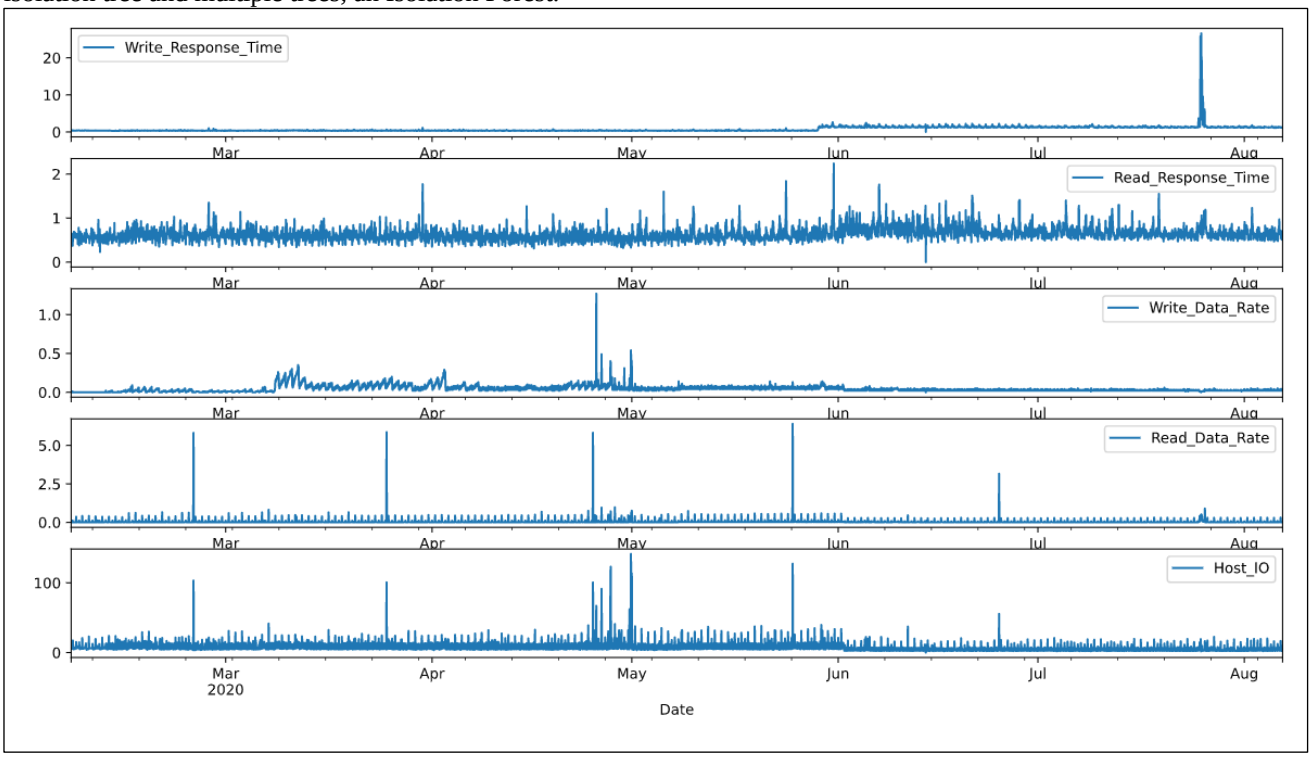

Fig. 4. Multivariate time series.

anomalies. Observations identified as normal or anomalous, by algorithm, are shown in Fig. 3.

But in several situations, including edge device storage, there is a strong relationship between successive readings. For example, considering the Write_Response_Time for one observation, it is reasonable to expect the next observation to also be in the same range. Where it is not, the change should either be explainable by a change in the Write_Data_Rate or Host_IO, or as a seasonal pattern. If not, this is an anomaly.

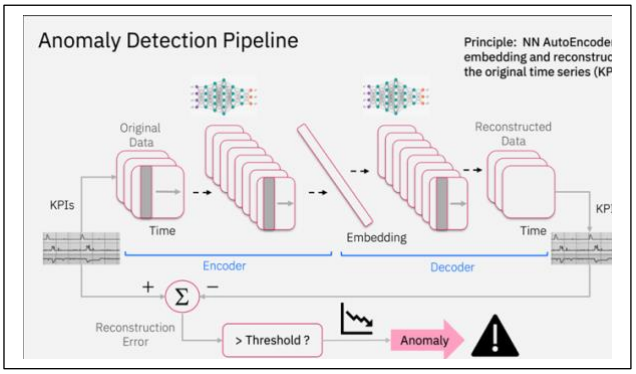

Fig. 5. Multivariate temporal convolutional neural network Auto-Encoder

Fig. 4. shows the same multivariate dataset presented as timeseries data. 'Read_Data_Rate' shows a spike towards the end of each month till June, but not in July or August. The sudden spikes till June are not anomalies, as they are part of the temporal pattern. In fact, the absence of a spike in August might be an anomaly.

Approaches like a simple Auto-Encoder, ISF or LOF would mistakenly identify such a "once a month" event, till June, as an anomaly. But a temporal approach would see this as a valid pattern. Hence, a multivariate temporal convolutional neural network (T-CNN) Auto-Encoder, shown in Fig 5. [4] was chosen. The need for a multivariate model that captures the interdependence of features was previously explained. The temporal requirement was added, to capture time dependent patterns.

III. Solution Architecture

The T-CNN model uses a sequence of past data to predict one step into the future. Hence, the data must be reshaped from single observations to sequences, as shown in Fig. 7. (Fig. 7. Shows 5 samples of six-hour sequences, just to illustrate the idea of sequences. The need for scaling was described earlier).

IV. Solution Architecture

Training data:

Date	Write_Response_Time	Read_Response_Time	Write_Data_Rate	Read_Data_Rate	Host_IO
2020-02-06 19:00:00	0.32	0.60	0.0	0.01	5.78
2020-02-06 20:00:00	0.34	0.55	0.0	0.01	5.30
2020-02-06 21:00:00	0.35	0.65	0.0	0.01	5.73
2020-02-06 22:00:00	0.36	0.53	0.0	0.01	5.65
2020-02-06 23:00:00	0.36	0.57	0.0	0.01	5.32

Scaled training data:

Date	Write_Response_Time	Read_Response_Time	Write_Data_Rate	Read_Data_Rate	Host_IO
2020-02-06 19:00:00	-0.56	-0.15	-1.0	-0.17	-0.32
2020-02-06 20:00:00	-0.51	-0.49	-1.0	-0.17	-0.38
2020-02-06 21:00:00	-0.48	0.18	-1.0	-0.17	-0.32
2020-02-06 22:00:00	-0.46	-0.62	-1.0	-0.17	-0.33
2020-02-06 23:00:00	-0.46	-0.35	-1.0	-0.17	-0.37

```
Sequenced data for a single feature, Write_Response_Time :
array([[-0.56, -0.51, -0.48, -0.46, -0.46, -0.53],
       [-0.51, -0.48, -0.46, -0.46, -0.53, -0.22],
       [-0.48, -0.46, -0.46, -0.53, -0.22, -0.56],
       [-0.46, -0.46, -0.53, -0.22, -0.56, -0.53],
```

Fig. 7. Pre-processing data for T-CNN Auto-Encoder

A relatively small neural network, shown in Fig. 8., with the following key features, was trained. The 1-dimensional convolution layers summarize data along the time dimension. In the top layer, the summarization is done using 64 filters that move with stride length of 2 over a 48 hour window of data. The transpose layers expand the data back into the input format.

Layer (type)	Output Shape	Param #
conv1d_9 (Conv1D)	(None, 24, 64)	27712
dropout_12 (Dropout)	(None, 24, 64)	0
conv1d_10 (Conv1D)	(None, 12, 32)	12320
dropout_13 (Dropout)	(None, 12, 32)	0
conv1d_11 (Conv1D)	(None, 6, 16)	3088
conv1d_transpose_12 (Conv1DT	(None, 12, 16)	1552
dropout_14 (Dropout)	(None, 12, 16)	0
conv1d_transpose_13 (Conv1DT	(None, 24, 32)	3104
dropout_15 (Dropout)	(None, 24, 32)	0
conv1d_transpose_14 (Conv1DT	(None, 48, 64)	12352
conv1d_transpose_15 (Conv1DT	(None, 48, 72)	27720

```
Total params: 87,848
Trainable params: 87,848
Non-trainable params: 0
```

Fig. 8. Auto-Encoder architecture

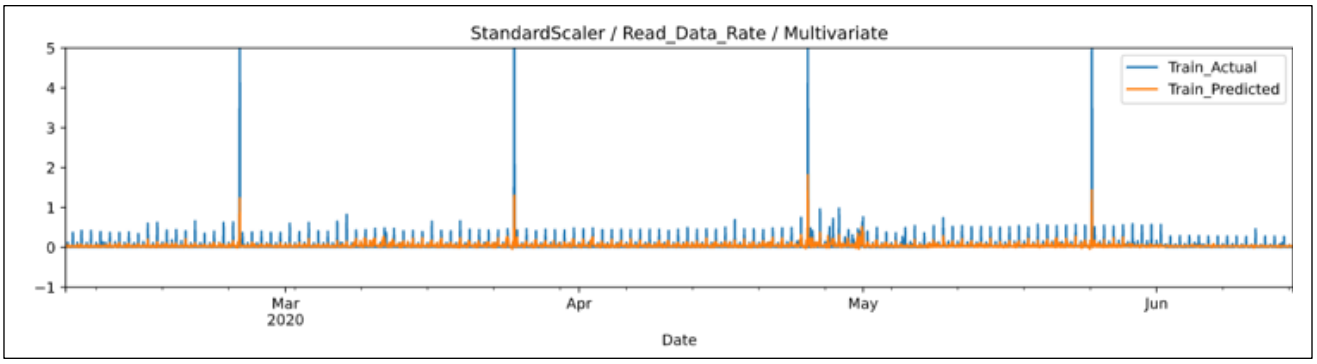

Fig. 6. Auto-Encoder correctly picking up the month end spikes.

The dropout layers deactivate different nodes, at random, during training, effectively presenting different network architectures at each 'epoch' or one full cycle of training through all the training data. First, at each epoch, the network becomes sparse, forcing it extract features better and not memorize specifics. Second, across all the training epochs, the different network architectures behave like an ensemble, further boosting generalization. Early stopping criteria terminated model training after 22 epochs, when the validation loss plateaued. To improve generalization, Gaussian noise was used in the previous Auto-Encoder architecture and has been replaced by dropout here.

Fig.6. shows the train and predicted data for the 'Read_Data_Rate' feature. Despite the training data having just four instances of the month end spikes, the time-series Auto-Encoder is able to learn the pattern. This temporal pattern detection meets the key reason for choosing a T-CNN Auto-Encoder.

The top half of Fig.9. shows the reconstruction error for the "Write Response Time" feature, with a spike just before July 27th. The reconstruction error being the difference between the actual and predicted values, the next question that arises is what amplitude and duration of error constitutes an anomaly. Too many false positives lead to wasted effort in investigation. Missing relevant anomalies leads to avoidable failures. To balance between the two, close collaboration with the administrators is required to set the right amplitude and duration for error thresholds. Users may choose to ignore one-time or short-lived spikes in reconstruction errors. But a sustained reconstruction error may be of interest. In this project, only those anomalies that sustained for at least one full day were flagged.

```
kdd cup 1999 shape:  (494021, 42)
kdd cup 1999 outcome column value counts:
smurf.              280790
neptune.            107201
normal.              97278
back.                 2203
satan.                1589
ipsweep.              1247
portsweep.            1040
warezclient.          1020
teardrop.              979
pod.                   264
nmap.                  231
guess_passwd.           53
buffer_overflow.        30
land.                   21
warezmaster.            20
imap.                   12
rootkit.                10
loadmodule.              9
ftp_write.               8
multihop.                7
phf.                     4
perl.                    3
spy.                     2
Name: outcome, dtype: int64

shape of train data with only 'normal' records
(92414, 80)
counts and shape of test data with mixed anomaly yes and no records
no     4864
yes    1653
Name: anomaly, dtype: int64
(6517, 80)
```

Fig. 10. KDD Cup 1999 data.

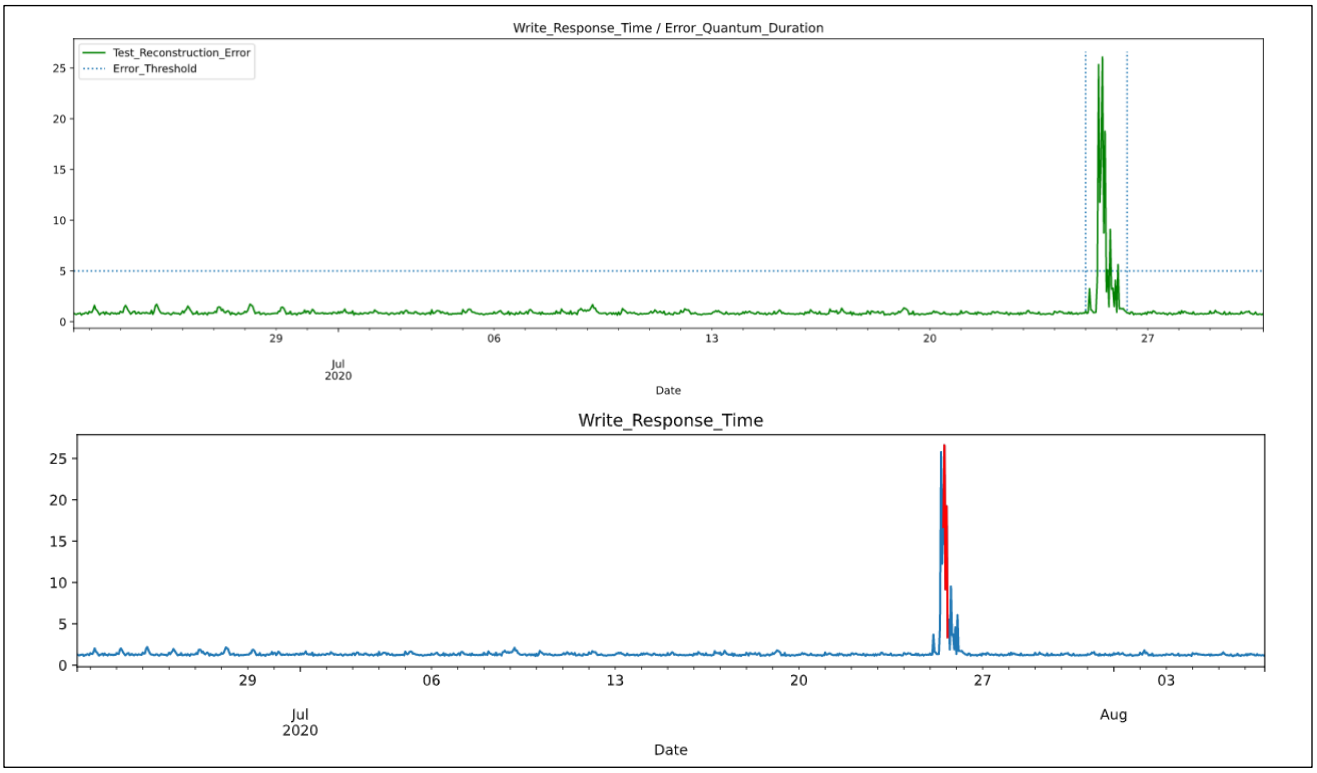

Fig. 9. Reconstruction error threshold tuning.

The bottom half of Fig.9. shows the anomaly being overlayed on the "Write Response Time" plot. Such highlighting can help the administrators to focus their attention on important issues.

V. Results

The effectiveness of this AD solution can only be assessed from user feedback and no objective measures are available since there is no labelled data. For objective measures, like precision, recall and f1score, the same Auto-Encoder is repurposed as a classification tool trained on labeled data. The 'KDD Cup 1999' data [6], shown in Fig.10., about simulated intrusions into a military network is used, with modifications.

The 'outcome' column of the data are the labels. Here, 'normal' records as set as 'anomaly = no' and rare labels, each of which occur less one thousand times, as 'anomaly = yes'. From this, only 'anomaly = no' records are used for training and mix of 'anomaly = no' and 'anomaly = yes' records for testing.

Precision is a measure of how many observations that are flagged as anomalous are actually anomalous. This is an important metric used to prevent users from being overwhelmed by too many alerts. Recall measures how many of the anomalies the AD model was able to detect. The f1 score is the harmonic mean of Precision and Recall and thus captures the information in both.

This approach is akin to novelty detection where the reconstruction error spikes when 'never-seen-before' data is injected. The Auto-Encoder architecture described earlier is reused with modifications made only to fit the shape and data types in the KDD data, as shown in Fig.11.

Layer (type)	Output Shape	Param #
conv1d_3 (Conv1D)	(None, 24, 64)	30784
dropout_4 (Dropout)	(None, 24, 64)	0
conv1d_4 (Conv1D)	(None, 12, 32)	12320
dropout_5 (Dropout)	(None, 12, 32)	0
conv1d_5 (Conv1D)	(None, 6, 16)	3088
conv1d_transpose_4 (Conv1DTr	(None, 12, 16)	1552
dropout_6 (Dropout)	(None, 12, 16)	0
conv1d_transpose_5 (Conv1DTr	(None, 24, 32)	3104
dropout_7 (Dropout)	(None, 24, 32)	0
conv1d_transpose_6 (Conv1DTr	(None, 48, 64)	12352
conv1d_transpose_7 (Conv1DTr	(None, 48, 80)	30800

Total params: 94,000
Trainable params: 94,000
Non-trainable params: 0

Fig. 11. Auto-Encoder architecture, modified for KDD Cup 1999 data.

This model is trained for 50 epochs and it is noticed that the validation loss continues to drop. There may be value in further training or other modifications, like adding more filters per layer. Even now, the trained model seemed to perform quite well, as seen in the charts in Fig.12. The threshold being tuned is for the reconstruction loss. Values exceeding threshold are marked as anomalies. Setting the threshold at about 0.75 might be a good balance between the cost of investigating too many false positives and missing important alerts. Apart from objective feedback, users' subjective feedback is also important. The SME that reviewed the AD model concurred that the model helped focus attention on the events that mattered and further investigation of these events was appropriate.

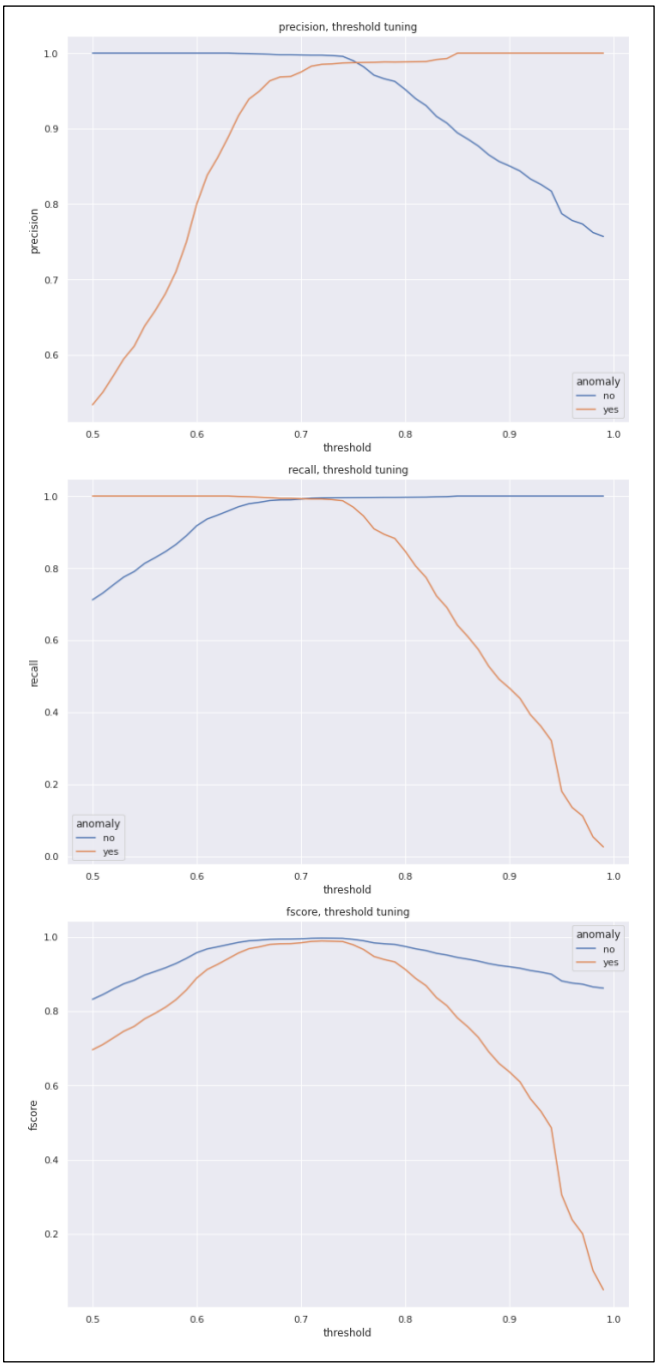

Fig. 12. Auto-Encoder model performance on KDD Cup 1999 data, with reconstruction loss threshold tunning.

VI. References

[1] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, Jefrey Voas, "DDoS in the IoT: Mirai and Other Botnets"

[2] Fei Tony Liu Kai Ming Ting and Zhi-Hua Zhou, "Isolation-based Anomaly Detection"

[3] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander , "LOF: Identifying Density-Based Local Outliers"

[4] Roy Assaf , Ioana Giurgiu , Jonas Pfefferle, Serge Monney, Haris Pozidis and Anika Schumann, "An Anomaly Detection and

Explainability Framework using Convolutional Autoencoders for Data Storage Systems”

[5] Fig 2. Source: https://www.kaggle.com/robinteuwens/anomaly-detection-with-auto-encoders

[6] KDD Cup 1999 Data: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html