Article

Intelligent Mirai Malware Detection for IoT Nodes

Tarun Ganesh Palla and Shahab Tayeb *

Department of Electrical and Computer Engineering, California State University, Fresno, CA 93740, USA; tarunpalla@mail.fresnostate.edu

* Correspondence: tayeb@csufresno.edu

Abstract: The advancement in recent IoT devices has led to catastrophic attacks on the devices resulting in breaches in user privacy and exhausting resources of various organizations, so that users and organizations expend increased time and money. One such harmful malware is Mirai, which has created worldwide recognition by impacting the digital world. There are several ways to detect Mirai, but the Machine Learning approach has proved to be accurate and reliable in detecting malware. In this research, a novel-based approach of detecting Mirai using Machine Learning Algorithm is proposed and implemented in Matlab and Python. To evaluate the proposed approaches, Mirai and Benign datasets are considered and training is performed on the dataset comprised of a Training set, Cross-Validation set and Test set using Artificial Neural Network (ANN) consisting of neurons in the hidden layer, which provides consistent accuracy, precision, recall and F-1 score. In this research, an accurate number of hidden layers and neurons are chosen to avoid the problem of Overfitting. This research provides a comparative analysis between ANN and Random Forest models of the dataset formed by merging Mirai and benign datasets of the Mirai malware detection pertaining to seven IoT devices. The dataset used in this research is “N-BaIoT” dataset, which represents data in the features infected by Mirai Malware. The results are found to be accurate and reliable as the best performance was achieved with an accuracy of 92.8% and False Negative rate of 0.3% and F-1 score of 0.99. The expected outcomes of this project, include major findings towards cost-effective Learning solutions in detecting Mirai Malware strains.

Keywords: Mirai; artificial neural network; random forest; IoT

Citation: Palla, T.G.; Tayeb, S. Intelligent Mirai Malware Detection for IoT Nodes. *Electronics* **2021**, *10*, 1241. <https://doi.org/10.3390/electronics10111241>

Academic Editor: Taeshik Shon

Received: 27 April 2021

Accepted: 19 May 2021

Published: 24 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 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

Securing IoT nodes is a primary and important aspect as the evolution of security advancements has led to the development of attacks on smart devices. The widespread adoption of IoT solutions has impacted our daily lives in ways not previously possible. Such an impact has brought along various integrated cyber threats to our society. Mirai malware has been under the limelight since September 2016, when a malware research group “Malware Must Die” found a malware type which has been used to launch malicious and catastrophic DDoS attacks [1]. DDoS is a type of Cyber-attack, where the flooding caused by incoming traffic occurs from multiple sources. It is similar to the DoS attack, but the initiation of attack comes from single source in DoS. Whereas, in the DDoS attack, the attack is initiated from multiple sources (computers) [2]. Since then, Mirai has impacted the digital world by making the IoT devices vulnerable to the malware, and the infected devices consequently performed operations with an unexpected increase in the bandwidth and performed slowly. Mirai (originated from the Japanese word, meaning future) creates a malicious botnet, whereby a single internet-connected device is primarily compromised, and thereby, infects other devices in the network, which forms a large-scale network attack.

Due to the increase in the deployment of IoT devices, there is an increased risk of large-scale DDoS attacks, and with the advancement of current technology, there would be more harmful attacks in the future if the issue is not addressed fundamentally. As a result of this, several multi-national companies have been the victims of Mirai malware,

suffering financially and in productivity. This has been the primary motivation to work on Mirai and analyze its impact on the IoT environment. The Mirai has resulted in the creation of device hardening and malware prevention scripts. Due to a lack of awareness and poor maintenance of devices, which includes not updating the devices when required and not analyzing the cause and effect of malicious malwares, Mirai variations are gaining popularity and becoming too advanced to mitigate. As the analysis of the attack trend of Mirai was considered initially from June 2018, there has been a small percentage of Mirai attack in the global market, but there is a gradual increase in the Mirai attack as time has passed. By November 2018, a sharp spike in the Mirai activity was observed and the percentage of attacks rose to nearly five times when compared to previous month (October 2018). While, the percentage of Mirai activity remained steady until January 2019, in February a sharp decline in Mirai activity was observed followed by a gradual increase in the upcoming months (March 2019, April 2019, May 2019) [3]. The components of Mirai include BoT, Command and Control (C&C) Server, Loader and Report Server [4]. The servers have information about the target victim and propel the bot to launch an attack on the vulnerable device in the Network.

Machine Learning is defined as a learning based curve, which provides solutions using sequence of algorithms. After collecting the dataset, the execution is undertaken in three stages: Training the data, testing the data and cross-validating the data. The testing phase and cross-validating phase can also be included in a single step. There are two types of Machine Learning Algorithms: Supervised Learning and Unsupervised Learning. In supervised Learning, as the name suggests, we train the system regarding the execution of steps to classify the classes in training data. In other words, we have complete control of the dataset and classes involved in it. On the other hand, unsupervised Learning learns the execution steps by itself without human training. It is also referred to as “Clustering Algorithm”, where the result is obtained by clustering the data and grouping where we do not have idea about the results, unlike Supervised Learning [5]. The knowledge gap in this research involves the comparison in the performance of the model before merging the Mirai and Benign datasets and after merging the datasets.

1.1. Motivation

Mirai attack is generated when botnets become effective and not all DDoS attacks rise from botnets. However, in cases like Mirai, where the threat is caused by the vulnerability in IoT devices (the vulnerability of IoT devices includes having large number of devices with internet connection at same location, weak security practices practiced by vendor, consumers negligence of security updates) creation of botnet occurs due to the vulnerabilities, and thereby, infect the whole network of internet connected devices. To overcome Mirai threats, the manufacturers resort to pushing firmware updates to the device. Such an approach lacks scalability, in the context of IoT. Machine Learning techniques have been accurate in performing Malware based analysis. The main advantage of using Machine learning algorithms is their scalability, where the issue of detecting the malware is made easy and can be implemented for complex scenarios. The purpose of this study is to determine an efficient and scalable approach to detect Mirai malware.

1.2. Novelty

There are various ways to train the model using Neural Networks by using Python, Tensor Flow, Matlab etc. The model used in this paper is implemented in both Matlab and Python. ANN and Random Forest is used to implement the model. ANN is implemented in Matlab whereas Random Forest is implemented using Python. The overall set size of the model varies with each dataset, but the allocation of the data is same for all the datasets. The allocation of the dataset is as follows for ANN: 70% data is allocated for training set, 15% for cross validation and the remaining 15% for test set. The allocation of data for Random Forest is: 75% of data is allocated for Training set and 25% of data is allocated for Test set. The accuracy and other parameters can be determined consistently

and comparison is done between two models and final accurate model is predicted. Our research is the first of its kind to implement and train the “N-BaIoT” dataset in Matlab using ANN and observe the performance of the model by evaluating the performance metrics, where a high accuracy and F-1 score is achieved when compared to the model implemented using Python. There are other researcher’s contributions, where Machine Learning Algorithms like SVM are used for classification approach, but have the problem of high overhead. Our model uses limited resources without using external modules and achieved high accuracy and F-1 score.

The contribution of this research is as follows:

- Analyzing the dataset of Mirai and benign and training the data using Supervised Algorithm.
- Implementing the training module in Machine Learning approach using Matlab and Python.
- Assessing the framework in each stage of the training set.
- Comparing the Machine Learning approaches with different testbeds.

2. Literature Review

This section provides some of the researchers’ contributions on the detection of malware in IoT Environment using various methodologies.

3. Methodology

Machine Learning is a branch of Artificial Intelligence, which provides accurate and reliable data by studying Computer Algorithms. The algorithms used in Machine Learning divide the dataset into three categories: Training set, Cross-Validation set and Test set. Training set consists of the set of data used to train our model. During each epoch, the model is trained on the data in the training set and learns about the features in the dataset. The Cross-Validation set is used to validate the model and the Test set is used to test the model which has already been trained. You can have the dataset divided into two categories: Training set and Test set, but to avoid the problem of over fitting, Cross- Validation set is used. Overfitting of the data generally occurs, when the model has high accurate data in the Training set and less accurate data in the Test set [5].

The algorithms in Machine Learning are classified into two types: Supervised Learning and Unsupervised Learning. In this experiment, Supervised Learning algorithm is being used to train the model and evaluate the performance of it. Classification is a sub-type of Supervised Learning approach, which is used to predict discrete data in the output. For example, imagine a situation like predicting whether a person is male or female or the tumor is malicious or non-malicious. There are several types of Classification Algorithms like Support Vector Machines, Decision Trees, Random Forest and Neural Networks. In this research, Artificial Neural Networks (ANN) and Random Forest are used to analyze the behavior of the seven IoT devices by predicting the amount of malware present in each IoT device using the relevant datasets by using Feed Forward Mechanism [5].

3.1. Performance Metrics

In this research, results are implemented using Supervised Learning, where the data is known and training is provided to the dataset in timely manner and accuracy, F-1 score, precision and recall are calculated using the formula listed below.

$$\text{Precision (P)} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})}$$

$$\text{Recall (R)} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative})}$$

$$F-1 \text{ Score} = \frac{2PR}{P+R}$$

Precision is defined as the ratio of correctly predicted positive observations to the overall positive observations generated from the dataset (Of all the data generated, which fraction of data is Malicious).

Recall is defined as the ratio of correctly predicted positive observations to the overall observations in the output label (From the malicious data, what fraction did the model correctly detect as Malicious). Accuracy is defined as the ratio of correctly predicted observations to the overall observations generated from the dataset.

To understand the definitions of TP, TN, FP, and FN, let us consider the scenario used in this research:

- TP represents correctly detected as Mirai.
- FP represents incorrectly detected as Mirai, which is actually benign.
- TN represents correctly detected as benign.
- FN represents incorrectly detected as benign which is actually Mirai.

For any research to provide accurate and reliable results the TP rate should be high and FN rate should be as low as possible because the outcome of FN is catastrophic, as in the above scenario, incorrect detection of data as benign instead of Mirai will affect the chance of malware attack on devices.

3.2. Dataset

The dataset is created by distinguishing between malicious data and benign data [2]. The dataset presents the real traffic data gathered from seven IoT devices infected by Mirai malware. Each IoT device corresponds to a unique value allocated in the dataset. Each dataset has 115 features in common but the overall size of the set varies with each device. The 115 features are divided into 5 time windows with 23 features present in each time window.

The details of the IoT devices can be understood from Table 1. The training has been performed on seven IoT devices by the “N-BaIoT” dataset, which captures normal network traffic patterns. For instance, the feature MI_dir_L5_weight indicates the traffic from packet’s host MAC -IP at timeframe of five seconds. Similarly, HH_dir_L3_weight indicates the traffic going from packet’s source host to destination host at timeframe of three seconds and HpHp_dir_L3_weight indicates the traffic going from packet’s source host port to packet’s destination host port at timeframe of three seconds.

Table 1. Summary of the Literature Review.

Author	Methodology	Pros	Limitations
Sedjelmaci et al. [6]	Host based botnet detection approach	Cost effective	Computation Overhead
Meidan et al. [7]	Auto-encoders	High accuracy with more TP data	None
Kabir et al. [8]	SVM	Accuracy in detection of botnet	Time Consumption and missing information related to samples
Jan et al. [9]	SVM	Compatible with IDS	Delay in the arrival of Packets
Su et al. [10]	Droiddeep	Low false alarm rate	Susceptible to attacks due to vulnerabilities
Khoda et al. [11]	Kernel based Learning	High accuracy and Robust	None
Azmoodeh et al. [12]	IoBT	High accuracy and Precision	Cannot be used in Autonomous driving
Niu et al. [13]	Fusion Features	Compatible in Autonomous Driving	None
Meng et al. [14]	Blockchain	Scalable with less computation cost	Not compatible with IDS
Clincy et al. [15]	Blockchain	Versatility in security	Time consuming

Table 1. Cont.

Author	Methodology	Pros	Limitations
Sharmeen et al. [16]	Static Analysis	Accuracy	Not compatible to detect zero-day attacks.
Raza et al. [17]	SVELTE	Less computation cost	Cannot solve traffic issues
Thamilrasu et al. [18]	Deep Learning	Detecting zero-day attacks	None
Schmidt et al. [19]	Signature based Analysis	Low computational overhead	High FNR
Su et al. [20]	CNN	Low FNR	Vulnerable to complex code obfuscation
Venkatraman et al. [21]	Visualization Technique	Efficient detection of zero-day attacks	None
Goyal et al. [22]	SVM	Accuracy	Computation Overhead
Liu et al. [23]	Black box	High detection rate	Weak security
Kumar et al. [24]	Naïve Bayes Algorithm	Accurate and Robust	None
Naeem et al. [25]	Static Analysis	High accuracy	High performance only on Windows OS
Karanja et al. [26]	CNN	Low computation overhead	Cannot be used in Complex architecture
Nisa et al. [27]	CNN	Improvement in accuracy	None
Hemalatha et al. [28]	CNN	High accuracy	High FN rate

Feature selection is termed as selecting features related to the data and cleaning futile data commonly termed as “Feature Pre-processing” and making no rearrangement of data in the time window. Data are cleaned and the model is trained until the desired accuracy (above 50%) is achieved. Based on the received F-1 score, precision and recall, the style of training model is modified and evaluated.

“Table 2 displays the names of the IoT devices are understood and there are few multiple devices with different model numbers”. For all the devices, the data are generated, collected and transformed into a dataset with multiple features.

Table 2. Names of the IoT devices and their corresponding Unique Numbers.

S.No	Device Name	Total Set Size	Size of Training Set	Size of Cross-Validation Set	Size of Test Set
1	Doorbell	49,548	34,684	7432	7432
2	Ennio Doorbell	39,400	27,370	6015	6015
3	Philips Monitor	175,240	122,668	26,286	26,286
4	PT_737E Security Camera	62,154	43,508	9323	9323
5	PT_838_ Security Camera	57,997	40,597	8700	8700
6	Samsung Webcam	46,585	32,609	6988	6988
7	XCS_1003 Security Camera	19,528	13,670	2929	2929

The dataset of first IoT device (1) from Table 2 has the set size of 49,548, of which 34,684 is allocated to training set, 7432 each to cross-validation set and test set. The dataset of second IoT device (2) from Table 2 has the set size of 39,400, where the size of the training set is 27,370 and the size of cross-validation set and test set is 6015 each. The dataset of third IoT device (3) from Table 2 has the set size of 175,240, where the size of the training set is 122,668 and the size of cross-validation set and test set is 26,286 each. The dataset of fourth IoT device (4) from Table 2 has the set size of 62,154, where the size of the training set is 43,508 and the size of cross-validation set and test set is 9323 each. The dataset of fifth IoT device (5) from Table 2 has the set size of 57,997, where the size of the training

set is 40,597 and the size of cross-validation set and test set is 8700 each. The dataset of sixth IoT device (6) from Table 2 has the set size of 46,585, where the size of the training set is 32,609 and the size of cross-validation set and test set is 6988 each. The dataset of seventh IoT device (7) from Table 2 has the set size of 19,528, where the size of the training set is 13,670 and the size of cross-validation set and test set is 2929 each.

3.3. Methodology for ANN

The primary methodology involved in this model is the use of Artificial Neural Networks and Random Forest to compute the accuracy level of the data. ANN is used to resolve problems related to tabular data and text data. Since the dataset used in this research consists of features, which indicates the traffic moving from packets source host to destination host, ANN is used which serves best performance compared to CNN and RNN [29]. Moreover, ANN has the ability to learn non-linear function by using activation function, where complex relationship between input and output of the neural network model is easily learnt. The neuron is a computational unit, which obtains the number of inputs through input wires, performs computation and sends the output via its axon to other nodes or neurons in the brain. In this model, the neuron consists of hidden layers because hidden layers prevent the formation of non-linearity and over-fitting and to achieve computational efficiency, specific neurons are added to hidden layer [30].

The block diagram of the overall methodology can be understood from Figure 1. Initially, when Mirai and benign datasets are independently analyzed, the accuracy level (35%) and reliability level (0% data in both TP and FN) is extremely low and the data are inconsistent in training the model. However, by merging both the datasets and cleansing the data, i.e., by removing the futile data, the accuracy can then be checked by adding neurons in the hidden layer of the neural network model.

The Mirai and benign datasets have different set sizes for each IoT device. The merging of datasets should be carefully checked as the dataset of Mirai and benign pertaining to the same device can be merged. For example, the Mirai dataset of the first device cannot be merged with benign dataset of second device. The first step in the process is to clean the dataset; this is done by identifying futile data. There are some sections in the dataset where there are no data available for that particular row or column, that section of data needs to be identified and removed. The main important point to be observed is to verify the time window before cleansing the data as the data from one time window cannot be merged with the data from a different time window in the same dataset. For example, the total dataset consists of five time windows. It does not mean that all the five time windows are to be considered. Based on the merging of Mirai and benign dataset, if three time windows are considered and the order of dataset is not changed, the data from the third time window cannot be merged into the data in the first time window.

The next step is to start the training model; this is done by importing the dataset into Matlab. The allocation of data pertaining to Training set, Cross-Validation set and Test set is mentioned in the program with epoch set as 1000 as the training data size is large. Epoch is defined as the number of cycles taken to complete the training dataset in the Neural Network Architecture. Upon running the program, the simulation time is based on the size of the dataset and Confusion Matrix is retrieved. This is how the model is started and functioning is carried. Next step would be to analyze the TP, TN, FP and FN parameters from Confusion Matrix. Figure 1 displays the Flow chart of the model. The efficiency and reliability of the model is based on the hidden neurons. If the number of hidden layer neurons is not chosen appropriately, the model overfits and affects the accuracy thus making the system inefficient. The number of hidden layer neurons should be less than twice the size of input layer and is normally, either one third or two-thirds of the size of input layer [31]. Considering the above conditions, based on the trial and error method, the number of hidden layer neurons is chosen. Optimum selection of the number of hidden layer neurons paves the way for the model to not over-fit. If there is any discrepancy in the parameters, which includes negligible data making the model unreliable due to the fact

that hidden layer neurons is selected, the number of hidden layer neurons is changed by adding or deleting the neurons until accurate model is achieved without overfitting.

Figure 1. Flow Chart of the model.

In this research, the data are pre-processed, where the features in the time windows possess the packets, which reach the destination port in different times commonly known as “Jitter”, which consists of futile data. Since there is a delay associated with packets, the data are not properly recorded and becomes futile when training, so these features are discarded. We are also going to normalize the features using the min-max normalization to make sure that all the inputs have the same weight to accelerate the training model. The next step would be to check the accuracy and reliability of the model. Even if the accuracy

of the model is good, it may not be reliable (negligible data present in the parameters or high amount of FN data than TP data). These factors need to be properly analyzed and if the accuracy of the model is less than the model before merging the datasets with poor reliability, the model can be discarded and again dataset cleaning should be performed.

The Confusion Matrix is an error matrix, which evaluates the performance of Supervised Learning Algorithm in the field of Machine Learning. In general, the Confusion Matrix has two classes—Predicted class and benign class. Based on the dataset, the row of the matrix can be predicted or actual class. If the row of the matrix is predicted class then column of the matrix becomes actual class and vice versa [32]. In our case, the row of the matrix is a predicted class and column of the matrix is actual class, as the first class in the dataset is benign. In the Confusion Matrix generated in this model, the row represents the actual class and the column represents benign class. Using these classes, the TP, TN, FP and FN data is generated and accuracy of the Model is determined. As we have seven different datasets, each for benign and Mirai, the neural network model is trained for each dataset and the performance is analyzed, in terms of accuracy and reliability.

3.4. Methodology for Random Forest

Random Forest is a Learning method that carries its operation by using multiple decision trees. The final result depends on the result of majority output of trees [33]. In this research, two label names are considered Mirai and benign, where the value of zero is assigned to benign data and the value of one is assigned to Mirai data. Upon training the data, the model predicts its accuracy to detect the Mirai malware present in the dataset.

Random Forest comes under the category of Supervised Learning Algorithm. The novelty of Random Forest is that it can be applied to both Classification and Regression approach [30]. In this research, Random Forest is used for Target Prediction, i.e., the amount of accuracy of Mirai malware present in the dataset similar to ANN and additionally, the precision, recall and F-1 score are also predicted and compared with the ANN model. The dataset of Mirai and benign, where the merging of dataset and pre-processing of dataset is same for Random Forest similar to Figure 1, but there is no usage of Neurons in Random Forest. The first step is to start training the model in Python is to import the pre-processed dataset into Python and initialize the dataset by allocating a label name to both Mirai and Benign pertaining to individual data. The next step is to allocate the percentage of data pertaining to the training set and test set. In this research, 75% of data are allocated to the training set and 25% of data are allocated to the test set. The input and output training data are initialized and the metrics like Accuracy, Precision, Recall and F-1 score are calculated using the predicted output model.

3.5. Experimental Setup

The testbed used in this experimentation is Matlab 2018b and Python. Matlab is a proprietary programming language tool used in solving complex problems by providing simple and efficient solutions. The main advantage of using Matlab for ANN is it provides user interface with programs written in other languages [34]. In this research, ANN algorithm is implemented in Matlab to provide accuracy and reliability in classification of data. Additionally, Microsoft Excel is also used for importing dataset into Matlab and also for modification of data and for initializing the features of the data. For Random Forest, Python is used to validate the model and determine the percentage of accuracy of IoT devices in detecting the Mirai malware.

4. Results and Discussion

Matlab provides simple and efficient mode of accurate data in handling ANN based Machine Learning Algorithm. The simulation time depends on the size of the dataset being trained and varies for each IoT device. The dataset corresponding to the first IoT device has been considered and training is performed on that model with 70% of the data allocated to training set, 15% of the data to Cross- Validation set and the rest 15% of the data to Test set.

ANN is generally composed of layers: Input, hidden and output layer. Based on the features in the dataset, the number of neurons in the input layer can be identified and depending on the output class, the output layer can also be summarized. The main challenge to perform is to identify the number of hidden layers and neurons present in it [30].

In this research, ANN is built using multi-layer perceptron with two hidden layers. The most important thing to perform neural network architecture is to decide the number of hidden layers and neurons present in the hidden layers. Implementing the model with an increased number of hidden layers can cause the model to overfit, and similarly, with very few neurons in the hidden layer, can cause the problem of underfitting, which destroys the accuracy of the Machine Learning model. Implementing the model with a high number of neurons can cause the problem of over fitting and increases the complexity of the model. Therefore, determining the number of neurons is a critical factor in completing the architecture of the model. In our model, initially the hidden layers are tested with different number of neurons and observed negligible data generated with parameters like TP, FN and accuracy. The number of hidden layer neurons should be less than twice the size of input layer and is normally, either one third or two-thirds of the size of input layer [31]. Considering these factors, the hidden layer neurons are chosen as 22 and 10, thus, the performance of the architecture increased with non-negligible data generated in the parameters.

“From Figure 2, it can be observed that the input parameters with weight (w) of neurons are combined with bias (b) and passed on to the activation layer termed as “Tangent Hyperbolic Function” or Tanh function”. Tanh function is a non-linear and extended form of sigmoid function, which is used in the case of binary classification. The main advantage of activation function is that it can perform complex tasks easier. If there is no activation function, the model learns only linear functions and cannot ever perform complex tasks. Bias refers to the improper conclusions generated from the data. The function of activation layer is that it decides the neurons to be forwarded first to the next layer from available neurons to avoid overlapping.

Figure 2. Block Diagram of the Neural Network Architecture.

“When individual datasets are analyzed, accuracy level is pretty low and incomparable with other dataset, which can be observed in Figure 3”. The confusion matrix with all the TP, TN, FP and FN cases can also be observed. The overall size of the set is 49,548 and training is performed with 67 features. The size of the training set is 34,684, size of the Cross-Validation set is 7432 and size of test set is 7432. From the overall confusion matrix, we can analyze the accuracy being 34.6% and the Precision being 0.00015, Recall being 0.45 and F-1 score being 0.00012. This training model is disregarded as the F-1 score is negligible. The TP, TN data is also negligible with 0% data generated in the matrix. Although there needs to be less number of FN data, but in Figure 3 we do find 0% FN data, which indicates the training has not been performed well and in the validation matrix section, Not a Number (NaN) data are found, which clearly indicates that there is an error while validating the data. The dataset which has been trained is a pure Mirai dataset and it can be observed from Figure 3 that there is missing data in sections. This is due to the non-comparable level of the datasets, i.e., when a particular dataset is trained, one cannot identify the issues present in that dataset. Only when two datasets are merged, one can

compare the data present in the datasets. The required parameters can be obtained from the All Confusion Matrix in Figure 3.

Figure 3. Confusion Matrix of first IoT device before merging the datasets: training (top left), validation (top right), test (bottom left), all (bottom right).

Merging of datasets is performed by cleaning the futile data present in the dataset and making no rearrangement of data in the time window. “Figure 4 displays the Confusion Matrix of the first IoT device after merging Mirai and benign datasets and illustrates the accuracy level of the model in detecting the Mirai Malware”. This dataset has the set size of 49,548, of which 34,684 is allocated to training set, 7432 each to cross-validation set and test set similar to Figure 3, but the difference is merging of datasets of Mirai and benign, which can be observed from Figure 4.

In the second IoT device, similar to Figure 4, the datasets of Mirai and benign are merged by cleaning the futile data. “Figure 5 displays the Confusion Matrix of the second IoT device and demonstrates the accuracy level of the model in detecting Mirai malware”. From Figure 5, it can be observed that the accuracy level of the model is higher than Figure 4 and has high TP data and less FN data with high F-1 score. The accuracy is found out to be 83.7% with precision being 0.99, recall being 0.96 and F-1 score being 0.99.

After merging the datasets into a single dataset, the dataset is also implemented using Random Forest in Python and this model generated accuracy of 55%, Precision being 0.63, Recall being 0.52 and F-1 score being 0.57. Since the F-1 score of Random Forest is less than ANN, preference of accurate model is provided to ANN over Random Forest. The name of second IoT device is Ennio Doorbell, so from the data generated in Figure 5, it can be observed that the ANN model has high reliability to detect Mirai malware on Ennio Doorbell with less FN data.

In the third IoT device, the datasets of Mirai and benign are merged by cleaning the futile data. “Figure 6 displays the Confusion Matrix of the third IoT device and demonstrates the accuracy level of the model in detecting Mirai malware.” From Figure 6, it can be observed that the accuracy level of the model is higher than Figure 2 and in Figure 6 there is less number of TP and FN data. The accuracy level of overall Confusion Matrix is found out to be 65.4%; Precision is calculated and found out as 0.2, Recall as 0.36 and F-1 score as 0.27. Though the accuracy is higher than Figure 2, F-1 score is lesser than Figure 5. The name of third IoT device is Philips Monitor, so from the data generated

from Figure 6, it can be observed that the model is accurate in detecting Mirai malware on Philips Monitor with minimum FN data. After merging the datasets into a single dataset, the dataset is also implemented using Random Forest in Python and this model generated accuracy of 54.2%, Precision being 0.28, Recall being 0.25 and F-1 score being 0.26. Though the F-1 scores of both the models are almost same (F-1 score of Random Forest is slightly less than ANN), preference of accurate model is provided to ANN over Random Forest.

Figure 4. Confusion Matrix of first IoT device after merging the datasets: training (top left), validation (top right), test (bottom left), all (bottom right).

Figure 5. Confusion Matrix of second IoT device: training (top left), validation (top right), test (bottom left), all (bottom right).

Figure 6. Confusion Matrix of third IoT device: training (top left), validation (top right), test (bottom left), all (bottom right).

In the fourth IoT device, the datasets of Mirai and benign are merged by cleaning the futile data. “Figure 7 displays the Confusion Matrix of the fourth IoT device and demonstrates the accuracy level of the model in detecting Mirai malware”. From Figure 7, it can be observed that the accuracy level of the model is higher than Figure 5 and in Figure 6, there are fewer FN data and a high number of TP data making it accurate and reliable in detecting the malware.

Figure 7. Confusion Matrix of fourth IoT device: training (top left), validation (top right), test (bottom left), all (bottom right).

The accuracy level of overall Confusion Matrix is found out to be 92.8%, Precision is calculated and found out as 1, Recall as 0.99 and F-1 score as 0.99. This model has excelled in all the parameters and is highly accurate and reliable in detecting the malware and thus

it provided an accuracy of 92.8% in detecting the malware. After merging the datasets into a single dataset, the dataset is also implemented using Random Forest in Python and this model generated accuracy of 72.3%, Precision being 0.62, Recall being 0.56 and F-1 score being 0.58. Since the difference between F-1 score of Random Forest and ANN is huge and F-1 score of ANN is more, preference of accurate model is provided to ANN over Random Forest. The name of fourth IoT device is PT_73 E Security Camera, so from the data generated from Figure 7, it can be observed that the ANN model is accurate and reliable in detecting Mirai malware on Security Camera with minimum FN data and maximum TP data.

In the fifth IoT device, the datasets of Mirai and benign are merged by cleaning the futile data. From Figure 8, it can be observed that the accuracy level of the model is higher than Figure 6 and there is less number of FN data and a high number of TN data, making it accurate and reliable in detecting malware. The accuracy level of the overall Confusion Matrix is found out to be 60.5%, Precision is calculated and found out as 0.17, Recall as 0.81 and F-1 score as 0.30. The accuracy level is partially lower than Figure 6, but the F-1 score is higher than Figure 6. Therefore, this model is equally accurate and reliable in detecting the malware in IoT environment. After merging the datasets into a single dataset, the dataset is also implemented using Random Forest in Python and this model generated accuracy of 57.8%, Precision being 0.32, Recall being 0.26 and F-1 score being 0.28. Though the accuracies of both the models look similar, the F-1 score of Random Forest is slightly lesser than ANN. So the preference of accurate model is provided to ANN over Random Forest. The name of the fifth IoT device is PT_838_ Security Camera, so from the data generated from Figure 8, it can be observed that the ANN model is accurate in detecting Mirai malware on Security Camera with minimum FN data and maximum TN data.

Figure 8. Confusion Matrix of fifth IoT device: training (top left), validation (top right), test (bottom left), all (bottom right).

In the sixth IoT device, the datasets of Mirai and benign are merged by cleaning the futile data. Figure 9 displays the Confusion Matrix of the sixth IoT device and demonstrates the accuracy level of the model in detecting Mirai malware. From Figure 9, it can be observed that the accuracy level of the model is similar to Figure 8 and there are fewer FN data and a higher number of TP data, making it accurate and reliable in detecting the malware. The accuracy level of overall Confusion Matrix is found out to be 78.3%, Precision is calculated and found out as 0.69, Recall as 0.99 and F-1 score as 0.81. This model has excelled in all the parameters similar to Figures 7 and 9 is highly accurate and reliable

in detecting the malware. After merging the datasets into a single dataset, the dataset is also implemented using Random Forest in Python and this model generated accuracy of 70.2%, Precision being 0.56, Recall being 0.86 and F-1 score being 0.67. Since the F-1 score of Random Forest is less than ANN, preference of accurate model is provided to ANN over Random Forest. “The name of sixth IoT device is Samsung Webcam, so from the data generated from Figure 9”, it can be observed that the model is accurate in detecting Mirai malware on Samsung Webcam with minimum FN data and maximum TP data.

Figure 9. Confusion Matrix of sixth IoT device: training (top left), validation (top right), test (bottom left), all (bottom right).

In the seventh IoT device, the datasets of Mirai and benign are merged by removing the futile data. “Figure 10 displays the Confusion Matrix of the seventh IoT device and demonstrates the accuracy level of the model in detecting Mirai malware”. From Figure 10, it can be observed that the accuracy level of the model is slightly higher than Figure 9 and there is less number of FN data and high number of TP data making it accurate and reliable to detect the malware. The accuracy level of overall Confusion Matrix is found out to be 83.9%, Precision is calculated and found out as 0.84, Recall as 0.99 and F-1 score as 0.92. This model has excelled in all the parameters similar to Figure 10 and is highly accurate and reliable in detecting the malware. After merging the datasets into a single dataset, the dataset is also implemented using Random Forest in Python and this model generated accuracy of 75.6%, Precision being 0.68, Recall being 0.92 and F-1 score being 0.78. Since the F-1 score of Random Forest is less than ANN, preference of accurate model is provided to ANN over Random Forest. The name of seventh IoT device is XCS_1003 Security Camera, so from the data generated from Figure 10, it can be observed that the model is accurate in detecting Mirai malware on XCS_1003 Security Camera with minimum FN data and maximum TP data.

Figure 10. Confusion Matrix of seventh IoT device: training (top left), validation (top right), test (bottom left), all (bottom right).

The results demonstrate that the data pertaining to the second, fourth, sixth and seventh devices are highly accurate for ANN with accuracies being 83.7%, 92.8%, 78.3%, and 83.9%, respectively in detecting the Mirai malware with high number of TP (83.6%, 86.6%, 47.8% and 83.8%) and high F-1 scores (0.97, 0.99, 0.81 and 0.92). The accuracy and F-1 score of the ANN model pertaining to all the seven IoT devices is higher compared to the Random Forest model observed in Table 3. Therefore, ANN is the best algorithm for detecting Mirai malware in IoT environment. “Table 3 gives the overall comparison of ANN and Random Forest with respect to various metrics observed in this research.”

Table 3. Comparison of the metrics between ANN and Random Forest.

Device Name	ANN				Random Forest			
	Accuracy	Precision	Recall	F-1 Score	Accuracy	Precision	Recall	F-1 Score
Door Bell	67.5%	0.97	0.67	0.79	43.1%	0.5	0.43	0.46
Ennio Doorbell	83.7%	0.99	0.96	0.97	55%	0.63	0.53	0.57
Philips Monitor	65.4%	0.2	0.36	0.27	54.2%	0.28	0.25	0.26
PT_737E_security camera	92.8%	1	0.99	0.99	72.3%	0.62	0.56	0.58
PT_838_security camera	60.5%	0.17	0.81	0.30	57.8%	0.32	0.26	0.28
Samsung Webcam	78.3%	0.69	0.99	0.81	70.2%	0.56	0.86	0.67
XLS_1003 Security Camera	83.9%	0.84	0.99	0.92	75.6%	0.68	0.92	0.78

Figure 11 displays the Receiver Operating Characteristic (ROC) curve for the fourth IoT device, which achieves the best performance of all IoT, in terms of accuracy, precision, recall and F-2 score. Here, Class 1 is Mirai and Class 2 is benign and performance is evaluated between TPR and FPR. The curve displays the Mirai data has higher threshold than benign data and from the Confusion Matrix, where TPR is found out as 0.97 and FPR as 0.04, the Area Under Curve (AUC) is calculated in Matlab by using the “Trapz” function and is found out to be 0.92, which is considered as a good performance model.

Additionally, a statistical analysis of our results is performed by reporting 95% confidence range, which predicts how well the model might make a mistake when implemented. Initially, classification error is calculated by evaluating the incorrect predictions to the total predictions. For the fourth IoT device, error is found out to be 0.071. The constant for 95% confidence range is 1.96 and confidence interval is calculated, which is either 6.3% or 7.6% chance that the model might make wrong predictions during implementation, which is beneficial for the implementation of the model as the percentage of the model making wrong predictions is low.

Figure 11. ROC Curve of the best accurate model (Fourth IoT Device): training ROC (top left), validation ROC (top right), test ROC (bottom left), all ROC (bottom right).

From Table 3, it is understood that ANN model gives the accurate data and has high F-1 score in detecting Mirai malware. A similar method of detecting Mirai Malware was proposed by Kotenko et al. [35] using Hadoop Framework and provided accurate data for seven IoT devices similar to our model. “Table 4 provides the comparison between the accuracy of our model and Kotenko et al. [35], in terms of ANN.”

From Table 4, it can be observed that there is similarity in the accuracy of the two models. Our ANN model has significantly high levels of accuracy in four devices, whereas the ANN model, proposed by Kotenko et al. [35] has a high level of accuracy in three devices. Our model is effective in detecting Mirai malware in four devices, primarily in the security cameras, Webcam and Ennio Doorbell, which are instrumental in becoming susceptible to Mirai Malware.

Table 4. Comparison between accuracy of ANN in different models.

Device Name	Accuracy of Our Model	Accuracy of the Model Proposed by Kotenko et al. [35]
Door Bell	67.5%	90.8%
Ennio Doorbell	83.7%	71.3%
Philips Monitor	65.4%	91.1%
PT_737 E _security camera	92.8%	86.6%
PT_838_security camera	60.5%	88.7%
Samsung Webcam	78.3%	73.6%
XLS_1003 Security Camera	83.9%	80.3%

5. Conclusions

The advancement of IoT devices is shaping the world in complete digitalized growth in the economy of the world by creating new and innovative techniques, like smart home, smart grid, smart healthcare etc. In future, almost every sector in the world will turn smart in one way or the other. One needs to be aware of the security issues while designing smart devices, malwares can cause huge problems to the industry. The approach proposed in this experiment provides better accuracy with 92.8% and less amount of FN rate with 0.3% and AUC being 0.92 and achieved good F-1 score being 0.99, when compared to the case when merging of the datasets not done. Additionally, the comparison between models is an accurate model, which provides efficient accuracy and reliability for detecting the malware. The classification algorithm, implemented in this scenario, accurately distinguishes between malicious and benign classes by generating large TP data and high F-1 score for the algorithm to function. However, in two of the IoT devices our research observed a slight higher value of FN data, although lesser than TP data due to which the accuracy of the device is reduced. The overall assessment showed that merging of datasets is very important to evaluate the performance of the dataset in any method of implementation and futile data should be discarded commonly termed as “Data Preprocessing” before working on any model and testing it. As a future work, we would like to supplement the data in the dataset to an existing model and observe the performance of the model and then compare the existing model with our model.

Author Contributions: Conceptualization, S.T. and T.G.P.; methodology, T.G.P.; software, T.G.P.; validation, T.G.P.; formal analysis, S.T.; investigation, S.T.; data curation, T.G.P.; writing—original draft preparation, T.G.P.; writing—review and editing, S.T.; visualization, T.G.P.; supervision, S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by a grant from Fresno State Transportation Institute.

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

References

1. Kambourakis, G.; Kolias, C.; Stavrou, A. The mirai botnet and the iot zombie armies. In Proceedings of the MILCOM 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 23–25 October 2017; pp. 267–272.
2. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Zhou, Y. Understanding the mirai botnet. In Proceedings of the 26th {USENIX} Security Symposium ({USENIX} Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 1093–1110.
3. Lim, M. I Can’t Believe Mirais: Tracking the Infamous IoT Malware—OnWire—Identity and Access Management Services and Cloud Solutions. Medium. Available online: <https://onwireco.com/2019/07/18/i-cant-believe-mirais-tracking-the-infamous-iot-malware/> (accessed on 20 April 2021).
4. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
5. Sidana, M. Intro to Types of Classification Algorithms in Machine Learning. Medium. 16 May 2020. Available online: <https://medium.com/sifium/machine-learning-types-of-classification-9497bd4f2e14> (accessed on 20 April 2021).
6. Sedjelmaci, H.; Senouci, S.M.; Al-Bahri, M. A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6. [CrossRef]

7. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
8. Kabir, E.; Hu, J.; Wang, H.; Zhuo, G. A Novel Statistical Technique for Intrusion Detection Systems. *Future Gener. Comput. Syst.* **2018**, *79*, 303–318. [CrossRef]
9. Jan, S.U.; Ahmed, S.; Shakhov, V.; Koo, I. Toward a Lightweight Intrusion Detection System for the Internet of Things. *IEEE Access* **2019**, *7*, 42450–42471. [CrossRef]
10. Su, X.; Zhang, D.; Li, W.; Zhao, K. A Deep Learning Approach to Android Malware Feature Learning and Detection. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 244–251. [CrossRef]
11. Khoda, M.E.; Imam, T.; Kamruzzaman, J.; Gondal, I.; Rahman, A. Robust Malware Defense in Industrial IoT Applications using Machine Learning with Selective Adversarial Samples. *IEEE Trans. Ind. Appl.* **2020**, *56*, 4415–4424. [CrossRef]
12. Azmoodeh, A.; Dehghantanha, A.; Choo, K.-K.R. Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning. *IEEE Trans. Sustain. Comput.* **2018**, *4*, 88–95. [CrossRef]
13. Niu, W.; Zhang, X.; Du, X.; Hu, T.; Xie, X.; Guizani, N. Detecting Malware on X86-Based IoT Devices in Autonomous Driving. *IEEE Wirel. Commun.* **2019**, *26*, 80–87. [CrossRef]
14. Meng, W.; Tischhauser, E.W.; Wang, Q.; Wang, Y.; Han, J. When Intrusion Detection Meets Blockchain Technology: A Review. *IEEE Access* **2018**, *6*, 10179–10188. [CrossRef]
15. Clincy, V.; Shahriar, H. IoT Malware Analysis. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019; pp. 920–921. [CrossRef]
16. Sharmeen, S.; Huda, S.; Abawajy, J.H.; Ismail, W.N.; Hassan, M.M. Malware Threats and Detection for Industrial Mobile-IoT Networks. *IEEE Access* **2018**, *6*, 15941–15957. [CrossRef]
17. Raza, S.; Wallgren, L.; Voigt, T. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Netw.* **2013**, *11*, 2661–2674. [CrossRef]
18. Thamilarasu, G.; Chawla, S. Towards Deep-Learning-Driven Intrusion Detection for the Internet of Things. *Sensors* **2019**, *19*, 1977. [CrossRef] [PubMed]
19. Schmidt, A.-D.; Bye, R.; Schmidt, H.-G.; Clausen, J.; Kiraz, O.; Yuksel, K.A.; Camtepe, S.A.; Albayrak, S. Static Analysis of Executables for Collaborative Malware Detection on Android. In Proceedings of the 2009 IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–5. [CrossRef]
20. Su, J.; Vasconcellos, V.D.; Prasad, S.; Daniele, S.; Feng, Y.; Sakurai, K. Lightweight Classification of IoT Malware Based on Image Recognition. In Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; pp. 664–669. [CrossRef]
21. Venkatraman, S.; Alazab, M. Use of Data Visualisation for Zero-Day Malware Detection. *Secur. Commun. Netw.* **2018**, *2018*, 1–13. Available online: <https://link.gale.com/apps/doc/A596644791/AONE?u=csufresno&sid=AONE&xid=e5d8cbb2> (accessed on 28 November 2020). [CrossRef]
22. Goyal, M.; Sahoo, I.; Geethakumari, G. HTTP Botnet Detection in IOT Devices using Network Traffic Analysis. In Proceedings of the 2019 International Conference on Recent Advances in Energy-Efficient Computing and Communication (ICRAECC), Nagercoil, India, 7–8 March 2019; pp. 1–6. [CrossRef]
23. Liu, X.; Du, X.; Zhang, X.; Zhu, Q.; Wang, H.; Guizani, M. Adversarial Samples on Android Malware Detection Systems for IoT Systems. *Sensors* **2019**, *19*, 974. [CrossRef] [PubMed]
24. Kumar, R.; Zhang, X.; Wang, W.Y.; Khan, R.U.; Kumar, J.; Sharif, A. A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features. *IEEE Access* **2019**, *7*, 64411–64430. [CrossRef]
25. Naeem, H.; Guo, B.; Naeem, M.R. A light-weight malware static visual analysis for IoT infrastructure. In Proceedings of the 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 26–28 May 2018; pp. 240–244. [CrossRef]
26. Karanja, E.M.; Masupe, S.; Jeffrey, M.G. Analysis of internet of things malware using image texture features and machine learning techniques. *Internet Things* **2020**, *9*, 100153. [CrossRef]
27. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features. *Appl. Sci.* **2020**, *10*, 4966. [CrossRef]
28. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Kadry, S.; Damaševičius, R. An Efficient DenseNet-Based Deep Learning Model for Malware Detection. *Entropy* **2021**, *23*, 344. Available online: <https://www.mdpi.com/1099-4300/23/3/344/html> (accessed on 15 May 2021). [CrossRef] [PubMed]
29. Davis, A.; Gill, S.; Wong, R.; Tayeb, S. Feature Selection for Deep Neural Networks in Cyber Security Applications. In Proceedings of the 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Vancouver, BC, Canada, 21–24 April 2020; pp. 1–7. [CrossRef]
30. Tayeb, S.; Raste, N.; Pirouz, M.; Latifi, S. A cognitive framework to secure smart cities. In Proceedings of the MATEC Web of Conferences, Las Vegas, NV, USA, 26 September 2018; Volume 208, p. 1.
31. De Villiers, J.; Barnard, E. Backpropagation neural nets with one and two hidden layers. *IEEE Trans. Neural Netw.* **1993**, *4*, 136–141. [CrossRef] [PubMed]

-
32. Haghighi, S.; Jasemi, M.; Hessabi, S.; Zolanvari, A. PyCM: Multiclass confusion matrix library in Python. *J. Open Source Softw.* **2018**, *3*, 729. [[CrossRef](#)]
 33. Oshiro, T.M.; Perez, P.S.; Baranauskas, J.A. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 154–168.
 34. Chalmers, M. Research Guides: MATLAB Resources: MATLAB Advantages. Available online: <https://guides.libraries.uc.edu/c.php?g=461109&p=3152738#:~:text=Matlab%20Advantages> (accessed on 6 April 2021).
 35. Kotenko, I.V.; Saenko, I.; Branitskiy, A. Applying Big Data Processing and Machine Learning Methods for Mobile Internet of Things Security Monitoring. *J. Internet Serv. Inf. Secur.* **2018**, *8*, 54–63.