



Article

IMTIBOT: An Intelligent Mitigation Technique for IoT Botnets

Umang Garg ¹, Santosh Kumar ^{2,*} and Aniket Mahanti ^{3,*}¹ Computer Science and Engineering, Amity University, Gwalior 201301, India; umangarg@gmail.com² Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun 248002, India³ School of Computer Science, University of Auckland, Auckland 1010, New Zealand

* Correspondence: drsantosh.cse@geu.ac.in (S.K.); a.mahanti@auckland.ac.nz (A.M.)

Abstract: The tremendous growth of the Internet of Things (IoT) has gained a lot of attention in the global market. The massive deployment of IoT is also inherent in various security vulnerabilities, which become easy targets for hackers. IoT botnets are one type of critical malware that degrades the performance of the IoT network and is difficult to detect by end-users. Although there are several traditional IoT botnet mitigation techniques such as access control, data encryption, and secured device configuration, these traditional mitigation techniques are difficult to apply due to normal traffic behavior, similar packet transmission, and the repetitive nature of IoT network traffic. Motivated by botnet obfuscation, this article proposes an intelligent mitigation technique for IoT botnets, named IMTIBoT. Using this technique, we harnessed the stacking of ensemble classifiers to build an intelligent system. This stacking classifier technique was tested using an experimental testbed of IoT nodes and sensors. This system achieved an accuracy of 0.984, with low latency.

Keywords: IoT botnet; botnet mitigation techniques; stacking ensemble classifier



Citation: Garg, U.; Kumar, S.; Mahanti, A. IMTIBOT: An Intelligent Mitigation Technique for IoT Botnets. *Future Internet* **2024**, *16*, 212. <https://doi.org/10.3390/fi16060212>

Academic Editor: Iwona Grobelna

Received: 30 April 2024

Revised: 20 May 2024

Accepted: 11 June 2024

Published: 17 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Internet of Things (IoT) provides machine-to-machine or sensor-to-machine communication in the immediate vicinity. This communication is possible using a wireless sensor network, Bluetooth, Wi-Fi, or Zigbee with a unique identifier of the hardware devices. To maximize these capabilities and build significant functions, it has inherent leverage for connecting other wireless devices. There is no standard policy or set of regulations for making IoT devices. So, manufacturers are building these devices without any significant embedding of security mechanisms in order to save costs and time. As a result, unsecured smart IoT devices are vulnerable to security threats, raising the security concerns around smart applications [1]. IoT botnets are one type of critical malware that can degrade the performance of the system via distributed denial of services (DDoS) attacks, ransomware, packet losses, financial scams, etc. An IoT botnet is an infected network that consists of IoT devices infected with malicious software or commands [2].

The major objective of an IoT botnet is to cause the disruption of services and obtain a ransom for the hackers. IoT botnets are used to infect millions of IoT devices in a network. It has distinct vectors of attacks, such as mass identity fraud, information leakage, adware installation, controlling, and cryptocurrency mining. Distinct IoT botnets have been introduced in recent years, such as Bashlite, Mirai, IoT Reaper, Muhstik, Dark Nexus, and Mozi. The Mirai botnet was a milestone in the expansion of IoT botnets due to the expansion of high-profile hacks. It hacked almost 2.5 million IoT devices, launching DDoS attacks [3]. Since then, distinct variants have been launched for ransom and hack IoT applications. In the IoT botnet environment, a botmaster tries to control the network and devices. The controlling of the entire system is performed with the command and controller (C&C) server by the botmaster. It sends the commands to the IoT devices and hacks or compromises the target device [4]. To initiate the IoT botnet, there are a few steps that need to be followed:

- The botmaster initially sends a command to the C&C server for the scanning of IP addresses of IoT nodes.
- Then, it chooses the legitimate communication channel using distinct techniques such as Telnet, IRC, etc.
- The loader server is utilized to load the malware or compromise the IoT nodes.
- Once the IoT node is compromised, it joins the army of bots and tries to infect other nodes in the same network.
- These devices are controlled by the C&C server and provide the instructions for propagating the same.

To substantiate the IoT botnet, distinct algorithms and methods have been introduced by various researchers. The detection of an IoT botnet can be achieved by using host-based or network-based techniques. Due to its limited processing and power, botnet detection based on a host intrusion detection system (HIDS) achieved low reliability and accuracy [5]. The IoT domain's network-based botnet detection approach was proposed using a hierarchical classification [6]. Machine learning algorithms (MLAs) are the current trend for the identification and detection of IoT botnets that are working based on the network traffic [7]. Distinct intrusion detection systems (IDS) are utilized to detect IoT botnets with misuse, anomaly, DNS, stateful, and hybrid techniques [8]. The misuse-based method is used to detect a botnet with pre-defined signatures of the malware, while unknown malware is detectable by using an anomaly-based technique. DNS anomalies can be identified with fast flux or domain-generation algorithms. Stateful detection utilizes predetermined profiles of IoT devices and networks. If the traffic or behavior deviates from the original behavior, it will be treated as an anomaly. Hybrid IDS utilizes the power of two or more detection approaches. After characterizing, comprehending, and analyzing the behavior of the bots, honeypots are used to detect the botnet [9]. Additionally, honeypots need signature extraction, data analysis, etc., to detect the presence of bots.

To confront the IoT botnet security issues, mitigation is an important step that takes the necessary actions against the detected attack [10]. Some traditional mitigation techniques utilized for IoT botnets are as follows:

- **Secured Device Configuration:** The configuration of IoT devices should be secured via the modification of the default username and password. It also ensures that disabled unused features are regularly updated in firmware.
- **Data Encryption:** The data should be in an encrypted form during data in transit and protected from interception or unauthorized access. The involvement of protocols such as SSL/TLS is required for data encryption stored in the cloud.
- **Network Segmentation:** IoT networks are different than other networks that need to segregate IoT devices to minimize potential attacks and limit security breaches.
- **Access Controls:** A strong user-oriented access control is required that can involve biometric or multi-factor authentication (MFA) to define the restrictions of the access.
- **Vendor Security Evaluation:** IoT devices or sensors need to be bought from reputed vendors that provide security on a priority basis.
- **Physical Security:** A locked cabinet or enclosed environment is needed for the physical security of the IoT devices.
- **Real-time Monitoring:** IoT botnet mitigation requires real-time monitoring of the IoT traffic and analyzing it for any unrelated network packets.
- **Patch Management:** Regular updates to the firmware and the software including current security patches to deal with potential attacks and vulnerabilities.

These traditional mitigation techniques do not provide a suitable solution if the network system gets hacked or compromised. This requires an effective and modern technique that reacts automatically as per the requirement. Recovery from this unprecedented situation is also an important factor rather than the implementation of the usual security policies. From the above discussion, we are motivated to develop a system for the mitigation of IoT botnets. The proposed system, IMTIBoT, is a technique that can detect and mitigate IoT botnet traffic from the IoT environment. It also utilizes an ensemble learning classifier

mechanism for the classification of normal and infected traffic. The major contributions of the current article are as follows:

- A novel IMTiBoT technique is proposed for the mitigation of IoT botnets.
- An efficient algorithm is proposed for the implementation of the stacking of ensemble classifiers.
- We implement the distinguishing classifier models for classification and regression tasks to predict the performance of the models.
- We compare and evaluate the results of the classifiers in terms of the distinct parameters.

The rest of this paper is organized as follows: Section 2 examines the literature review on the topic of IoT botnet detection and mitigation. Section 3 proposes an intelligent mitigation technique consisting of three modules. It includes the description of all three modules with the configuration details and experimental setup. Section 4 presents the results and a discussion, in which the performance of the proposed model with distinguished parameters is evaluated. Finally, the conclusion and future scope are presented in Section 5.

2. Literature Review

Significant efforts have been made in the field of detection and mitigation of IoT botnets. Some researchers have tried to detect IoT botnet attacks by stacking machine learning algorithms. In this section, we cover some of the past works conducted by these researchers.

The feature selection procedure for effective botnet attack detection in a network was defined by Khaire et al. [11]. To train classifiers and apply them to the data gathered from an IoT network, these authors utilized the stable feature selection algorithms and handled the instability of the distinct sources. This technique can provide better outcomes with the selection technique, but is less stable. Christos Tzagkarakis et al. [12] proposed a technique for botnet detection that builds a model using the sparsity of the dataset and has no prior knowledge of the malicious traffic. The proposed mechanism can be applicable to IoT botnet traffic only, and it cannot be applied to other malware. Popoola et al. [13] analyzed network traffic by locating the network, utilizing network flow and classification algorithms for the filtering of normal and compromised network traffic. However, this method does not consider the stealthy propagation behavior of a botnet, which needs cooperation and information sharing to launch an attack. On the other hand, a framework for blocking malware propagation in IoT networks is described in [14] which captures the network properties. The proposed method is utilized to solve an optimized problem and prevent botnet formation with minimum overheads. This work can be improved by some strategic optimized policies for defenders and hackers.

A behavior analysis approach for botnet detection in P2P networks was presented by Beiknejad et al. [15]. This approach analyzes the nodes' behavior using flow information. This approach has a greater level of accuracy and uses flow information to identify the botnet. The proposed approach cannot be used effectively for large IoT networks. A decision tree-based approach to botnet detection using a multi-layer neural network was proposed by Gao et al. [16]. The PCA Softmax regression combination results in low computational complexity. Abbas et al. [17] reduced features and removed redundant data and irrelevant features using PCA and singular value decomposition algorithms. These algorithms contribute to minimizing memory use and execution time, providing a good detection rate but with more complexity. Zheng and Zhou [18] performed an analysis to improve the functionality of the PCA method. The experiment results demonstrate a better performance effect with the reduction in dimensions at an accuracy of 99.7689%. The classification of the traffic can be obtained from a multi-layer feed-forward neural network. However, the classifiers did not work the same for all IoT networks. Salo et al. [19] mitigated the issue of dimensionality using the t-SNE nonlinear dimensionality reduction method. The proposed method shows the effectiveness of reducing the dimensions, provided that the target dimensions are not too low, to prevent the classes from collapsing with each other. Mutlaq et al. [20] reduced the data dimensions by using a genetic algorithm. The proposed

algorithm contributes to producing a subset of relevant features. Susanto et al. [21] investigated the impact of dimensionality reduction and utilized a fast independent component analysis (ICA) for the mitigation of IoT botnet. They observed that using fast ICA and the KNN classification algorithm provides better results in terms of performance parameters.

Chaganti et al. [22] explored the existing IoT botnet mitigation mechanisms using blockchain technology. The authors implemented a state-of-the-art survey based on deployment location, victim location, and hybrid solutions. Authors did not include surveys with the integration of blockchain and IoT botnets. Djenna et al. [23] proposed a systematic mechanism for the classification of Android malware (CICAndMal2017) with dynamic deep-learning methods. They used five malware families for classification with behavior-based CNN, heuristic methods, and behavior-based DNN. The proposed approach can be applied to real-time malware with ensemble classifiers for better detection. Lawal et al. [24] proposed a framework for anomaly mitigation using fog computing. They utilized two approaches for the detection and mitigation of IoT botnet, named signature and anomaly. Dimensionality reduction is another important parameter for converting larger datasets to small and correlated data features to achieve good accuracy. This approach can perform better with additional malware signatures.

The research that has applied machine learning techniques to the issue of intrusion detection has mostly concentrated on raising accuracy scores, given less attention to enhancing system performance, and has displayed essentially no desire to deal with the interpretability and adaptability difficulties. To successfully apply these techniques in the real world, it is necessary to reevaluate the placement of sensors, the gathering of training data, the creation of classifiers, the use of classifiers for detection, and the evaluation of the results in a decision process. In the IoT context, gathering and processing only the bare minimum amount of data is necessary to achieve high accuracy scores and more easily understandable outputs.

3. Proposed Intelligent Mitigation Technique

Since an IoT botnet attack is implemented by using compromised IoT nodes, it is challenging to identify compromised nodes in the IoT environment. The proposed approach collects the data from distinguished sensor nodes, which are then filtered during preprocessing and extraneous data are removed. The presence of IoT botnet attacks can be detected using a variety of classification approaches. After the detection of the compromised node, mitigation techniques must be applied to remove or mitigate the compromised node from the IoT network [25]. Several classifiers can be utilized for the classification of malicious and regular IoT networks, including bagging, boosting, and voting [26,27]. Bagging and boosting are quite similar techniques; however, bagging considers prior bag faults. Boosting can result in overfitting, where the classifier's model performs better on the training dataset but is unable to recognize an attack on unidentified data. Voting and stacking are the two primary methods for model integration. A majority vote from the distinct classifiers predicts the class in a voting situation. However, the ensemble learning classifier is one of the most important classifiers, integrating the distinct classifiers and trains them on a random subset of data for a better classification module. The stacking of ensemble classifiers is performed with two levels of classifiers; the results of the low-level classification field unit are supplied to the second level and utilized for meta-classification to train the data. Stacking is the process of integrating distinct classifiers such as $C_1, C_2, \dots, C_n$ on a single dataset and building the model. It is a kind of mechanism where multiple models are integrated and applied on a single training dataset. The base classifiers may differ according to the hyperparameter used, algorithms, training set, and reduction in bias and variance. Furthermore, the training set is built with k-fold cross-validation and predicts the output as the outcome for M models.

In the proposed methodology, a botnet attack is carried out with the IoT-simulated environment. In the current module, two attacks are considered, namely DDoS and spam, for the evaluation of the proposed system. The proposed module is divided into three

submodules, namely the IoT botnet module for data collection, the pre-processing and feature selection module, and the stacking ensemble classifier module, which are shown in Figure 1.

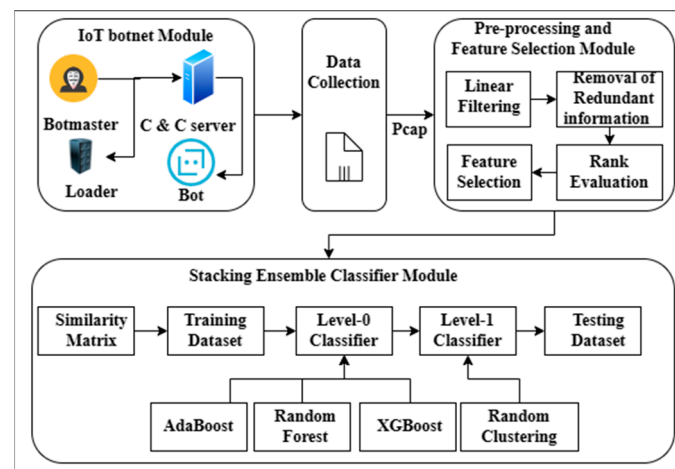


Figure 1. Proposed intelligent system.

3.1. IoT Botnet Module

The IoT botnet module is implemented or simulated by using a packet tracer (<https://www.netacad.com/courses/packet-tracer> accessed on 29 April 2024), which is an open source tool for IoT simulation. It is implemented with three IoT nodes and Raspberry Pi 3B+, which are connected to the router and access points. A brute-force attack is used on the Telnet port to deliver the attacks using Raspberry Pi 3. The C&C server was implemented with a Python script for controlling the IoT nodes. Scapy, a Python script module, was utilized to configure the setup or installments. As a result of the attack simulation, spam and DDoS attacks are generated within an IoT network. DDoS attacks are a type of cyberattack in which the perpetrator tries to render a machine or system inaccessible by unintentionally or ambiguously upsetting the Internet-related operations of a group of people. Spam attacks can be a real security issue, leading to exposure to several attacks such as worms, spyware, ransomware, and crowdsourcing. The propagation and infection are communicated through the weak default credentials and generate a payload that is installed. Once the payload has been installed on the targeted IoT device, both attacks are successfully installed on the target device.

The IoT network traffic is generated with the distinct IoT nodes implemented via a packet tracer. Figure 2 shows the experimentation setup with three IoT devices, one C&C server, a DHCP server, and a loader server. The Wireshark (<https://www.wireshark.org/> accessed on 29 April 2024) tool is utilized for the collection of data from distinct sensors and IoT nodes from IoT network traffic. The network traffic is captured and analyzed by using Wireshark and utilized for port mirroring and packet sniffing. The acquired traffic data from the experimental setup were also used to assess the performance of the suggested stacking ensemble classifiers. For the extraction of feature data, three window spans were utilized, namely as 2 ms, 4 ms, and 6 ms. The captured data were integrated with the IoT botnet dataset [28] with the selected features. This dataset consists of Telnet-based attacks that compromise the IoT devices and misuse the attack on other IoT devices. This dataset contains 733,705 rows and 19 features, indicating the large dataset available for analysis. To remove or correct unwanted data, pre-processing must be applied.

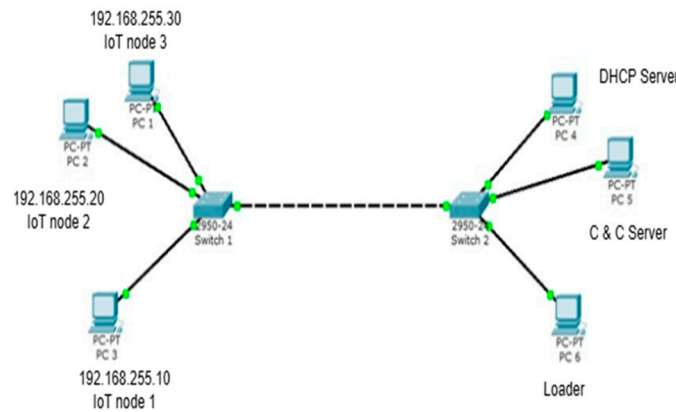


Figure 2. Experimental testbed setup for IoT botnets.

3.2. Pre-Processing and Feature Selection Module

The collected pcap files are captured through Wireshark, which consists of data packets, and converted into CSV files for data processing. The redundant or unwanted data are removed by linear filtering within a controlled range of values. The complex-valued variables are removed from the data after linear filtering. For analysis purposes, we required only network traffic information from the raw dataset. Therefore, the removal of redundant information is performed by adapting 15 attributes of the data, such as packet length, source address, destination address, packet arrival time, port number, etc. The selection of features is the most relevant step for the accurate classification of the model. We used a rank-based method for feature selection, along with score evaluation [29]. The rank of the features is measured by using Equation (1):

$$R(x) = \frac{\text{rank}(x)}{m(m-1)} \quad (1)$$

where the evaluation of parameter $R(x)$ is performed done by using the rank of each feature from m number of features. The fitness of the feature can be evaluated by using the score of the parameters using Equation (2):

$$\text{score}(x) = k * R(x), \quad x = 1, 2, \dots, m \quad (2)$$

where $R(x)$ is evaluated by using Equation (1) and k indicates the constant random selective parameter. The range of k 's value varies between 1 and 2. The method for evaluating the importance of a feature and creating a data frame for the importance is represented below.

```
# Extract feature importance's
{
    importances = model.feature_importances_feature_names = x.columns
}
# Create a DataFrame for the importances
{
    feature_importance_df = pd.DataFrame({ 'Feature': feature_names, 'Importance':
importances }).sort_values(by = 'Importance', ascending = False)
}
```

3.3. Stacking Ensemble Classifier Module

Although, classifiers can be applied as individuals for the evaluation of the performance of the model, stacking generalization and ensemble learning classifiers are used to combine multiple machine learning classifier models based on similar features and validate more skillful results. These can be applied in two distinct levels. Level 0 is the base model that fits the training data and compiles the predictions. Meanwhile, level 1 is a meta model

applied to learn how to combine the prediction of the base model. The meta-model is not trained with the base model; instead, it is used to predict the best outcomes and provide input–output pairs for best-fit models. In the proposed module, the stacking of Adaboost, Random-forest, and XGBoost is applied at the base level for training purposes, while the random cluster classifier is applied at the meta-model.

The AdaBoost technique is used to improve the performance of call trees in binary classification problems. It is utilized for classification rather than regression and works best with slow learners. It has been acknowledged that the accompanying algorithmic software XGBoost is the industry leader in producing structured or tabular data using applied machine learning. A similar version of gradient-boosted call trees called XGBoost was developed to raise speed and efficiency simultaneously. The testing of the model is evaluated by using hierarchical clustering algorithms. To create additional correctness and consistent prediction, random forest constructs distinct decision trees and integrate the results. Due to its effectiveness and accuracy, random forest is used as a base classifier and as a meta-classifier. Clustering samples are adapted in a meta-classifier for the prediction of unknown samples using Equation (3):

$$s' = \frac{1}{T} \sum_1^T s_t * P(s) \quad (3)$$

where s' refers to the predictions for unknown samples, T indicates the time taken during the training of the random samples, s_t refers to the observation's time span, and $P(s)$ indicates the Poisson distribution, which is a discrete probability distribution and is defined as in Equation (4).

$$P(s = x) = \frac{e^{-x} * x^s}{s!} \quad (4)$$

where $P(s = x)$ indicates the probability for the observation of x events, s is the Poisson random variable, x indicates the average rate, and e is the logarithmic base. The clusters can be built for a similar IoT botnet that has repetitive structures with the help of a distinct classifier. The cluster is built with different observations, $O_1, O_2, \dots, O_n$, and m distinct terms are used for the n indexed features $t_1, t_2, \dots, t_m$. Therefore, the similarity index among the two observations of the cluster is defined by Equation (5).

$$sim_{index}(s_i, s_j) = \frac{\sum_{p=1}^{p=m} (O_{ip}, O_{jp})}{\sqrt{\sum_{p=1}^{p=m} (O_{ip})^2} \sqrt{\sum_{p=1}^{p=m} (O_{jp})^2}} \quad (5)$$

where observations can be represented in terms of the vector (O_{ip}, O_{jp}) . Finally, we applied the stacking of the classifiers by combining the distinct classifiers $C_1, C_2, \dots, C_n$ and applying them on a single dataset. Base-level classifiers are integrated with the meta-level classifiers, as one of the critical factors providing this technique's advantage. The major aim of the proposed approach is to enhance the learning process [25]. It will improve by learning from the errors and alternate classifiers, while the next step is to use meta-classifiers to obtain better performance and apply the approach for testing purposes. The stacking of classifiers can be obtained using the proposed algorithm. Algorithm 1 shows the algorithmic steps for stacking the classifier.

Algorithm 1. StackingClassifier (D, E, T)

Input: Supply training data that is present in terms of

$$D = \sum_{i=1}^{i=m} \{x_i, y_i\}$$

Output: Classification of data using similarity index

Processing steps.

1. Build model using base-level classifiers.
2. Loop:
3. Till $t = 1$ to T do:
4. Learn models s based on Dataset and observations.
5. End loop;
6. Build new observations using new data values
7. Loop:
8. Till $i = 1$ to m do:
9. $Ds = \{x_i, y_i\}$ where $x_i' = \{s_1(x_i) \dots sT(x_i)\}$
10. End loop;
11. Apply meta classifier and learn to build intelligent model.
12. Return Trained model E

The above algorithm takes into account three factors, namely the dataset (D), ensemble classifier (E), and time taken by the model for training (T). The stacking of ensemble classifiers involves observations and learning from the new values of datasets. For the applicability of the algorithms and classification model, we follow the step-by-step procedure according to the proposed methodology [26].

4. Results and Discussion

This section evaluates the performance of the proposed model by using distinguished parameters such as the packet loss, throughput, packet delivery ratio, computed as the number of received/sent packets, and the packet arrival time. We observed the overfitting and underfitting of the model for evaluation.

4.1. Parameters Based on the IoT Network Traffic

There are several common parameters for classification evaluation. The very first parameter is the packet delivery ratio (PDR), which can be evaluated as the ratio of the number of delivered packets to the total number of packets transmitted from the source to the destination. Figure 3 shows the PDR for normal and botnet traffic. The results shown in the figure indicate that normal traffic has a better delivery ratio as compared to botnet traffic. Normal and botnet traffic delivers the same number of packets when the number of IoT nodes is six. Meanwhile, in other cases, normal traffic shows a higher packet delivery ratio as compared to botnet traffic. The packet delivery ratio (PDR) is defined as in Equation (6).

$$\text{PDR} = \frac{\sum \text{Total number of delivered packets}}{\sum \text{Total packet transmitted}} \quad (6)$$

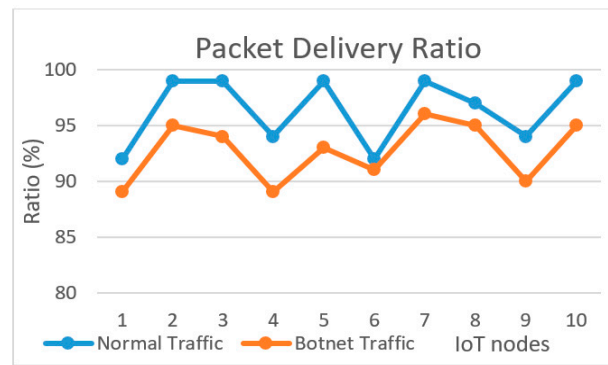


Figure 3. Packet delivery ratio for normal and botnet traffic.

4.2. Average End-to-End Delay

This parameter indicates the time taken for a packet to be transmitted from source to destination through the selected route. It can be evaluated as the mean of the end-to-end delay of all successfully delivered packets. Equation (7) defines the end-to-end delay mathematically as

$$D = \frac{1}{n} \sum_{i=1}^n (R_i - S_i) * 1000 \quad (7)$$

where D indicates the average delay time in milliseconds that can be evaluated with the difference between the reception time and sending time of a packet, and n indicates the total number of packets transmitted successfully. If the distance between the two nodes increases, it may increase the chances in packet drop. In this case, it is better to include other delays as well, such as latency, transmission delay, and propagation time. Figure 4 shows the end-to-end delay time for normal and botnet traffic. It indicates that botnet traffic takes more time in comparison to normal traffic flows.

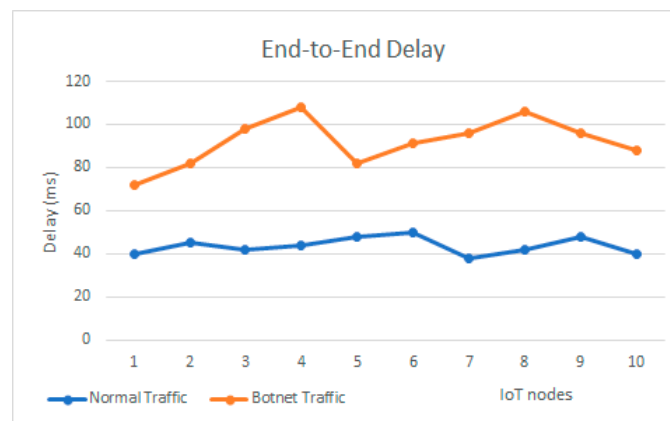


Figure 4. End-to-end delay for normal and botnet traffic.

4.3. Average Throughput

Average throughput can be defined as the ratio of total packets transmitted in the entire session and total time. Total packets can be measured as the number of packets transmitted successfully within a session. The total time is evaluated with the difference between the timestamp between the last packet and the first packet. Mathematically, it is defined as in Equation (8),

$$T = \frac{\sum \text{Total number of packets transmitted}}{\sum_{i=1}^n T_{ri} - T_{si}} \quad (8)$$

where T_{ri} and T_{si} indicate the reception and sending time of the i th packet. The throughput can be measured in kb/sec for each successfully transmitted packet. Figure 5 shows the throughput for normal and botnet traffic. The average throughput can be evaluated by using the above formula, as follows:

Average throughput for normal traffic = 0.5687

Average throughput for botnet traffic = 0.1892

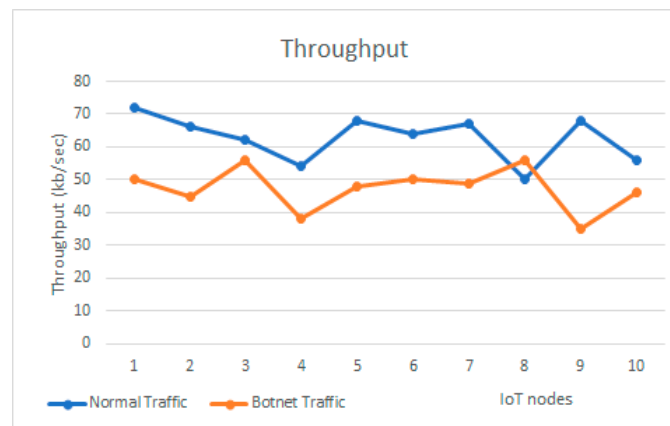


Figure 5. Throughput for normal and botnet traffic.

4.4. Packet Arrival Time

The packet arrival time is an important factor that affects the working of IoT nodes. In the absence of security techniques and easy access to the network, IoT nodes may distribute false or modified information to the other nodes. This may cause various attacks or malicious activity in the IoT traffic. The traversing of packet arrival time is a critical factor that can protect the IoT network from suspicious activities. Variations in packet arrival time may cause illegal access to nodes or latency in the traffic. Figure 6 indicates the packet arrival time for the number of packets transmitted per second. This figure indicates the superior results in the case of normal traffic.

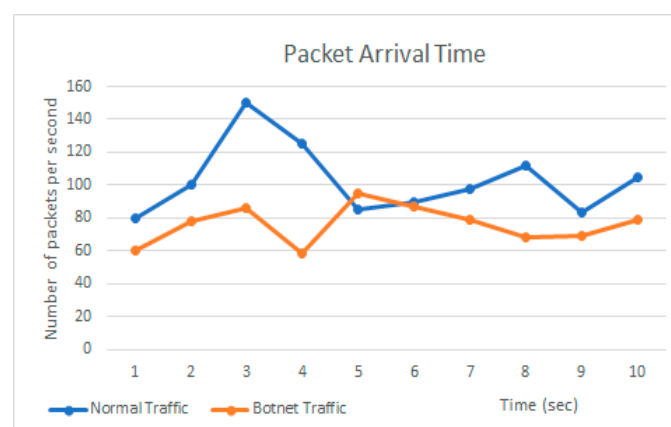


Figure 6. Packet arrival time for normal and botnet traffic.

4.5. Packet Losses

If the packet does not reach the destination successfully, it can be counted as packet loss. It can be evaluated as the difference between the total number packets received and the total number of sent packets. Figure 7 shows the packet losses as per the number of nodes in an IoT network.

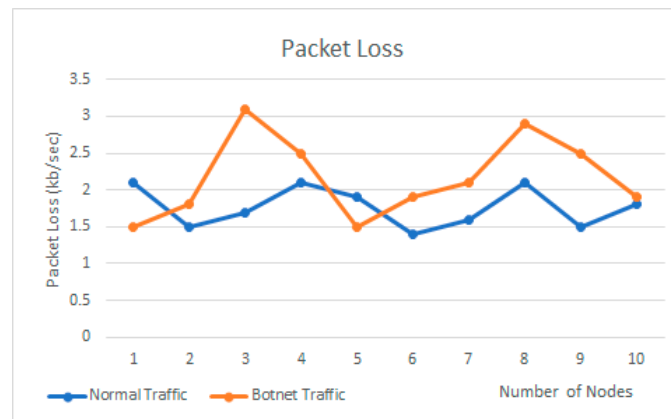


Figure 7. Packet loss for normal and botnet traffic.

4.6. Comparative Analysis

This section focuses on the evaluation of performance parameters such as accuracy, precision, recall, F1-score, and latency. All of these parameters are evaluated by using a confusion matrix that measures distinct values like true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs). These parameters can be represented mathematically as in Equations (9)–(12), respectively.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)$$

$$\text{F1-score} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \quad (11)$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TotalPredictions}} \quad (12)$$

Through the comparison of all of the parameters, the proposed work is compared some past works conducted by several researchers for the classification and mitigation of IoT botnets.

Table 1 shows the comparative analysis of the proposed model with existing models. The results show better performance with ensemble learning classifiers as compared to the other classifiers, such as KNN, CNN, RF, DT, and ANN. All performance parameters show improved results with the proposed model.

Table 1. Comparative analysis of the proposed model.

Reference	Classifier	Dataset	Accuracy	Precision	Recall	F1-Score
Nataraj et al. [30]	KNN	Anubis	0.9808	-	-	-
Su et al. [31]	CNN	IoTPOT	0.9400	-	-	-
Gibert et al. [32]	Multi-level Deep NN	BIG	0.973	0.96	0.93	0.940
Susanto et al. [21]	KNN, RF, DT	N-BaIoT	0.9995	0.9993	0.9977	0.9977
Khan et al. [29]	ANN	NSL-KDD	0.9986	0.652	1.000	0.955
Proposed Model	Ensemble Learning (Adaboost + XGBoost + Random Forest) and Random Clustering	Simulated Environment and IoT botnet dataset	0.984	0.982	0.975	0.981

5. Conclusions

IoT botnets are created by compromised nodes that generate DDoS and spam attacks. However, it is difficult to detect IoT botnets due to their non-uniform, dissimilar activities, their deletion of their history, and their invisible nature. The mitigation of IoT networks is more difficult due to normal traffic behavior, similar packet transmission, and the repetitive nature of IoT traffic. The major goal of the current article is to classify IoT network traffic in terms of normal and anomalous traffic. For this purpose, IMTIBoT is proposed for the mitigation of IoT botnets. This involves stacking of ensemble learning classifiers, which is an efficient way to classify normal and anomalous traffic in the IoT network. The performance of this system has been evaluated in terms of distinct factors such as end-to-end delay, packet delivery ratio, throughput, packet loss, and packet arrival time. Furthermore, several experiments have been performed to check the proposed model's accuracy, and the mean accuracy of the model is 0.984, with a precision value of 0.982. In the future, the proposed method will be considered for other IoT attacks, like man-in-the-middle, software exploits, and remote execution, etc. This system can be implemented in real-time scenarios as well.

Author Contributions: Software, hardware, evaluation, Writing, and data acquisition: U.G.; Original draft preparation and final version: S.K.; writing—review and editing: U.G., S.K. and A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: The original contributions presented in the study are included in the article and are available on a by-request basis.

Acknowledgments: The experimental testbed was designed by using a Raspberry Pi 3B+ module with its software tools.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kalmeshwar, M.; Prasad, N. Internet Of Things: Architecture, Issues, and Applications. *Int. J. Eng. Res. Appl.* **2017**, *7*, 85–88. [\[CrossRef\]](#)
2. De Donno, M.; Dragoni, N.; Giaretta, A.; Spognardi, A. DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation. *Secur. Commun. Netw.* **2018**, *2018*, 7178164. [\[CrossRef\]](#)
3. Providers, C.S.; Intelligence, T. Nokia Threat Intelligence Report—2019. *Netw. Secur.* **2018**, *2018*, 4. [\[CrossRef\]](#)
4. Sasi, T.; Lashkari, A.H.; Lu, R.; Xiong, P.; Iqbal, S. A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges. *J. Inf. Intell.* **2023**, *in press*. [\[CrossRef\]](#)
5. Baz, M. SEHIDS: Self Evolving Host-Based Intrusion Detection System for IoT Networks. *Sensors* **2022**, *22*, 6505. [\[CrossRef\]](#)
6. Masoudi-Sobhanzadeh, Y.; Emami-Moghaddam, S. A real-time IoT-based botnet detection method using a novel two-step feature selection technique and the support vector machine classifier. *Comput. Netw.* **2022**, *217*, 109365. [\[CrossRef\]](#)
7. NChaabouni; Mosbah, M.; Zemhari, A.; Sauvignac, C.; Faruki, P. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [\[CrossRef\]](#)
8. Zhao, H.; Shu, H.; Xing, Y. A Review on IoT Botnet. In Proceedings of the the 2nd International Conference on Computing and Data Science, Stanford, CA, USA, 28–30 January 2021; ACM: New York, NY, USA, 2021. [\[CrossRef\]](#)
9. Jain, S.; Pawar, P.M.; Muthalagu, R. Hybrid intelligent intrusion detection system for internet of things. *Telemat. Inform. Rep.* **2022**, *8*, 100030. [\[CrossRef\]](#)
10. Ali, I.; Ahmed, A.I.; Almogren, A.; Raza, M.A.; Shah, S.A.; Khan, A.; Gani, A. Systematic Literature Review on IoT-Based Botnet Attack. *IEEE Access* **2020**, *8*, 212220–212232. [\[CrossRef\]](#)
11. Khair, U.M.; Dhanalakshmi, R. Stability of feature selection algorithm: A review. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 1060–1073. [\[CrossRef\]](#)
12. Tzagkarakis, C.; Petroulakis, N.; Ioannidis, S. Botnet Attack Detection at the IoT Edge Based on Sparse Representation. In Proceedings of the 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 17–21 June 2019; IEEE: New York, NY, USA, 2019. [\[CrossRef\]](#)
13. Popoola, S.I.; Adebisi, B.; Hammoudeh, M.; Gui, G.; Gacanin, H. Hybrid Deep Learning for Botnet Attack Detection in the Internet-of-Things Networks. *IEEE Internet Things J.* **2021**, *8*, 4944–4956. [\[CrossRef\]](#)
14. Farooq, M.J.; Zhu, Q. Modeling, Analysis, and Mitigation of Dynamic Botnet Formation in Wireless IoT Networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 2412–2426. [\[CrossRef\]](#)

15. Beiknejad, H.; Vahdat-Nejad, H.; Moodi, H. P2P botnet detection based on traffic behavior analysis and classification. *Int. J. Comput. Inf. Technol.* **2018**, *6*, 2–16.
16. Gao, Q.; Wu, H.; Zhang, Y.; Tao, X. Differential game-based analysis of multi-attacker multi-defender interaction. *Sci. China Inf. Sci.* **2021**, *64*, 222302. [[CrossRef](#)]
17. Abbas, S.H. IDS feature reduction using two algorithms. *Int. J. Civ. Eng. Technol.* **2017**, *8*, 468–478.
18. Lin, Y.; Zhu, X.; Zheng, Z.; Dou, Z.; Zhou, R. The individual identification method of wireless device based on dimensionality reduction and machine learning. *J. Supercomput.* **2017**, *75*, 3010–3027. [[CrossRef](#)]
19. Salo, F.; Nassif, A.B.; Essex, A. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Comput. Netw.* **2019**, *148*, 164–175. [[CrossRef](#)]
20. Mutlaq, K.A.A.; Madhi, H.H.; Kareem, H.R. Addressing big data analytics for classification intrusion detection system. *Period. Eng. Nat. Sci.* **2020**, *8*, 693–702.
21. Susanto; Stiawan, D.; Rini, D.P.; Arifin, M.A.; Idris, M.Y.; Alsharif, N.; Budiarto, R. Dimensional Reduction with Fast ICA for IoT Botnet Detection. *J. Appl. Secur. Res.* **2022**, *18*, 665–688.
22. Chaganti, R.; Bhushan, B.; Ravi, V. A survey on Blockchain solutions in DDoS attacks mitigation: Techniques, open challenges and future directions. *Comput. Commun.* **2023**, *197*, 96–112. [[CrossRef](#)]
23. Djenna, A.; Bouridane, A.; Rubab, S.; Marou, I.M. Artificial Intelligence-Based Malware Detection, Analysis, and Mitigation. *Symmetry* **2023**, *15*, 677. [[CrossRef](#)]
24. Lawal, M.A.; Shaikh, R.A.; Hassan, S.R. An anomaly mitigation framework for iot using fog computing. *Electronics* **2020**, *9*, 1565. [[CrossRef](#)]
25. Khazane, H.; Ridouani, M.; Salahdine, F.; Kaabouch, N. A Holistic Review of Machine Learning Adversarial Attacks in IoT Networks. *Future Internet* **2024**, *16*, 32. [[CrossRef](#)]
26. Pozzebon, A. Edge and Fog Computing for the Internet of Things. *Future Internet* **2024**, *16*, 101. [[CrossRef](#)]
27. Alrubayyi, H.; Alshareef, M.S.; Nadeem, Z.; Abdelmoniem, A.M.; Jaber, M. Security Threats and Promising Solutions Arising from the Intersection of AI and IoT: A Study of IoMT and IoET Applications. *Future Internet* **2024**, *16*, 85. [[CrossRef](#)]
28. Ullah, I.; Mahmoud, Q.H. A Technique for Generating a Botnet Dataset for Anomalous Activity Detection in IoT Networks. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 134–140. [[CrossRef](#)]
29. Khan, S.; Mailewa, A.B. Discover botnets in IoT sensor networks: A lightweight deep learning framework with hybrid self-organizing maps. *Microprocess. Microsyst.* **2023**, *97*, 104753. [[CrossRef](#)]
30. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware Images, Visualization and Automatic. *ACM*. July 2011. Available online: https://vision.ece.ucsb.edu/sites/vision.ece.ucsb.edu/files/publications/nataraj_vizsec_2011_paper.pdf (accessed on 29 April 2024).
31. Su, J.; Vasconcellos, V.D.; Prasad, S.; Daniele, S.; Feng, Y.; Sakurai, K. Lightweight Classification of IoT Malware Based on Image Recognition. *Proc. Int. Comput. Softw. Appl. Conf.* **2018**, *2*, 664–669. [[CrossRef](#)]
32. Gibert, D.; Mateu, C.; Planes, J. HYDRA: A multimodal deep learning framework for malware classification. *Comput. Secur.* **2020**, *95*, 101873. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.