MIRAI Botnet Attack Detection with Auto-Associative Dense Random Neural Network

Mert Nakip and Erol Gelenbe†, *Fellow, IEEE*
Institute of Theoretical and Applied Informatics
Polish Academy of Sciences, IITIS-PAN
44-100 Gliwice, Poland
& Yaşar University, Bornova, Izmir, Turkey
†Lab. I3S, Université Côte d'Azur,
Grand Château, 06103 Nice Cecex 2, France

***Abstract*—Internet connected IoT devices have often been particularly vulnerable to Botnet attacks of the Mirai family in recent years. Thus we develop an attack detection scheme for Mirai Botnets, using the Auto-Associative Dense Random Neural Network that has recently been successful for other attacks such as the SYN attack. The resulting method is trained with normal traffic and tested with attack traffic, and shown to result in high accuracy detection of attacks with low false alarms. The approach is compared on the same data set with two other common Machine learning methods (Lasso and KNN) and shown to have higher accuracy, and much lower computation times than KNN and slightly higher (but comparable) computation times with respect to Lasso.**

***Index Terms*—Mirai Botnet Attacks, Attack Detection, Auto-Associative Dense Random Neural Networks, Machine Learning**

I. Introduction

The need to use large numbers of low cost and low maintenance devices for the IoT created vulnerabilities which dramatically manifested themselves in 2016, when a massive distributed denial of service (DDoS) attack took down large numbers of web sites including Spotify, Twitter, Reddit, Netflix, through the DNS service for domain name management [1], [2]. It also created malicious accesses from IP addresses numbering tens of millions, towards servers of some leading cyber-security companies [3].

Known as the Mirai ("future" in Japanese) Botnet, this form of DDoS attack sends TCP SYN requests to a large number of IP addresses, and if the victim responds it then uses the weak login credentials of many IoT devices based on default usernames and passwords initially set in the factories that produce the IoT devices, when these credentials are not changed after installation and connection to the Internet. If the attacker is successful, it installs malware at its victims; it blocks the victim's ports that used for updates and generates traffic to overwhelm other servers and devices with nonsense requests, also leading to threats and protection rackets [4], [5]. However Botnets can also target other critival infrastructures including smart vehicles [6] as well as the core Internet itself and its data collection capabilities which are critical for managing the Internet in real-time [7].

If the attack is detected at a device,a rapid change of password and reboot would be needed, but the reboot itself can be hampered if the malware has blocked the victims ports that are used for maintenance and updates.

Different forms of Mirai such as "Satori" have been known to infect mainstream network routers, while the "Okiru" version has aimed at infecting popular processors for embedded systems such as PowerPC, MIPS, ARM, x86, PowerPC, and Linux devices such as the popular Argonaut RISC Core processor (ARC), all of which have literally been shipped at many billions of devices per year. Other variants such as "Masuda" and "Wicked" also have targeted routers, while many versions of Mirai have been observed to attack IoT devices, and machines equipped with the Linux operating system. and Android based mobile phones [8].

Thus detailed studies to understand the characteristics have been conducted on these attacks [9], [10], recent work has studied the characteristics of their attack traffic [11], [12] and blockchain has been suggested [13] to protect IoT devices against Botnets.

In this paper we develop a Mirai Botnet attack detection technique based on machine learning (ML) with a sspecific the Random Neural Network (RNN) architecture [14]–[16], called a Dense RNN [17] which uses tight clusters of spiking neuronal cells for deep learning. Such techniques have been previously used with success to detect SYN attacks [18]. Earlier work for video quality evaluation [19] and network design [20] have shown the effectiveness of the conventional RNN model [21] to address problems in communication systems. There has also been other work [22] where the RNN has been used to optimize IoT systems for home climate control. An extensions of the RNN has also successfully been used for modeling adaptation in Gene Regulatory Networks [23].

In [24] it was shown that the Dense RNN with an auto-associative learning algorithm provides more accurate SYN attack detection than some other lML models. Thus in this paper we also use the Dense RNNs for deep learning in auto-associative mode, and compare the outcome to several other ML techniques both for detection accuracy and speed –

This research has been supported by the European Commission H2020 Program under the IoTAC Research and Innovation Action, under Grant Agreement No. 952684.

including the learning and detection computation times.

II. Auto-Associative Dense Random Neural Network for Attack Detection

In this section we describe the attack detector which is based on the Auto-Associative Dense Random Neural Network (AA-Dense RNN-AD), whose architecture is shown in Figure 1. It is composed of three modules, namely Metric Extraction, Auto-Associative Dense RNN (AA-Dense RNN), and the Attack Decision Maker.

First, via a Metric Extraction Module we extract the relevqnt metrics of the attack packets in order to capture the footprints of Botnet attacks. Figure 1, shows the metrics x_k^i for the packet k. These metrics are determined in-advance, and the Metric Extraction Module only extracts them from the data packets of IoT traffic; i.e., this module does not select the metrics.

1) Selecting the Metrics: Recall that Mirai is a type of attack that spreads to IoT devices over the network. In addition, every device infected by Mirai generates additional traffic to cause a massive DDoS in the network and infect even more nodes, so that the resulting traffic pattern of infected devices, including the inter-transmission time characteristics. Thus, we intuitively know that, when a device is affected by the Mirai attack, it will increase the total size of the traffic to overload the network by generating more packets. Accordingly we select the candidates for the important metrics of MIRAI Botnet traffic as follows:

- **Metric 1**: The total size of the last K transmitted packets,
- **Metric 2**: The average inter-transmission times of the packets over the last K packets, (The inter-transmission time is the length of time separating the transmission of a packet from the transmission of the previous packet from same source.)
- **Metric 3**: Total number of packets that are transmitted in a time window with a duration of T.

We analyze the importance of each of the candidate metrics in Section III-C in order to determine an importance coefficient for each of these candidate metrics with respect to the effect on the detection of Mirai attacks.

A. Auto-Associative Dense RNN (AA-Dense RNN) Model

Next, we will describe an auto-associative network constructed via the Dense RNN model, which we call the AA-Dense RNN.

The Dense RNN model was introduced in [17], [25] as a mathematical model for neural networks with soma-to-soma interactions. In addition to the usual interactions between axons and dendrites, this model allows a direct soma-to-soma connectivity, so that firing at a given soma (i.e. neuron cell) can have three effects: induced direct firing at a neighbouring neuron, plus excitation and inhibition of any other cell in the network via excitatory and inhibitory weights. In the Dense RNN the soma-to-soma interactions are represented by the probability p that any other cell in the network fires when a given cell fires, that represents random saccades of cells which fire in unison. The Dense RNN that has been used so far also assumes that the internal structure of the network is homogenous, i.e. all the cells have the same connectivity. However this particular constraint can be relaxed if needed.

B. Clusters of Identical and Densely Connected Cells

Let us now consider the construction of a cluster that contains n identically connected cells, each of which has firing rate r, and receives external inhibitory and excitatory arrivals of spikes denoted by λ^- and λ^+, respectively. Since all the cells are identical, the state of each cell is denoted by q. We also assume that the total firing rate of each cell is r. Each cell also receives an *inhibitory input* from some cell u which does not belong to the network, and whose activation probability is q_u. Thus for any cell in our network we will have an inhibitory weight incoming $w_u^- > 0$ from the external cell u to this particular cell.

The dense network itself has no excitatory or inhibitory weights, but whenever a cell i fires, it triggers the firing of any other cell j at random with probability $\frac{p}{n-1}$ due to the *soma to soma* interactions, and creates a cascade of cells that fire until either a non-excited cell is reached so it cannot fire, or a cell in the chain sends an excitatory spike to another cell which does not fire. Since all the cells behave in a statistically identical manner, the probability q that a cell is excited satisfies the equation:

$$q = \frac{\lambda^+ + rq(n-1)\sum_{y=0}^{\infty}[\frac{p(n-2)}{n-1}]^y \ \frac{(1-p)}{n-1}}{r + \lambda^- + q_u w_u^- + rq(n-1)\sum_{y=0}^{\infty}[\frac{qp(n-2)}{n-1}]^y \ \frac{p}{n-1}} \quad (1)$$

which reduces to:

$$q = \frac{\lambda^+ + \frac{rq(n-1)(1-p)}{n-1-qp(n-2)}}{r + \lambda^- + q_u w_u^- + \frac{rqp(n-1)}{n-1-qp(n-2)}}, \quad (2)$$

which is a second degree polynomial in q. Since q is a probability, only its positive root(s) which are less than one are of interest.

When n is large, the expression (2) simplifies to:

$$q^2 p\lambda - q[\lambda + p(\lambda^+ + r)] + \lambda^+ = 0, \quad (3)$$

where $\lambda = \lambda^- + q_u w_u^-$.

C. Sructure of the Dense RNN used in this Work

We define the Dense RNN model by its inputs, outputs and its internal architecture. As shown in Figure 1, the input of the Dense RNN is the collection of the extracted metrics $\{x_{k-1}^i\}_{i\in\{1,\ldots,l\}}$ related to the transmission of packet $k-1$, and its output is the collection of the "normally expected metrics" $\{\hat{x}_k^i\}_{i\in\{1,\ldots,l\}}$ for the transmission of the following packet k. Note that l denotes the total number of metrics that are collected for the analysis.

Let X denote the input matrix whose entry (k,i) is x_k^i, and $\hat{X}$ denote the output matrix whose entry (k,i) is $\hat{x}_k^i$. Moreover, we let O_m denote the output vector of layer m and W_m denote the connection weight matrix between the layer m and layer $m+1$ for $m \in \{0,\ldots,L\}$, where $m=0$ is the input layer of the Dense RNN.

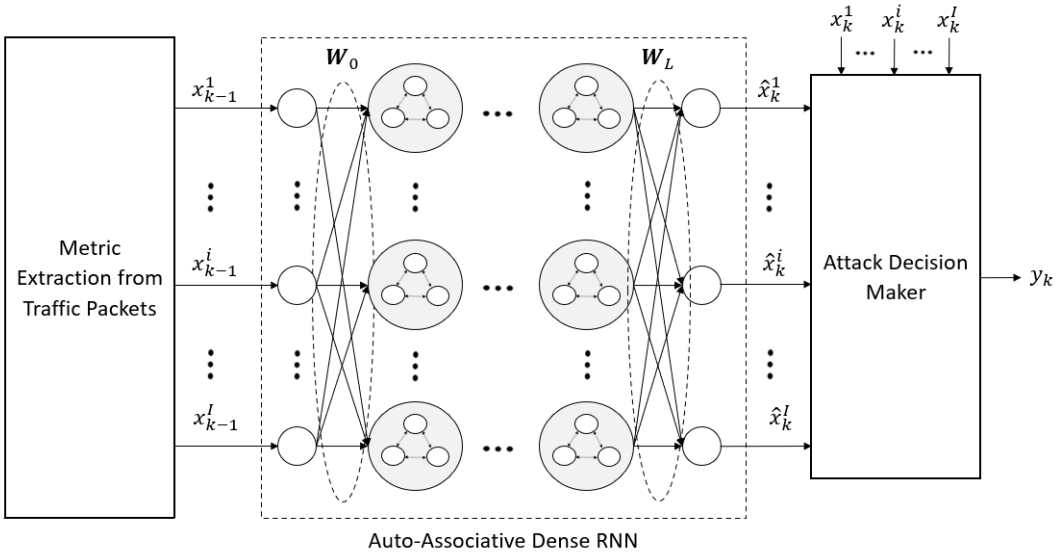

Fig. 1. Architecture of the Dense RNN based attack detector with its three modules: Metric Extraction from Traffic Packets, AA-Dense RNN and Attack Decision Maker.

Each hidden layer $m \in \{1, \ldots, L\}$ of the Dense RNN contains l Random Neural Network cell clusters each of whose probability of activation is denoted by $\zeta(x)$ which is the positive root obtained from the expression (**??**) with $x = q_u.w_u^-$.

Accordingly, for the given input matrix X, the forward pass of the Dense RNN is computed as:

$$O_0 = \min(X, 1), \tag{4}$$

$$O_l = \zeta(O_{l-1}W_{l-1}) \qquad \forall l \in \{1, \ldots, L\}, \tag{5}$$

$$\hat{X} = O_L W_L, \tag{6}$$

where $\zeta(\cdot)$ is a term-by-term activation function for vectors or matrices.

The connection weights of the Dense RNN are computed with an efficient training procedure which is developed in [17] that combines unsupervised and supervised learning. In order to create the auto-associative memory, we train the Dense RNN by using "only" the data of benign IoT traffic, and in Section III-G we show that the training time of the Dense RNN is low and competitive with the computational time associated with simpler models,

D. The Attack Decision Maker

Finally, the Attack Decision Maker module aims to give the final attack decision for the current data packet based on the actual and the predicted metrics of the packet. To this end, in this module, we calculate the absolute difference between the actual and the predicted value (which is the expected value for the normal traffic) of each metric and apply threshold on the difference as

$$d_k^i = |x_k^i - \hat{x}_k^i| \qquad \forall i \in \{1, \ldots, I\} \tag{7}$$

$$y_k = \mathbb{1}\Big[\sum_{i \in \{1,\ldots,I\}} \alpha_i.d_k^i \;\geq\; \Theta\Big] \tag{8}$$

where Θ is a threshold for the binary decision. Note that, clearly, the small values of Θ cause the false positive alarms while the large values of that cause false negative alarms. In addition, α_i is an coefficient for the attack decision with respect to the Metric i, and $\sum_{i \in \{1,\ldots,I\}} \alpha_i = 1$. Furthermore, $\mathbb{1}\big[\Xi\big] = 1$ if Ξ is a true statement and $\mathbb{1}\big[\Xi\big] = 0$ otherwise.

III. Experimental Results

In order to evaluate the performance of our attack detection method, we use the Mirai botnet attack data from the publicly available Kitsune dataset [26], [27]. This dataset contains $764,137$ packet transmissions including both normal and attack traffic. We use *only* 70 % of the normal traffic packets for training and all of the packets (both normal and attack traffic) for the test of the attack detector.

A. Parameters of the AA-Dense RNN

To implement the AA-Dense RNN, we use a two-layer Dense RNN (i.e. $L = 2$), with $n_l = I \;\forall l \in \{1, \ldots, L\}$, where $I = 3$, and recall that I is the total number of metrics that are being used. In addition, we set $p = 0.05$, $r = 0.001$ and $\lambda^+ = \lambda^- = 0.1$. Moreover, we set $K = 500$ packets and $T = 100secs$ in the Metric Extraction module.

B. Comparison with other Techniques

We first selected the Simple Threshold method as the simplest benchmark for attack detection. In this method, we basically use (8) by replacing d_k^i with x_k^i. In order to achieve the best performance of this method, we search for the best value of Θ on the test set which includes both normal and attack traffic.

1) Least Absolute Shrinkage and Selector Operator (Lasso): We selected the Lasso as the linear model that replaces the AA-Dense RNN in the proposed method in Figure 1. To this end, we create an auto-associative memory based on Lasso by training it on 70 % of the normal traffic. For the implementation of Lasso, we use the scikit-learn library [28] with "alpha = 0.1".

2) K-Nearest Neighbours Regressor (KNN): We use KNN to replace the AA-Dense RNN module in Figure 1. Similar with other methods, we trained KNN on 70 % of the normal traffic. In addition, for the implementation of this model, we use the scikit-learn library with "n_neighbours = I".

C. Importance Analysis of the Metric Candidates to Determine the Value of the α_is.

We now aim to analyze how important each feature candidate for the detection of Mirai botnet attacks in the considered dataset. To this end, we first perform the following analysis:

1) Pearson Correlation: For each Metric i, we compute Pearson correlation coefficient [29] between that metric and the attack label. This coefficient measures the strength and the direction of the linear relationship between the considered metric and the attack label. Since we desire to measure only the importance of Metric i for the detection of attack, we need only the strength of the relationship so we let ρ_i denote the absolute value of the coefficient for Metric i.

2) ANOVA: We compute the other coefficients as the F-ratios [30] that are calculated via the Analysis of Variance (ANOVA) method. The value of the F-ratio corresponding to Metric i measures the statistical significance of that metric for the decision of attack. We let f_i denote the normalized F-ratio for Metric i in the range $[0, 1]$.

After we compute the coefficients via each of the Pearson correlation coefficient and ANOVA, we calculate the importance coefficients of the metrics as

$$\alpha_i = \frac{\rho_i + f_i}{\sum_{i\in\{1,\dots,I\}} (\rho_i + f_i)} \qquad \forall i \in \{1, \dots, I\}. \tag{9}$$

In Figure 2, we present the value of α_i as well as the values of ρ_i and f_i for each Metric i, $i \in \{1, 2, 3\}$. In this figure, we see that the importance of each of Metrics 1 and 3 is higher than that of the Metric 2 with respect to each of α_i, ρ_i and f_i. In addition, the values of α_1, ρ_1 are close to those of α_3, ρ_3 although there is a significant gap between f_1 and f_3.

D. Performance Evaluation of AA-Dense RNN with respect to the Selection of Metrics

In Figure 3, we show the performance of the proposed attack detection method with the selection of different metrics, as well as the combination of all metrics, and see that AA-Dense RNN achieves the highest accuracy at 99.84% when we use the weighted combination of Metric 1, Metric 2, and Metric 3, as we do for the performance evaluations in the rest of this paper.

The high detection performance result shows that the AA-Dense RNN is able to classify normal and malicious traffic although it has been trained with only normal traffic.

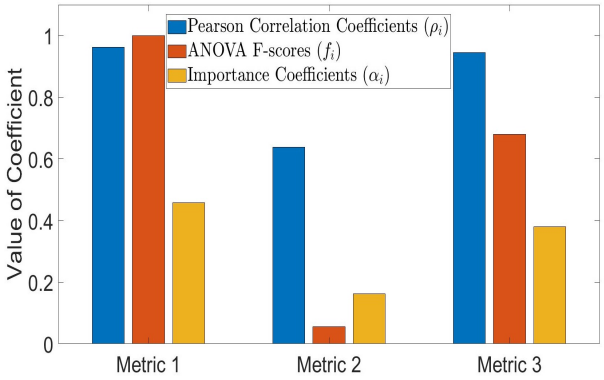

Fig. 2. Pearson correlation coefficient ρ_i, normalized by coefficient ANOVA f_i and importance coefficent α_i for each Metric i, $i \in \{1, 2, 3\}$.

Moreover, our results show that the accuracy of the AA-Dense RNN is more than 95% under the selection of any metric. In addition, we observe a close relationship between the importance coefficients of metrics in Figure 2 and the performance of the AA-Dense RNN detector in Figure 3.

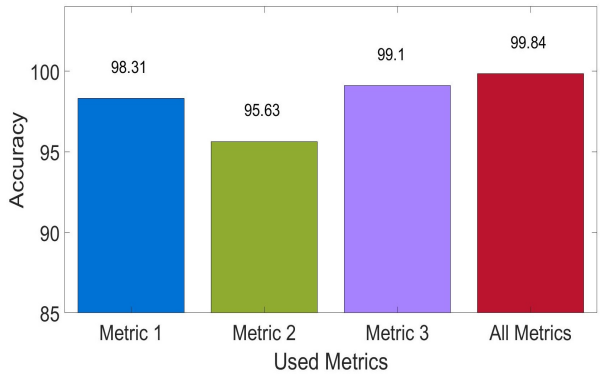

Fig. 3. Performance of the AA-Dense RNN under each of Metric 1, Metric 2, Metric 3, and the α_i weighted combination of all metrics.

the

E. Selection of the Value of Threshold Θ

We now analyze the performance of the AA-Dense RNN with respect to the value of threshold Θ. Figure 4 presents the True Positive and True Negative percentages with respect to the increasing value of Θ from 0 to 0.5 with 0.01 increments.

In Figure 4, we see that the AA-Dense RNN detector is highly robust with respect to $\Theta \in [0.01, 0.25]$. Thus, in the practical usage of the proposed method, we may select any value of Θ in the range $[0.01, 0.25]$ without a significant performance loss. In addition, in this range the AA-Dense RNN is fair in detecting both attack and normal traffic, and it achieves high performance for both.

F. Comparison of the AA-Dense RNN's Performance with KNN and Lasso

Let us now In this compare the attack detection performance of the AA-Dense RNN with the Simple Thresholding, Lasso,

TABLE I
COMPARISON OF ATTACK DETECTION METHODS WITH RESPECT TO ACCURACY AS WELL AS EACH OF THE TRUE POSITIVE, FALSE NEGATIVE, TRUE NEGATIVE AND FALSE POSITIVE PERCENTAGES

Attack Detection Methods	Accuracy	True Positive	False Negative	True Negative	False Positive
AA-Dense RNN	99.84	99.82	0.18	99.98	0.02
KNN	99.79	99.79	0.21	99.75	0.25
Lasso	99.78	99.75	0.25	99.95	0.05
Simple Thresholding	93.18	93.09	6.94	93.63	6.37

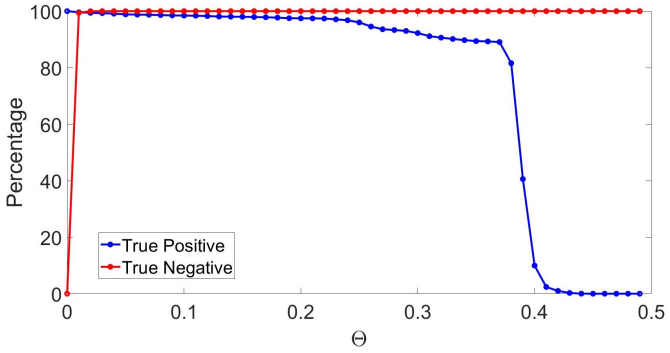

Fig. 4. True Postive and True Negative percentages of the AA-Dense RNN for the increasing value of Θ.

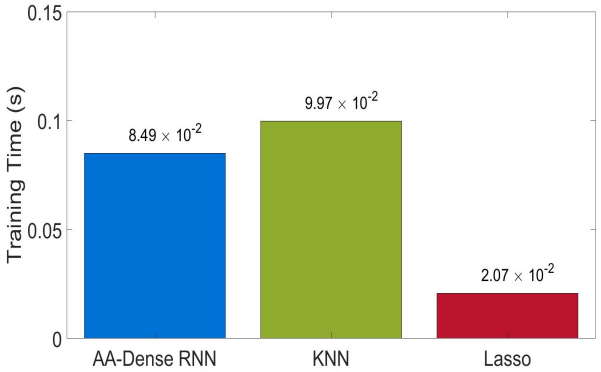

Fig. 5. Training times of the different attack detection methods.

and KNN methods, where both the Lasso and KNN are trained as auto-associative memories.

In Table I, we present the comparison of the detection methods with respect to each of the accuracy and percentages of true positive, false negative, true negative and false positive. The detection methods in this table are placed in descending order with respect to their accuracy. ur results show that AA-Dense RNN attack detection significantly outperforms the other methods with respect to accuracy. In addition, we see that this auto-associative network achieves much higher accuracy than Simple Thresholding. We see that the AA-Dense RNN achieves 99.82% true positive and 99.98% true negative acuracy, higher than the other methods. Among all the methods, the Lasso obtains the true negative percentage closest to the AA-Dense RNN, and significantly higher than KNN and Simple Thresholding.

G. Computation Time

We now compare the AA-Dense RNN with KNN and Lasso with respect to the training and execution times, both being measured on a workstation with 32 Gb RAM and an AMD 3.7 GHz (Ryzen 7 3700X) processor.

Figure 5 shows the training time of each of the AA-Dense RNN, KNN, and Lasso models, where the training is performed for 70 % of the normal traffic (83138 samples). While the attack detector may be trained offline in real-life usage, the training time is not a major issue as long as it is acceptable. In the sae figure we see that the training time of the AA-Dense RNN is less than 0.1 sec which is highly acceptable. In addition, the training time of the AA-Dense RNN is significantly less than that of KNN; however, it is higher than that of the Lasso method.

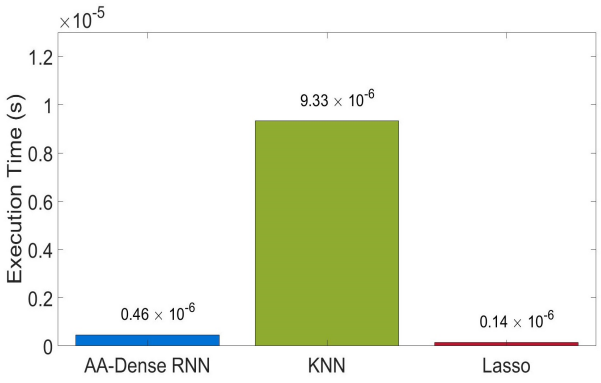

Fig. 6. Execution times of the different attack detection methods.

Figure 6 shows the execution time of each of the AA-Dense RNN, KNN, and Lasso models for the classification, evaluated per single traffic packet. We see that the execution time of the AA-Dense RNN detector is around 0.5 μ secs. While that of all other methods is less than 10 μ secs, the KNN's execution

time is quite high, and LASSO's execution time is the shortest. This shows that the AA-Dense RNN and LASSO detectors are suitable for use in real-time attack detection.

IV. Conclusions and Future Work

IoT devices are often rapidly installed with known factory parameters, that attract Mirai-type Botnet attacks. Therefore we have introduced an attack detection scheme for the IoT using the Auto-Associative Dense Random Neural Network (AA-Dense RNN) for Mirai-like Botnets.

The approach has the added advantage that it is trained with normal traffic to detect attack traffic, and it compares favorably with two known ML techniques: the Least Absolute Shrinkage and Selector Operator (Lasso) and the K-Nearest Neighbours (KNN), as well as with a simple thresholding technique.

Our experimental results on a publicly available dataset containing $764,137$ packet transmissions, show that the method introduced in this paper achieves 99.84% accuracy with 99.82% true positive and 99.98% true negative rates, which is much better than KNN detection and better than Lasso. Both the AA-Dense RNN and Lasso have training and testing times that are shorter than KNN.

The computation times and accuracies of AA-Dense RNN and Lasso are well within the needs of real-time on-the-fly lightweight attack detection, with AA-Dense RNN being best for accuracy and Lasso being best for computation times.

Future work will further evaluate the performance of the proposed attack detector on other available Botnet datasets, and extend the design to detect different attacks with a single AA-Dense RNN detector that is trained on benign traffic.

References

[1] J. Biggs, "Hackers release source code for a powerful DDoS app called Mirai," *TechCrunch*, October 2018. [Online]. Available: https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/

[2] R. Hackett, "Why a hacker dumped code behind colossal website-trampling botnet," October 2016.

[3] N. Statt, "How an army of vulnerable gadgets took down the web today," October 2016. [Online]. Available: https://www.theverge.com/2016/10/21/13362354/dyn-dns-ddos-attack-cause-outage-status-explained

[4] A. S. Feily qnd Maryam and S. Ramadass, "A survey of botnet and botnet detection," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, 2009.

[5] D. Goodin, "100,000-strong Botnet built on router 0-day could strike at any time," *Ars Technica*, December 2017. [Online]. Available: https://arstechnica.com/information-technology/2017/12/100000-strong-botnet-built-on-router-0-day-could-strike-at-any-time/

[6] J. Dibbelt et al., "Eco-aware vehicle routing in urban environments," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 208–213.

[7] K. Fotiadou et al., "Network traffic anomaly detection via deep learning," *Information*, vol. 12, no. 5, 2021. [Online]. Available: https://www.mdpi.com/2078-2489/12/5/215

[8] C. Cimpanu, "New Mirai variant focuses on turning IoT devices into proxy servers," *Bleeping Computer*, February 2018. [Online]. Available: https://www.bleepingcomputer.com/news/security/new-mirai-variant-focuses-on-turning-iot-devices-into-proxy-servers/

[9] M. Antonakakis et al., "Understanding the Mirai Botnet," in *Proceedings of the 26th USENIX Security Symposium*, 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[10] A. Roukounaki et al., "Scalable and configurable end-to-end collection and analysis of iot security data : Towards end-to-end security in iot systems," in *2019 Global IoT Summit (GIoTS)*, 2019, pp. 1–6.

[11] A. Kumar and T. J. Lim, "Early detection of mirai-like iot bots in large-scale networks through sub-sampled packet traffic analysis," *CoRR*, vol. abs/1901.04805, 2019. [Online]. Available: http://arxiv.org/abs/1901.04805

[12] G. Baldini et al., "Iot network risk assessment and mitigation: The SerIoT Approach," in *Security Risk Management for the Internet of Things: Technologies and Techniques for IoT Security, Privacy and Data Protection*, J. Soldatos, Ed. Now Publishers, 2020, p. 88–104.

[13] Z. Ahmed, S. M. Danish, H. K. Qureshi, and M. Lestas, "Protecting IoTs from Mirai Botnet attacks using blockchains," in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2019, pp. 1–6.

[14] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, no. 4, pp. 502–510, 1989.

[15] C. E. Cramer and E. Gelenbe, "Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 2, pp. 150–167, 2000.

[16] E. Gelenbe and T. Koçak, "Area-based results for mine detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 1, pp. 12–24, 2000.

[17] E. Gelenbe and Y. Yin, "Deep learning with dense random neural networks," in *International Conference on Man–Machine Interactions*. Springer, 2017, pp. 3–18.

[18] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against iot-connected home environments," *Procedia Computer Science*, vol. 134, p. 458–463, 2018.

[19] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1071–1083, 2002.

[20] H. Cancela, F. Robledo, and G. Rubino, "A grasp algorithm with rnn based local search for designing a wan access network," *Electronic Notes in Discrete Mathematics*, vol. 18, pp. 59–65, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571065304010674

[21] E. Gelenbe and A. Stafylopatis, "Global behavior of homogeneous random neural systems," *Applied mathematical modelling*, vol. 15, no. 10, pp. 534–541, 1991.

[22] A. Javed, H. Larijani, A. Ahmadinia, and D. Gibson, "Smart random neural network controller for hvac using cloud computing technology," *IEEE Transactions on Industrial Informatics*, vol. 13, pp. 351–360, 2017.

[23] E. Gelenbe, "Steady-state solution of probabilistic gene regulatory networks," *Physical Review E*, vol. 76, no. 3, p. 031903, 2007.

[24] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural network architectures for the detection of syn flood attacks in IoT systems," in *Proc. 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2020, pp. 1–4.

[25] E. Gelenbe and Y. Yin, "Deep learning with random neural networks," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 1633–1638.

[26] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *The Network and Distributed System Security Symposium (NDSS)*, 2018.

[27] "Kitsune Network Attack Dataset," August 2020. [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, no. 347-352, pp. 240–242, 1895.

[30] R. G. Lomax, *Statistical concepts: A second course*. Lawrence Erlbaum Associates Publishers, 2007.