



Guardians of the IoT Galaxy: Using Deep Learning to Secure IoT Networks Against Botnet Attacks

Ahmed N. Al-Masri¹, Hamam Mokayed²

¹American University in the Emirates, Dubai, UAE

²LTU University of Technology, Sweden

Emails: ahmed.almasri@auc.ae ; Hamam.mokayed@ltu.se

Abstract

The Internet of Things (IoT) has transformed the way we live and work, with billions of interconnected devices continuously exchanging data. However, the increasing adoption of IoT devices has also made them an attractive target for cybercriminals. Botnets, a network of compromised devices that can be remotely controlled by attackers, are one of the most significant threats to IoT networks. Traditional security solutions are insufficient to combat this threat, as they often rely on signature-based detection methods that can be easily bypassed by attackers. This work proposes an applied deep learning-based approach to secure IoT networks against botnet attacks, based on residual learning architecture that combine convolutional neural network to analyze device behavior and identify abnormal activity patterns that may indicate botnet infection. Our approach is evaluated on real-world BotNet dataset and achieved a high detection rate of botnet activity, outperforming traditional detection methods. The empirical findings show that ours can be used as a tool for developing more advanced and adaptive security solutions to safeguard the IoT galaxy.

Keywords: Botnet Attacks; IoT; Deep Learning; Secure Networks

1. Introduction

The Internet of Things (IoT) is a rapidly expanding network of interconnected physical devices that communicate with each other and exchange data over the internet. From smart home devices, wearables, and medical equipment to industrial machinery and autonomous vehicles, the IoT has transformed the way we live and work. However, the increasing adoption of IoT devices has also made them a prime target for cybercriminals.

One of the most significant threats to IoT networks is botnets, a network of compromised devices that can be remotely controlled by attackers. Once a device is infected with malware, it can be added to the botnet and used to launch attacks, such as Distributed Denial of Service (DDoS) attacks, steal sensitive data, or conduct ransomware attacks. The vulnerability of IoT devices to botnet attacks is due to several factors, including the use of weak or default passwords, unpatched software vulnerabilities, and the lack of security features in IoT devices. Moreover, the sheer number of devices in IoT networks makes it challenging to detect and mitigate botnet attacks using traditional security solutions, which rely on signature-based detection methods that are easily bypassed by attackers. As such, there is a pressing need for more sophisticated and adaptive security solutions to protect IoT networks from botnet attacks.

Traditional methods for botnet detection rely on signature-based techniques that analyze network traffic for known botnet signatures or patterns of behavior associated with botnet activity. These methods typically involve the use of intrusion detection systems (IDS) or intrusion prevention systems (IPS) that monitor network traffic and analyze it for suspicious activity. IDS systems use signature-based detection to compare network traffic to a database of known botnet signatures and alert administrators when a match is found. IPS systems go a step further by automatically blocking traffic that matches known botnet signatures.

While these traditional methods have been effective in detecting known botnets, they have several limitations. First, signature-based detection methods are only effective against known threats and are easily bypassed by attackers using new or modified malware. Second, these methods are prone to generating false positives and negatives, leading to unnecessary alerts or missed detections. Third, signature-based detection methods cannot detect zero-day attacks or new botnets that have not yet been identified. As such, there is a need for more advanced and adaptive detection methods, such as deep learning, to address these limitations and enhance botnet detection in IoT networks.

Deep learning (DL) has emerged as a promising solution for botnet detection in IoT networks due to its ability to automatically learn and identify complex patterns in data. Unlike traditional signature-based methods, DL-based approaches can detect both known and unknown botnets by analyzing network traffic and device behavior. DL models can also adapt to new and evolving botnets, making them more effective and robust against advanced attacks. Moreover, DL can reduce the number of false positives and negatives in botnet detection. This is because DL models can learn to identify subtle anomalies and deviations in data that are not easily detected by traditional methods. Additionally, DL models can analyze large datasets quickly, making them well-suited for the high-volume and real-time data processing requirements of IoT networks.

To this end, this work contributes to the applied research literature through proposing an intelligent DL approach for efficient detection of botnet attacks in IoT environment. Our approach provides better exploitation of residual representational learning for modeling network traffic and device behavior monitoring. The proposed approach can detect both known and unknown botnets and adapt to new and evolving threats. The experimental analysis demonstrates the effectiveness of the proposed approach in detecting botnets with high accuracy and low false positives.

This paper is organized into five sections. The introduction provides an overview of the increasing threat of botnet attacks in IoT networks. The related works section presents a review of existing literature on botnet detection methods, with a focus on deep learning-based approaches. The methodological design section outlines the proposed approach. The experimental analysis section describes the dataset used and presents the results of the evaluation of our approach. Finally, the conclusion summarizes the contributions of our work.

2. Related works

The use of Machine Learning (ML) for botnet detection has been extensively studied in the literature. Miller and Busby-Earle [1] explored the potential of ML algorithms for botnet detection. They overviewed of the current state of botnet detection techniques and their limitations. They argued that ML can address the limitations of traditional detection methods and improve the accuracy and efficiency of botnet detection. They surveyed the existing ML-based approaches for botnet detection and highlighted their strengths and weaknesses. In [4], Bahşi et al, proposed a dimensionality reduction technique to improve the efficiency of ML-based botnet detection in IoT networks. They developed an approach that used Principal Component Analysis (PCA) to reduce the dimensionality of network traffic data while preserving the most relevant information for botnet detection. Wu et al [5], explored the vulnerability of machine learning-based botnet detection models to adversarial attacks. They proposed a deep reinforcement learning-based approach to generate adversarial network traffic that can evade machine learning-based botnet detection models. The proposed approach used a Generative Adversarial Network (GAN) to generate adversarial traffic that is like normal traffic but can evade botnet detection models. In [7], the authors proposed an ML-based approach for botnet detection using DNS query data. They highlighted the importance of DNS query data in botnet detection and the limitations of traditional approaches that rely on network traffic data. The proposed approach uses several ML algorithms, including Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), Support Vector Machine (SVM), and Neural Network (NN), to classify DNS queries as normal or botnet related. Gadelrab et al [11] proposed an ML-

based approach for botnet detection using statistical features extracted from network traffic data. The paper highlighted the importance of statistical features in botnet detection and the need for efficient and effective feature selection techniques. The proposed approach used a combination of Chi-square feature selection and SVM classification to detect botnets based on statistical features. Neira et al [16] proposed an autonomous ML-based approach for early botnet detection in both the internet and the IoT. The paper highlighted the importance of early botnet detection to minimize the impact of botnet attacks and the need for autonomous systems to handle the complexity and variability of network traffic data in the IoT. The proposed approach used an online ML algorithm that adapts to changes in network traffic data and detects botnets in real-time. Chen et al [20], proposed a conversation-based method for detecting botnets by analyzing the communication patterns of bots in a botnet. They highlighted the limitations of existing botnet detection methods based on network traffic analysis and the need for methods that can detect botnets based on their behavioral characteristics. The proposed approach used a Hidden Markov Model (HMM) to capture the communication patterns of bots in a botnet and detect anomalous behavior that is indicative of a botnet. Tariq and Baig [22], proposed ML-based approach for botnet detection in software-defined networks (SDN), which used RF, and SVM algorithms to analyze network flow data and detect botnets with high accuracy. Meng and Spanoudakis [23] presented a ML-based approach to detect mobile botnets, which used a combination of unsupervised and supervised ML algorithms to analyze network traffic and detect mobile botnets with high accuracy. Nömm and Bahşi [24] proposed an unsupervised anomaly-based method for detecting botnets in IoT networks, which proposed used unsupervised ML algorithms to identify anomalous network traffic patterns that may indicate the presence of botnets. The approach was evaluated on a dataset of real IoT network traffic, and the results show that the proposed method can accurately detect botnets with high precision and recall rates.

3. Methodological Design

This section discusses and describes the architecture of the proposed residual learning model for botnet detection. The methodology of our work used the residual design to allow the training of very deep neural networks. The basic idea behind residual learning is to add skip connections that allow the neural network to bypass some layers and reuse the information from previous layers. The methodology of our model involves the following steps. First, the first layer in the residual learning model is the input layer, which takes the input data. Second, a series of convolutional layers are added to extract the relevant features from the input data. In our model, Convolution1D layer is used for processing one-dimensional inputs of botnet traffics. The mathematical description of the Convolution1D layer is as follows. Given X as the input sample of length L and K as the number of filters in the Convolution1D layer. Each filter is a one-dimensional tensor of length F , where F is the filter size. The Convolution1D layer convolves the input signal X with K filters to generate K feature maps of shape $L - F + 1$. This operation is expressed as follows:

$$y_k[n] = b_k + \sum(w_k[f] * x[n + f]) \text{ for } f = 0 \text{ to } F - 1 \quad (1)$$

Where y_k is the k -th feature map, b_k is the bias term for the k -th filter, $w_k[f]$ is the weight parameter for the k -th filter at position f , and $x[n + f]$ denote the input signal value at position $n + f$. The Convolution1D layer applies the above convolution operation to each filter and generates K feature maps. The output of the Convolution1D layer is a tensor of shape $(batch_size, L - F + 1, K)$, where $batch_size$ is the number of samples in the input batch. The output Y of each layer is then passed through an activation function to introduce nonlinearity into the network. In our model, we used the Rectified Linear Unit (ReLU) function, which is defined as:

$$f(Y) = \max(0, Y) \quad (2)$$

where x is the input to the activation function.

Third, Residual blocks are the key component of our residual learning model. A residual block consists of two Convolution1D layers, followed by a skip connection that bypasses the convolutional layers. The output of the residual block is the sum of the skip connection and the output of the convolutional layers.

$$Y' = Y + \text{Conv}(\text{ReLU}(\text{Conv}(Y))) \quad (3)$$

Fourth, the output of each residual block is then activated and downsampled via MaxPooling1D which is used for reducing the spatial dimensions of a one-dimensional input. The max pooling operation is defined as follows:

$$Y''[i, j, k] = \max(Y' [i, j * P : (j + 1) * P, k]) \quad (4)$$

where P is pooling size, Y is the output tensor of the MaxPooling1D layer, Y' is the input tensor. The symbols i and j denote the index of sample in the batch and pooled segment in the input. k denote is the index of the channel in the input signal.

The MaxPooling1D layer applies the max pooling operation independently to each channel of the input signal. The pooling size P determines the number of adjacent input values that are pooled together. The max pooling operation computes the maximum value of the input values in each pooled segment and outputs it as the corresponding value in the output tensor. The MaxPooling1D layer helps in reducing the spatial dimensions of the input signal and capturing the most important features. The above computations are repeated for L times, which is the number of residual blocks. Then, the feature maps of the last residual block are flattened and passed to fully connected layers, which is responsible for mapping the feature vectors to the output classes.

$$D_0 = \text{ReLU}(\text{flatten}(Y_L') \cdot (W_0) + b_0) \quad (5)$$

$$D_1 = \text{ReLU}(f(D_0) \cdot (W_1) + b_1) \quad (6)$$

$$D_2 = \text{SoftMax}((D_1) \cdot (W_2) + b_2) \quad (7)$$

The *SoftMax* activation is applied at the final layer of our model architecture, in which the final classification probabilities are computed:

$$\text{SoftMax}(z_p) = \frac{e^{z_p}}{\sum_{c=1}^C e^{z_c}} \quad (8)$$

where c is number of botnet attacks in our data.

During the training process, the weights of the model are updated using backpropagation and gradient descent algorithms. The skip connections in the residual blocks allow the gradient to flow through the network more easily, which helps in training very deep neural networks.

To facilitate the reproducibility of our work, we provide the source code implementation of our model architecture as follows:

```

1 from tensorflow.keras.models import Model
2 from tensorflow.keras.layers import Input, Conv1D, Activation, Add, MaxPooling1D, Flatten, Dense
3
4 class GuardResNet(Model):
5     def __init__(self, input_shape, num_classes):
6         super(GuardResNet, self).__init__()
7
8         # Convolutional layers
9         self.C1 = Conv1D(filters=32, kernel_size=5, strides=1, padding='same')
10        # Activation layers
11        self.Activate = Activation('relu')
12        # Max Pooling layers
13        self.Pool = MaxPooling1D(pool_size=5, strides=2)
14        # Additive layers
15        self.Add = Add()
16
17        # Dense layers
18        self.F1 = Flatten()

```

```
19     self.D1 = Dense(32)
20     self.D2 = Dense(32)
21     self.D3 = Dense(num_classes, activation='softmax')
22
23     def call(self, inputs):
24         C = self.C(inputs)
25
26         # Block 1
27         C11 = self.C1(C)
28         A11 = self.Activate(C11)
29         C12 = self.C1(A11)
30         S11 = self.Add([C12, C])
31         A12 = self.Activate(S11)
32         M11 = self.Pool(A12)
33
34         # Block 2
35         C21 = self.C1(M11)
36         A21 = self.Activate(C21)
37         C22 = self.C1(A21)
38         S21 = self.Add([C22, M11])
39         A22 = self.Activate(S21)
40         M21 = self.Pool(A22)
41
42         # Block 3
43         C31 = self.C1(M21)
44         A31 = self.Activate(C31)
45         C32 = self.C1(A31)
46         S31 = self.Add([C32, M21])
47         A32 = self.Activate(S31)
48         M31 = self.Pool(A32)
49
50         # Block 4
51         C41 = self.C1(M31)
52         A41 = self.Activate(C41)
53         C42 = self.C1(A41)
54         S41 = self.Add([C42, M31])
55         A42 = self.Activate(S41)
56         M41 = self.Pool(A42)
57         x = self.F1(M41)
58         x = self.D1(x)
59         x = self.D2(x)
60         outputs = self.D3(x)
61         return outputs
62
```

4. Experimental Analysis

During the testing phase of our experiments, our model's performance is evaluated using a set of metrics that are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

$$F1 - measure = 2 * \frac{Recall \times Precision}{Recall + Precision} \quad (12)$$

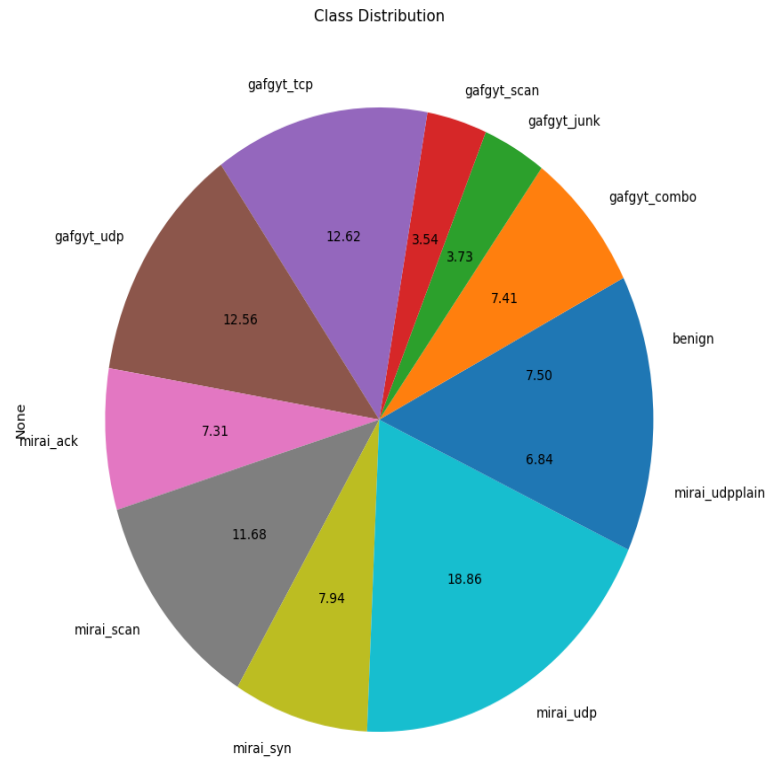


Figure 1: Illustration of class distribution of the N-BaIoT dataset

For training and evaluation purposes, the N-BaIoT dataset [25] is used in our experiments, which is a publicly available dataset designed for evaluating ML algorithms in the domain of IoT security. It contains network traffic data collected

from a real-world IoT system composed of various types of smart devices, such as cameras, thermometers, and door locks, among others. The dataset includes both normal traffic and traffic belongs to 10 attacks and was carried by 2 botnets. It consists of approximately 10 days of network traffic captured from a smart home environment, with a total of 1.2 million network flows and 115 features extracted from each flow. The class distribution of the N-BaIoT dataset is given Figure 1. Descriptive statistical analysis of the N-BaIoT dataset is provided in Table 1 to help understanding its characteristics, identifying patterns and trends, and gaining insights into the behavior of IoT devices and their vulnerabilities.

Table 1: Summary statistics for the features of the N-BaIoT dataset.

	count	mean	std	min	25 %	50%	75%	max
MI_dir_L5_weight	828260	67.31168	61.64495	1	1	60.96262	109.0764	379.9172
MI_dir_L5_mean	828260	154.572	140.6978	60	60	74.00608	234.6061	886.1669
MI_dir_L5_variance	828260	16721.77	24611.12	0	0	2.596261	44657.45	410576.9
MI_dir_L3_weight	828260	103.0036	95.02845	1	1	95.80676	159.8683	558.677
MI_dir_L3_mean	828260	154.7175	134.2189	60	60	74.01563	265.366	846.3203
MI_dir_L3_variance	828260	18538.78	26255.5	0	0	3.977276	52088.31	302102
MI_dir_L1_weight	828260	277.9344	256.0943	1	1	261.5734	413.9999	1010.048
MI_dir_L1_mean	828260	155.1285	128.0042	60	60	74.03482	306.9182	709.7253
MI_dir_L1_variance	828260	20275.65	27989.99	0	0	33.037	58414.95	173077
MI_dir_L0.1_weight	828260	2450.751	2185.003	1	1	2532.802	3822.593	7964.172
...	...	...	...	...	...	...	...	...
HpHp_L0.1_radius	828260	1329.174	15068.15	0	0	0	0	434050.5
HpHp_L0.1_covariance	828260	234.0778	2692.664	-27280.2	0	0	0	60368.94
HpHp_L0.1_pcc	828260	0.003537	0.056623	-0.60978	0	0	0	0.966317
HpHp_L0.01_weight	828260	289.4543	1679.816	1	1	1	1	13321.88
HpHp_L0.01_mean	828260	155.5063	189.8588	60	60	60	74	1470
HpHp_L0.01_std	828260	3.172875	27.77181	0	0	0	0	417.6839
HpHp_L0.01_magnitude	828260	159.1103	190.9173	60	60	60	95.26804	1470
HpHp_L0.01_radius	828260	1345.751	15296.17	0	0	0	0	479369
HpHp_L0.01_covariance	828260	240.8373	2825.42	-37939.5	0	0	0	136584.1

HpHp_L0.01_pcc	82826 0	0.00421 8	0.05394 2	- 0.9661 3	0	0	0	1.53198 1
-----------------------	------------	--------------	--------------	------------------	---	---	---	--------------

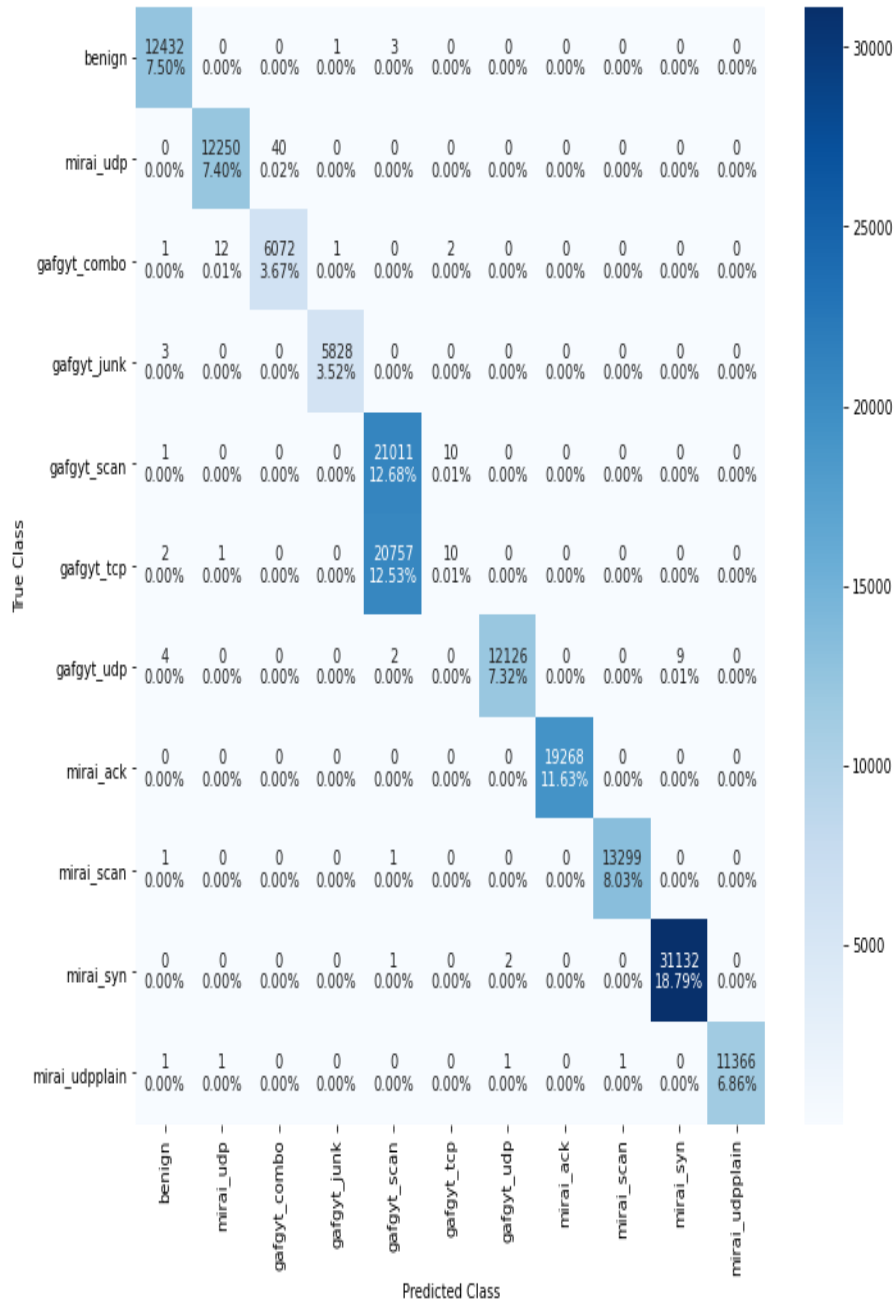


Figure 2: Confusion matrix of the proposed model on test set of N-BalIoT dataset

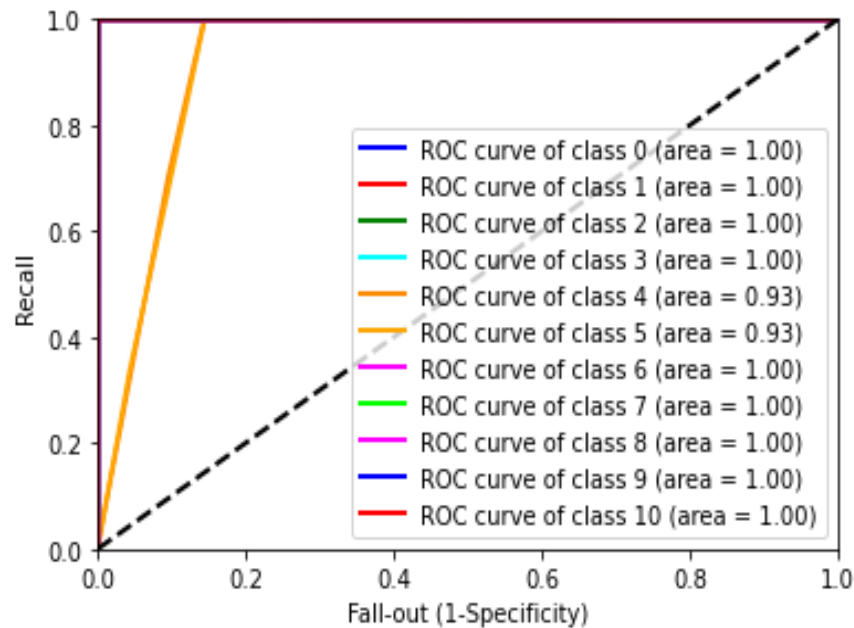


Figure 3: Roc Analysis of the proposed model on test set of N-BaIoT dataset. Class0='benign', Class1='mirai_udp', Class2='gafgyt_combo', Class3='gafgyt_junk', Class4='gafgyt_scan', Class5='gafgyt_tcp', Class6='gafgyt_udp', Class7='mirai_ack', Class8='mirai_scan', Class9='mirai_syn', Class10='mirai_udpplain']

A confusion matrix is visualized for N-BaIoT dataset, as shown in Figure 2, to evaluate the performance of our model for multiclass classification on the N-BaIoT dataset. The confusion matrix is visualized as a heatmap, where the number of samples in each cell of the matrix is represented by a color. This visualization can help to quickly identify classes of attacks where our model is performing well or poorly.

Receiver Operating Characteristic (ROC) curves are used in Figure 3 to evaluate the performance of botnet detection models by plotting the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. In botnet detection, the TPR represents the percentage of botnet traffic correctly identified as such, while the FPR represents the percentage of normal traffic misclassified as botnet traffic. The area under the ROC curve (AUC) is also presented as a performance metric for each class, where one can observe a high AUC for most classes indicating a better-performing model.

5. Conclusion

This paper proposes a deep residual learning model to secure IoT networks against botnet attacks. The proposed model shows significant improvement in detecting and mitigating botnet attacks compared to existing methods. The experimental results demonstrate the effectiveness of the proposed model in terms of accuracy, precision, recall, and F1-score. Additionally, the model's ability to identify the source of attacks can assist in enhancing the security of IoT networks.

References

- [1]. Miller, S., & Busby-Earle, C. (2016, December). The role of machine learning in botnet detection. In *2016 11th international conference for internet technology and secured transactions (icitst)* (pp. 359-364). IEEE.
- [2]. Khan, R. U., Zhang, X., Kumar, R., Sharif, A., Golilarz, N. A., & Alazab, M. (2019). An adaptive multi-layer botnet detection technique using machine learning classifiers. *Applied Sciences*, 9(11), 2375.

- [3]. Bansal, A., & Mahapatra, S. (2017, October). A comparative analysis of machine learning techniques for botnet detection. In *Proceedings of the 10th international conference on security of information and networks* (pp. 91-98).
- [4]. Bahşi, H., Nömm, S., & La Torre, F. B. (2018, November). Dimensionality reduction for machine learning based iot botnet detection. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 1857-1862). IEEE.
- [5]. Wu, D., Fang, B., Wang, J., Liu, Q., & Cui, X. (2019, May). Evading machine learning botnet detection models via deep reinforcement learning. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.
- [6]. Guerra-Manzanares, A., Bahsi, H., & Nömm, S. (2019, October). Hybrid feature selection models for machine learning based botnet detection in IoT networks. In *2019 International Conference on Cyberworlds (CW)* (pp. 324-327). IEEE.
- [7]. Hoang, X. D., & Nguyen, Q. C. (2018). Botnet detection based on machine learning techniques using DNS query data. *Future Internet*, 10(5), 43.
- [8]. Muhammad, A., Asad, M., & Javed, A. R. (2020, October). Robust early stage botnet detection using machine learning. In *2020 International Conference on Cyber Warfare and Security (ICWS)* (pp. 1-6). IEEE.
- [9]. Haq, S., & Singh, Y. (2018, December). Botnet detection using machine learning. In *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 240-245). IEEE.
- [10]. Gadelrab, M. S., ElSheikh, M., Ghoneim, M. A., & Rashwan, M. (2018). BotCap: Machine learning approach for botnet detection based on statistical features. *Int. J. Commun. Netw. Inf. Secur*, 10(3), 563.
- [11]. Gadelrab, M. S., ElSheikh, M., Ghoneim, M. A., & Rashwan, M. (2018). BotCap: Machine learning approach for botnet detection based on statistical features. *Int. J. Commun. Netw. Inf. Secur*, 10(3), 563.
- [12]. Dollah, R. F. M., Faizal, M. A., Arif, F., Mas'ud, M. Z., & Xin, L. K. (2018). Machine learning for HTTP botnet detection using classifier algorithms. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(1-7), 27-30.
- [13]. Dong, X., Hu, J., & Cui, Y. (2018, September). Overview of botnet detection based on machine learning. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)* (pp. 476-479). IEEE.
- [14]. Ryu, S., & Yang, B. (2018). A comparative study of machine learning algorithms and their ensembles for botnet detection. *Journal of Computer and Communications*, 6(05), 119.
- [15]. Ryu, S., & Yang, B. (2018). A comparative study of machine learning algorithms and their ensembles for botnet detection. *Journal of Computer and Communications*, 6(05), 119.
- [16]. de Neira, A. B., Araujo, A. M., & Nogueira, M. (2020, December). Early botnet detection for the internet and the internet of things by autonomous machine learning. In *2020 16th International Conference on Mobility, Sensing and Networking (MSN)* (pp. 516-523). IEEE.
- [17]. Silva, L., Utimura, L., Costa, K., Silva, M., & Prado, S. (2020). Study on machine learning techniques for botnet detection. *IEEE Latin America Transactions*, 18(05), 881-888.
- [18]. Rasheed, M. M., Faieq, A. K., & Hashim, A. A. (2020). Android Botnet Detection Using Machine Learning. *Ingénierie des systèmes d'Inf.*, 25(1), 127-130.
- [19]. Jabbar, A. F., & Mohammed, I. J. (2020, November). Development of an optimized botnet detection framework based on filters of features and machine learning classifiers using CICIDS2017 dataset. In *IOP Conference Series: Materials Science and Engineering* (Vol. 928, No. 3, p. 032027). IOP Publishing.
- [20]. Chen, R., Niu, W., Zhang, X., Zhuo, Z., & Lv, F. (2017). An effective conversation-based botnet detection method. *Mathematical Problems in Engineering*, 2017.

- [21]. Yerima, S. Y., & Alzaylaee, M. K. (2020, June). Mobile botnet detection: A deep learning approach using convolutional neural networks. In *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)* (pp. 1-8). IEEE.
- [22]. Tariq, F., & Baig, S. (2017). Machine learning based botnet detection in software defined networks. *Int. J. Secur. Appl*, 11(11), 1-12.
- [23]. Meng, X., & Spanoudakis, G. (2016). MBotCS: A mobile botnet detection system based on machine learning. In *Risks and Security of Internet and Systems: 10th International Conference, CRiSIS 2015, Mytilene, Lesbos Island, Greece, July 20-22, 2015, Revised Selected Papers 10* (pp. 274-291). Springer International Publishing.
- [24]. Nömm, S., & Bahşi, H. (2018, December). Unsupervised anomaly-based botnet detection in IoT networks. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 1048-1053). IEEE.
- [25]. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3), 12-22.