Detection of IoT Botnet Attacks

Bhagyashri Bhamare
Dept. of Information Technology
NITK Surathkal
Mangalore, India
bhamare.bhagyashri1999@gmail.com

Chinmayi C. R.
Dept. of Information Technology
NITK Surathkal
Mangalore, India
chinmayicr27@gmail.com

Utkarsh Meshram
Dept. of Information Technology
NITK Surathkal
Mangalore, India
utkarshsudhir.181it250@nitk.edu.in

***Abstract*—The growing adoption of Internet-of-Things devices brings with it the increased participation of said devices in botnet attacks, and as such novel methods for IoT botnet attack detection are needed. The paper proposes a network-based anomaly detection method for the IoT called N-BaIoT. This extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic from compromised IoT devices. The autoencoders have more than one hidden layer. To evaluate the method, the paper has taken dataset of nine infected commercial IoT devices with two widely known IoT-based botnets, Mirai and BASHLITE. The results proved the accuracy of the model. An anomaly detection and attack classification pipeline has been implemented. LIME has been used to explain the features that contribute to the prediction.**

***Index Terms*—botnet, IoT, autoencoders, LIME**

I. Introduction

A ‘bot’ is a computer program which enables the operator to remotely control the infected system where it is installed. A network that is compromised with the attack by such bots is called a botnet. It is essential to detect such bots in the network to ensure safety of a system. The proliferation of IoT devices which can be more easily compromised than desktop computers has led to an increase in the occurrence of IoT based botnet attacks. There is a need to differentiate between hour and millisecond long IoT based attacks. For detecting attacks launched from IoT bots we propose N-BaIoT, a network-based approach for the IoT that uses deep learning techniques to perform anomaly detection. Specifically, we extract statistical features that capture behavioral snapshots of benign IoT traffic, and train a deep autoencoder (one for each device) to learn the IoT device’s normal behaviors. The auto encoders attempt to compress snapshots. When an auto encoder fails to reconstruct a snapshot, it is a strong indication that the observed behavior is anomalous (the IoT device has been compromised and is exhibiting an unknown behavior). An advantage of using deep autoencoders is their ability to learn complex patterns—for example, of various device functionalities. This results in an anomaly detector with hardly any false alarms. An attack classification is done with the help of deep neural network.

II. Literature Survey

Previous IoT-related botnet detection studies focused mainly on the early steps of propagation and communication with the C&C server [1] [3]. Botnet detection approaches are either host-based [3] or network-based [1] [2]. We consider host-based techniques less realistic because not all IoT manufacturers can be relied on to install designated host-based anomaly detectors on their products. A hierarchical taxonomy of network-based botnet detection approaches, not limited to the IoT domain, was proposed by Sebastián García, Alejandro Zunino, and Marcelo Campo [4]. One of the detection sources they surveyed was honeypots, which have commonly been used for collecting, understanding, characterizing, and tracking botnets14 but are not necessarily useful for detecting compromised endpoints or the attacks emanating from them. Moreover, honeypots normally require a substantial investment in procurement or emulation of real devices, data inspection, signature extraction, and keeping up with mutations. Kalyan Veeramachaneni and his colleagues [5] extended autoencoders for outlier detection but they still required security analysts to actively label data for subsequent supervised learning. Closer to our approach, Aaron Tuor and his co authors [6] apply deep learning to system logs for detecting insider threats. Differently from us, they use DNNs and recurrent neural networks (RNNs), and depend on further manual inspection.

III. Methodology

A. Feature Extraction

The data [12] is obtained by extracting a total of 115 traffic statistics. Data set is split into train and test subsets as 80:20. From the data, 23 features are taken over 5 temporal windows: the most recent 100 ms, 500 ms, 1.5 sec, 10 sec, and 1min.

B. Training an Anomaly Detector

For training and optimization, we use two separate data sets that only contain benign data.
The first data set is the training set (DS_{trn}):

- Used for training the autoencoder
- Given input parameters such as the learning rate (η , the size of the gradient descent step) and the number of epochs (complete passes through the entire DS_{trn}).

The second dataset is the optimization set (DS_{opt}):

- Used to optimize these two hyper parameters (η and epochs) iteratively until the mean square error (MSE) between a model’s input (original feature vector) and output (reconstructed feature vector) stops decreasing.
- Stopping at this point prevents overfitting DS_{trn}, thus promoting better detection results with future data.

DS_{opt} is later used to optimize a threshold (tr) that discriminates between benign and malicious observations and, finally, the window size (ws), by which the FPR is minimized.

$$tr* = \overline{MSE}_{DS_{opt}} + s(MSE_{DS_{opt}})$$

Fig. 1. Threshold value

The abnormality decision is based on a sequence of instances by implementing a majority vote on a moving window.

We determine the minimal window size ws* as the shortest sequence of instances, a majority vote that produces 0 percent FPR on DS_{opt}:

$$ws^* = \arg\min_{|ws|}(|\{packet \in ws \mid MSE(packet) > tr^*\}| > \frac{ws}{2})$$

Fig. 2. Minimal Time Window size

Autoencoder [9] uses 5 hidden layers of sizes 0.75, 0.5, 0.25, 0.5, 0.75 of the input feature vector size.
Hyperbolic tangent is used as an activation function [9] for our hidden unit neuron. A nonlinear function is used to retain the power of nonlinear models.

C. Attack Classification

The problem of classification is solved with a deep neural network with two hidden layers each with 8 neurons. The proposed neural network consists of:

- Hyperbolic tangent activation function for hidden neurons

$$g(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Fig. 3. Hyperbolic Tangent Activation Function

- Softmax function applied to the last layer

$$softmax(x_j) = \frac{e^{x_j}}{\sum_{i=1}^{N} e^{x_i}}$$

Fig. 4. Softmax Function

- Categorical cross-entropy as a loss function

$$P(y = 1 \mid \boldsymbol{x}) = \max\left\{0, \min\left\{1, \boldsymbol{w}^\top \boldsymbol{h} + b\right\}\right\}$$

Fig. 5. Cross Entropy as Loss Function

D. Evaluation metrics

A two-class confusion matrix is used to evaluate the results as in Table 1.

	Predicted normal	**Predicted attack**
Actual normal	True Negative (TN)	False Positive (FP)
Actual attack	False Negative (FN)	True Positive (TP)

TABLE I
CONFUSION MATRIX

E. Local Interpretable Model-Agnostic Explanations

LIME is used to explain the predictions of complex model in a human interpretable form. This technique works by perturbing values of a particular instance that we want to explain and learning a local linear approximation for classification.This method provides features that resulted in the prediction by the model. Feature weights from a simple model make explanations for the complex model's local behaviour.

IV. Tools and technologies

The language of choice is Python2 3.6. Recently Python has been extremely popular with machine learning community thanks to many factors. The wealth of libraries and frameworks for data processing and data analysis. Most of the experiments for this thesis were done in a form of a Jupyter3 notebook which allows for better visualization and rerunning of different parts of an experiment. Pandas4 library is used to handle the csv data for this work. Pandas data frames, backed by Numpy5 This work also relies on scikit-learn6 library for the scaling and evaluation metrics functions. Neural network models are composed using Keras7 2.2 library, which in turn relies on Tensorflow8 framework for all the computations.

V. Results

A. Feature Scoring

Fisher's score was calculated on training sets for each data subset used respectively for attack detection, botnet type classification and Mirai attack type classification.

2 class (normal, attack)		**3 class (2 botnets + 1 normal)**		**5 classes of Mirai attacks**	
Feature	**F Sc**	**Feature**	**F Sc**	**Feature**	**F Sc**
MI_dir_L0.1_weight	3.34	MI_dir_L3_weight	1.96	MI_dir_L0.01_var	43.75
H_L0.1_weight	3.34	H_L3_weight	1.96	H_L0.01_var	43.75
MI_dir_L1_weight	3.18	MI_dir_L5_weight	1.93	MI_dir_L0.1_var	41.43
H_L1_weight	3.18	H_L5_weight	1.93	H_L0.1_var	41.43
MI_dir_L3_weight	3.01	MI_dir_L1_weight	1.87	MI_dir_L0.01_mean	30.05
H_L3_weight	3.01	H_L1_weight	1.87	H_L0.01_mean	30.05
MI_dir_L5_weight	2.86	MI_dir_L0.1_weight	1.70	MI_dir_L0.1_mean	27.03
H_L5_weight	2.86	H_L0.1_weight	1.70	H_L0.1_mean	27.03
MI_dir_L0.01_weight	1.65	MI_dir_L0.01_weight	1.43	MI_dir_L1_var	19.62
H_L0.01_weight	1.65	H_L0.01_weight	1.43	H_L1_variance	19.62

TABLE II
FISHER'S SCORES

B. Attack detection

The auto encoder was created and trained on all 115 features. Default Keras optimization hyper parameters were used. Training was limited to 100 epochs with additional condition of early stopping (Figure 6) using functionality provided by Keras. The model trained in total for 23 epochs, taking about 25 seconds for each epoch, totaling in 584 seconds of training time. Loss is defined as MSE between original and predicted values. Training set had 185310 samples and optimization (validation) set 185311 samples.

```
es = EarlyStopping(monitor='val_loss', patience=5)
```

Fig. 6. Keras early stopping

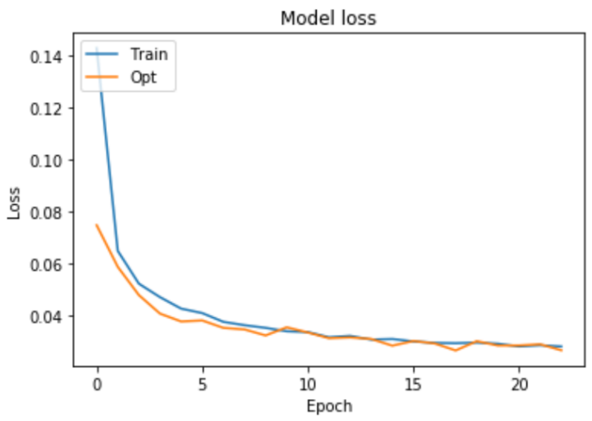


Fig. 7. Autoencoder Training Curve

Values of integers from 1 to 10 were tried for the threshold equation (Figure 1) parameter N on the optimization dataset.

N	Accuracy	False positives	False negatives
1	0.9894	3911	23
2	0.9934	2203	25
3	0.9961	1396	28
4	0.9973	978	35
5	0.9979	744	38
6	0.9983	587	41
7	0.9986	467	43
8	0.9988	387	45
9	0.9990	292	51
10	0.9992	245	52

TABLE III
RESULTS WITH DIFFERENT N VALUES FOR THRESHOLD

With model trained and threshold selected it is now possible to do an evaluation on the test set. Achieved accuracy is 0.9991 and precision is 0.9985. A more detailed breakdown is given by a confusion matrix Figure 8. False positive rate is thus 0.0015.

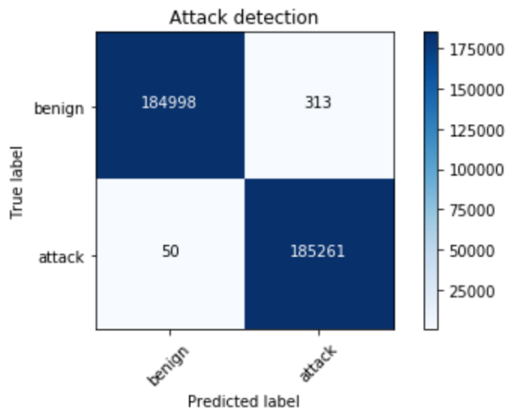


Fig. 8. Attack detection on test data set

C. Attack Classification

Two types of deep learning classifier models were trained to differentiate between different botnet types (Mirai, BASHLITE and also non-botnet) and between different Mirai botnet attack types (SCAN, ACK, SYN, UDP, UDPPlain).

1) Botnet type classification: -

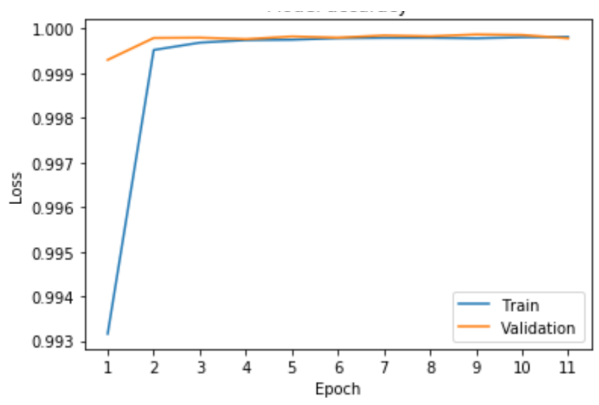


Fig. 9. Botnet classification learning curve

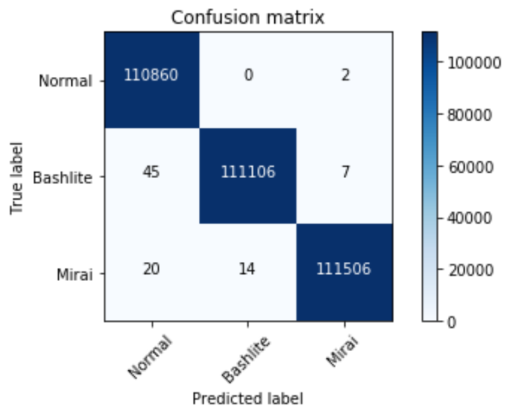


Fig. 10. Botnet classification confusion matrix

2) Mirai attack type classification:

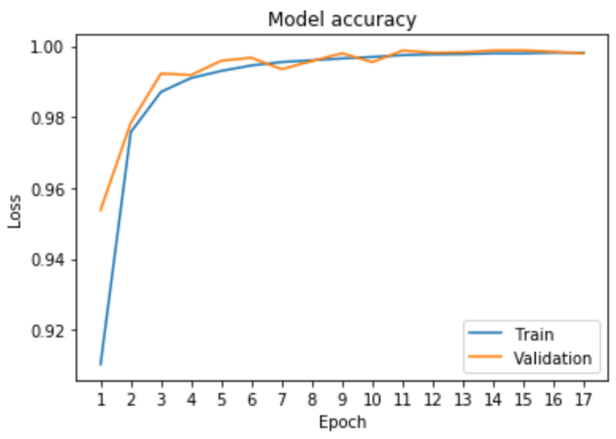
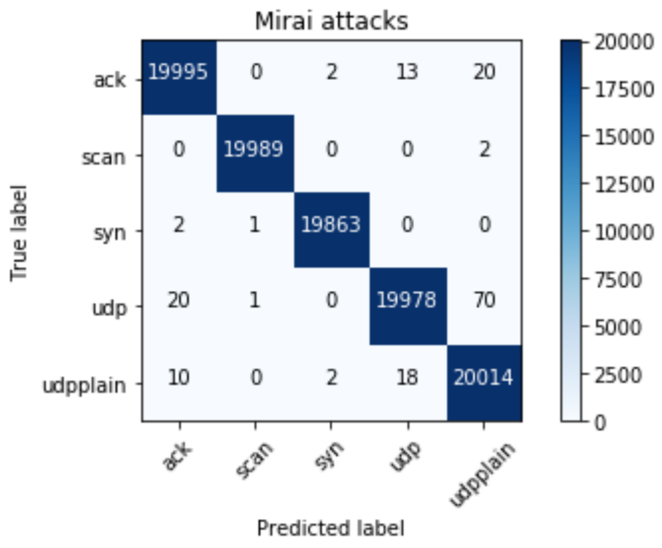


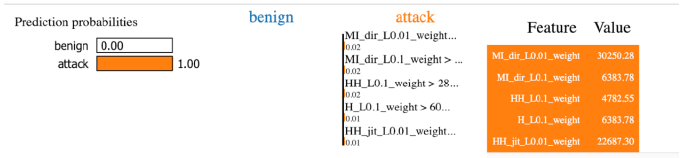
Fig. 11. Mirai attack classification learning curve



	ack	scan	syn	udp	udpplain
ack	19995	0	2	13	20
scan	0	19989	0	0	2
syn	2	1	19863	0	0
udp	20	1	0	19978	70
udpplain	10	0	2	18	20014

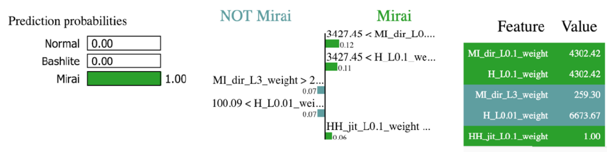
Fig. 12. Mirai attack classification confusion matrix

D. Local Interpretable Model-Agnostic Explanations



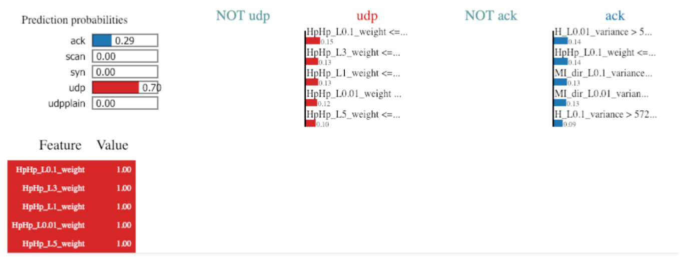
Feature	Value
MI_dir_L0.01_weight	30250.28
MI_dir_L0.1_weight	6383.78
HH_L0.1_weight	4782.55
H_L0.1_weight	6383.78
HH_jit_L0.01_weight	22687.30

Fig. 13. LIME explanation for attack detection



Feature	Value
MI_dir_L0.1_weight	4302.42
H_L0.1_weight	4302.42
MI_dir_L3_weight	259.30
H_L0.01_weight	6673.67
HH_jit_L0.1_weight	1.00

Fig. 14. LIME explanation for botnet classification



Feature	Value
HpHp_L0.1_weight	1.00
HpHp_L3_weight	1.00
HpHp_L1_weight	1.00
HpHp_L0.01_weight	1.00
HpHp_L5_weight	1.00

Fig. 15. LIME explanation for Mirai attack type UDP

VI. Implementation in Real Environment

There would be however, many practical considerations. The anomaly detection part of the pipeline should be easier to implement in practice, as there would be no need to obtain and maintain a data set for different botnets and their attacks. It would still require sampling traffic from an IoT network that is not yet infected, 40 of course, which could be a problem in and on itself. Further, it is not clear, how the method performance would change if the network itself is changed, for example: new devices are added. If it would be a device of the same type as the one we already have in the network, e.g. one more security camera, but from another manufacturer, perhaps the data distribution will not change much, but this is just speculation. As data set actually contains multiple devices from similar categories, The theoretically good false positive rate of 0.0015 in practice would still mean that more than 1 in 1000 normal data points would be falsely labeled as an attack. If a data point is generated every 100 milliseconds, that would give us a false alarm every 100 seconds, and this is not practical. One data point by itself, however, could hardly be considered an attack, where a continuous stream of packets is usually sent. To that regard, an approach suggested by [2]could be implemented, where an alarm only goes off if there is a majority of attack data points within a window of some specified size. While the attack detection mechanism based on autoencoder could be developed locally, maybe by an engineer maintaining the IoT network, the botnet classifier presents a challenge in the form of data set requirements. Ideally, this is a task for a cyber security firm or a research university team, that specializes in IoT botnets and collects and maintains data sets needed to train an effective classifier that would recognize a wide range of different botnets and attack types

VII. Conclusion

With the rapid development the IoT technologies, the cyber-attacks are mostly targeting these devices. Especially, the botnets attacks are vastly challenging in these environments. After infecting the botnets, the attackers can control these devices via the C & C server and launch attacks to the victim hosts. We hypothesize that the difficulty in capturing the normal traffic behavior varies among IoT devices, and that this difficulty may be correlated with [1] the device's capabilities, and [2] the network communications it normally produces. A similar notion was raised by [2], who argues that the specialized functionality of today's IoT devices leads to predictable behaviors. In turn, the ease of establishing baseline behaviors for IoT devices facilitates anomaly detection as a means of detecting attacks.

Deep learning methods demonstrated good IoT botnet attack detection and classification accuracy. Moreover, these methods can work well with varying number of features and generally their performance does not suffer from extra features, meaning that in real world environment they offer the possibility of using all existing data features.

The attack detection with deep autoencoder showed outstanding results with accuracy of 0.9991 and false positive rate of 0.0015. The anomaly detection with autoencoders can be applied to the data coming from multiple different IoT devices. The LIME technique was applied to an attack data point that gave an intuitive interpretation of the features and predicted output. Both botnet type classification and Mirai attack type classification showed very good results on the whole 115 feature set with accuracies over 0.99.

We presume that the predictability of traffic behavior can be directly translated into performance measures of anomaly detection. For example, an IoT device with a high level of traffic predictability would make any anomalous action stand out, and thus the TPR should increase and detection times should decrease in this case. For empirical validation we extracted static and dynamic features from the (benign) training set. Then we trained regression models to study these features' effect on the average FPR and detection times, obtained on the test set by the four detection methods we evaluated.

Ultimately, a solid predictability score can be leveraged by large organizations in order to ensure network functionality and limit the impact that compromised devices might have on the network. That is, security policies may not allow the connection of IoT devices with low predictability scores to their networks, since they pose difficulties in attack detection. In our future work we plan to further define and investigate the subject of traffic predictability, both theoretically and empirically.

References

[1] M. Özçelik, N. Chalabianloo, and G. Gür, "Software-Defined Edge Defense against IoTBased DDoS," Proc. 2017 IEEE Int'Conf. Computer and Information Technology (CIT17), 2017; doi.org/10.1109/CIT.2017.61.

[2] H. Bostani and M. Sheikhan, "Hybrid of Anomaly-Based and Specification-Based IDS for Internet of Things Using Unsupervised OPF Based on MapReduce Approach," Computer Comm., vol. 98, 2017, pp. 52–71.

[3] D.H. Summerville, K.M. Zach, and Y. Chen, "Ultra-Lightweight Deep Packet Anomaly Detection for Internet of Things Devices," Proc. 2015 IEEE 34th Int'l Performance Computing and Comm. Conf. (IPCCC 15), 2015; doi.org/10.1109/PCCC.2015.7410342.

[4] S. García, A. Zunino, and M. Campo, "Survey on Network-Based Botnet Detection Methods," Security and Communication Networks, vol. 7, no. 5, 2014, pp. 878–903.

[5] K. Veeramachaneni et al., "AI$\hat{2}$: Training a Big Data Machine to Defend," Proc. IEEE 2nd Int'l Conf. Big Data Security on Cloud, IEEE Int'l Conf. High Performance and Smart Computing, and IEEE Int'l Conf. Intelligent Data and Security (BigDataSecurity-HPSC-IDS 16), 2016; doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.79.

[6] A. Tuor et al., "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams," Proc. AAAI 2017 Workshop on Artificial Intelligence for Cybersecurity, 2017; https://arxiv.org/pdf/1710.00811.pdf.

[7] M. Stevanovic and J. M. Pedersen, "On the Use of Machine Learning for Identifying Botnet Network Traffic," Journal of Cyber Security, vol. 4, pp. 1-32, 2016.

[8] A. A. Pol, "Anomaly detection using Deep Autoencoders for the assessment of the quality of the data acquired by the CMS experiment," in 23rd International Conference on Computing in High Energy and Nuclear Physics, Sofia, 2018

[9] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[10] C. Seaman, "Threat Advisory: Mirai Botnet," [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/state-of-theinternet/ akamai-mirai-botnet-threat-advisory.pdf. [Accessed 6 January 2019].

[11] Y. Meidan, et al, "N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders," IEEE PERVASIVE COMPUTING, vol. 18, no. 9, 2018.

[12] Y. Meidan, et al , "iot botnet dataset," [Online]. Available: https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT. [Accessed 27 December 2018].