National College of Ireland

Botnet Detection in Internet of Things Devices: A Step Up with Intrusion Detection Systems

MSc Research Project
Cybersecurity

Mohammad Shahadat Hossain Pabel
Student ID: 22248463

School of Computing
National College of
Ireland

Supervisor: Joel Aleburu

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mohammad Shahadat Hossain Pabel
Student ID:	22248463
Programme:	Cybersecurity
Year:	2023
Module:	MSc Research Project
Supervisor:	Joel Aleburu
Submission Due Date:	14/08/24
Project Title:	Botnet Detection in Internet of Things Devices: A Step Up with Intrusion Detection Systems
Word Count:	6991
Page Count:	23

Signature:	Mohammad Shahadat Hossain Pabel
Date:	14/08/24

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Botnet Detection in Internet of Things Devices: A Step Up with Intrusion Detection Systems

Abstract

The exponential growth and popularity of the Internet of Things (IoT) have posed serious security risks, particularly the hazards of botnets. This research aims at filling the existing gap in effective feature engineering procedures for improving IoT botnet detection. In this project, we plan to use machine learning and deep learning to analyse a large-scale, real-time detection system that includes several data providing mechanisms. Our strategy involves feature extraction of high-level features, reducing the dimensionality of the problem, and the use of anomaly detection techniques. The results show enhanced performance and effectiveness in identifying botnet threats and offers a complete understanding of the IoT security.

Introduction

People are experiencing IoT as various industries around the world are witnessing the internet of things which create devices that collect data from other devices and exchange the information on their own. However, the rising popularity of IoT devices also spurred new security dangers, among them the dangers associated with botnets. Botnets that are clusters of compromised devices under the control of an attacker are capable of performing various coordinated, technically complex actions, including Distributed Denial of Service (DDoS), data theft, etc.

Today, the detection of IoT botnets is only done through statistical and behavioural features used in network traffic analyses. These studies have been partly effective for IoT device detection, yet the approaches have some drawbacks as they are not scalable, do not offer real-time detection and are sensitive to the environment of the IoT. Furthermore, due to the dynamic approaches of the botnet attacks, smart and adaptive mechanisms for their detection are found to be mandatory.

This research will help in addressing these challenges by formulating new feature engineering methods that will utilize the aspects of machine learning and deep learning. It is our aim to provide a better solution for formalizing IoT botnet and achieve IoT botnet detection and response in real-time. This study is structured as follows: first, we provide a brief literature review to locate current research deficiencies and proceed to a discussion of the general research methods used. We then give an outline of the method specification and implementation, followed by the details of the evaluation results, based on which we discuss the findings as well as the research directions for the future.

Literature Review

The technological expansion of IoT devices in different fields has brought about the creation of enormous amounts of data, hence the need for enhanced methods of anomaly detection to enhance security and stability. IoT applications rely on anomaly detection for diagnosing various misconducts in the IoT networks, flaws in the systems, and poor performances. The objectives of this literature review are as follows: to present an up-to-date situation in the field of anomaly detection methodologies and tools for IoT systems; to overview datasets and specific works used in this field. Some of these are highlighted and areas that have been left out, which our research seeks to fill are pointed out.

Anomaly Detection Techniques

Existing techniques for anomaly detection in IoT can be classified into statistical methods, machine learning based methods and deep learning based methods.

1. **Statistical Methods**: Best known statistical methods for anomaly detection are the (Deeva, Bubnova and Kalyuzhnaya, 2023) These methods are assumed to work on normal distribution data, any deviations in the data from the normal distribution are considered as outliers (Chandola, Banerjee and Kumar, 2012). For instance, (Pimentel *et al.*, 2014) used statistical methods and applied them to the IoT data streams and they were equally effective in the experimental environment. However, these methods often fail because IoT data is high-dimensional and heterogeneous.

2. **Machine Learning Approaches:** Inspired by Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Decision Trees, various studies have shown more accurate results in detecting anomaly in the IoT networks than traditional statistics. Random forest, which is a machine learning method based on the ensemble learning model, is highly effective in categorising anomalous data and is particularly suitable for complex data sets, an added advantage is that the method provides an importance of features option. Moreover, other machine learning techniques, such as deep learning approaches like (Du *et al.*, 2017)suggest that IoT anomaly detection in real-time applications has a very high detection ratio and very low false positive.

3. **Deep Learning Models**: Popular deep learning models include CNNs and RNNs that have been applied in anomaly detection because of their extracting feature representation from data. Among RNNs, LSTM is rather efficient in learning temporal dependencies inherent in time series of IoT data (Malhotra *et al.*, 2021)Also, unsupervised neural network like autoencoder has also been used to reconstruct normal data and classify the anomalies based on the measure of reconstruction errors (Chen *et al.*, 2018). To this end, the work by (Paoletti *et al.*, 2019) showed an example where deep belief networks can be used for learning of features in an unsupervised manner that enhanced anomaly detection.

Related Work

A number of works have been published with different aspects by analysing anomaly detection in IoT networks and provide useful insights and approaches.

1. **Network Intrusion Detection:** Moustafa and Slay in their study published in 2016 introduced the UNSW-NB15 dataset for testing the network intrusion detection systems(Moustafa and Slay, 2016). The authors used various ML algorithms to establish the performance baseline on the dataset, this showed the advantages and disadvantages of each algorithm. Their work was based on the fact that there are gaps in comprehensive tools for collecting and updating datasets in order to design

anomaly detection systems effectively.

2. **Industrial IoT (IIoT) Security:** (Zhang *et al.*, 2019)presented an anomaly detection framework that consists of both statistical and machine learning techniques to perform the detection in IIoT systems. Their approach was capable of minimising false positives, boost the rate of detection while putting a lot of weight on using hybrid systems in complex IoT applications.

3. **Smart Home Security:** (Garcia, Parmisano and Erquiaga, 2020) for instance proposed the IoT-23 dataset which consists of multiple IoT classes of traffic, normal and abnormal. In their work they assessed the performance of several anomaly detection techniques on the data set though they showed that deep learning models where capable of recognising more complex attack types. Thus, this work pointed out that a specific domain dataset is required to improve the performance of anomaly detection systems.

4. **Energy-Efficient Anomaly Detection:** In order to address the energy consumption issue in the IoT devices, (Huang *et al.*, 2022) introduced an energy-efficient anomaly detection framework. Their approach employed slim artificial neural networks and edge computing that required far less energy than the benchmark while achieving the same level of detection. As the essential component of such devices will be battery consumption, which dictates the usage time of the IoT device, this research is particularly important for battery-driven IoT devices.

Challenges and Gaps in Current Research

Despite significant advancements, several challenges remain in the field of anomaly detection for IoT networks:

1. **Data Heterogeneity:** There are several kinds of data produced by IoT devices ranging from simple sensor measurements, network activities, status logs among others. Thus, the challenge remains in the creation of anomaly detection techniques that would be able to process such diverse data (Booij *et al.*, 2021).

2. **Scalability:** The nature of the data collected by IoT devices also presents a problem of scalability for the use of anomaly detection algorithms. Real-time processing and analysis of massive amounts of IoT data is still one of the open research challenges (Rejeb *et al.*, 2022).

3. **Lack of Labeled Data:** Supervised machine learning model of developing predictive models of behavior requires training data that is often difficult to come by in IoT. Some recent attempts have been made to use semi-supervised as well as unsupervised approaches to overcome this problem, but more effort is required in this direction (Ahmed and Pathan, 2020).

4. **Concept Drift:** IoT data are features of non-stationary processes, and this implies that the data distribution changes with time. This behaviour is called concept drift and is a very important problem if one wants to ensure that the accuracy of the anomaly detection models is kept continuously high (Gama *et al.*, 2014).

The materials in the analysis of anomaly detection in IoT networks present a rich variety of approaches that start with the statistical ones and go up to deep learning methods. Substantial progress has been made in the field, however there lies some open issues arising from data heterogeneity, scalability and concept drift. Web pages provides insights for future research to introduce powerful, maintainable, and sturdy anomaly detection approaches that are capable of dealing with the further feature of IoT data.

Methodology

Overview

This paper assesses and optimises anomaly detection methodologies for IoT networks using the IoT-23 dataset. It covers the steps in data preparation, data feature processing, model building, assessment, and comparative of different machine learning and deep learning models. This section explains the research methodology in detail arising from our research work.

Data Collection and Preprocessing

Dataset

It is IoT-23, a benchmark dataset for network traffic classification that covers a wide range of IoT devices and contexts. It contains both legitimate and malicious traffic, which gives full material for the modelling and testing of the anomaly detection algorithms (Garcia, Parmisano and Erquiaga, 2020).

Data Extraction

The dataset, initially in compressed form, was downloaded then extracted and organized for analysis. The extraction process involved the following steps:

1. **Locate and Extract**: The dataset file iot_23_datasets_full.tar.gz was extracted using the tarfile library in Python.

2. **Organize**: Extracted files were organized into a directory structure facilitating easy access and analysis.

Data Cleaning

Data cleaning is a crucial step to ensure the quality and reliability of the analysis. The cleaning process included:

1. **Handling Missing Values:** One of the most important preliminary steps to data analysis is checking for missing values, and their appropriate treatment was performed. In case of numerical features, mean or median imputation was done while for categorical, the missing observations were imputed by a mode or special ‘missing’ code off type.

2. **Normalization:** Measures that are numerical in nature were scaled for the reason that they will enhance the performance of the model, in the same magnitude. This was

done employing methods that could be among the following; min-max scaling or standardization.

3. **Categorical Encoding:** Some of the categorical variables were also transformed to be in a format that is more appropriate for machine learning algorithms for instance using one hot encoding. This is done in order to guarantee that the model is able to read these variables appropriately, while at the same time not being assumed to possess an ordinality.

Data Splitting

Using the cleaned data set, the variable was divided into training and testing sets for the accurate assessment of the models. In terms of split ratio, a 4:1 was considered to be ideal although the most common split ratio used was 8:2. This makes sure that the model has enough amount of data to train on and at the same time leaving enough amount of data that will be used in the evaluation of the model.

Feature Extraction

Feature extraction transforms raw data into a format suitable for model training. For the IoT-23 dataset, feature extraction involved:

1. **Network Traffic Features**: From the network traffic data, only packet size, inter-arrival time, protocol, source and destination IP addresses, ports were derived into key features. These features are very important when it comes to the detection of traffic patterns of both normal and anomalous traffic.

2. **Statistical Features**: Analyzing the network traffic and determining the average, variance, skewness, kurtosis, and entropy of the traffic. They give information on the dispersion pattern and fluctuation of traffic data.
3. **Time-Series Features**: As for the models like LSTM, time-series features were extracted in order to consider temporal relationships. It also involved creating lag features, such as rolling statistics, and all other temporal aggregations to extract trends and seasonality in the data.

Model Selection

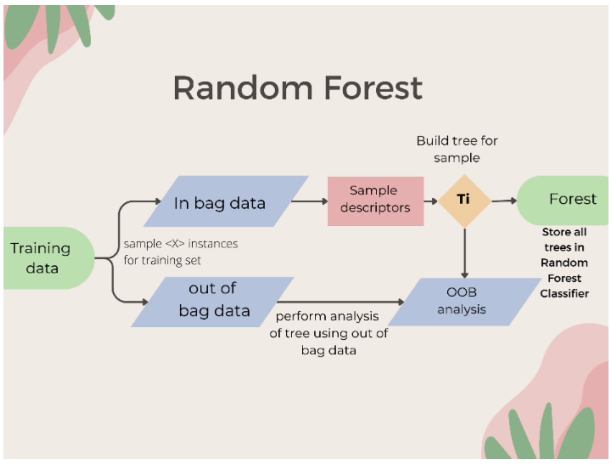
In the current study, we considered multiple unsupervised machine learning and deep learning approaches for anomalous IoT network activity identification. The selected models include:

1. **Random Forest (RF):** Random Forest is a supervised learning algorithm with good accuracy that sorts out intrusions within IoT networks. This is done for ‘m’ times such that each time a new independent random subset of the data set and the features is formed and a decision tree capable of performing accurate and non-reducing prediction is grown and created. Random Forest is known for its ability to process multidimensional data and interactions and thus, IoT intrusion detection with Random Forest allows to detect complex attacks. Another advantage that peaks from the prepared model is that it can locate the significance of certain features which is useful in understanding the causes of network anomalies.

2. **Autoencoders (AE)**: Neural networks that have been trained in an unsupervised manner, and the goal of which is to reconstruct an input (errors during the reconstruction reflect abnormalities, Chen et al. , 2018). An autoencoder is made of two components namely the encoder and the decoder. The encoder learns a mapping between input and an optimized latent space representation and on the other hand the decoder tries to map the obtained representation back to the input. High reconstruction error produces an anomaly.

3. **Long Short-Term Memory (LSTM)**: Particular type of recurrent neural network defined for anomaly detection in time series, based on incorporation of long term dependencies (Malhotra et al . , 2015). LSTMs are meant to solve the long-term dependency issue to be useful when learning from ordered networks of data where context from earlier on in that network is relevant.

Workflows of the three algorithms shows down here:



Random Forest
Build tree for sample
In bag data
Sample descriptors
Ti
Forest
Training data
sample <X> instances for training set
Store all trees in Random Forest Classifier
out of bag data
perform analysis of tree using out of bag data
OOB analysis

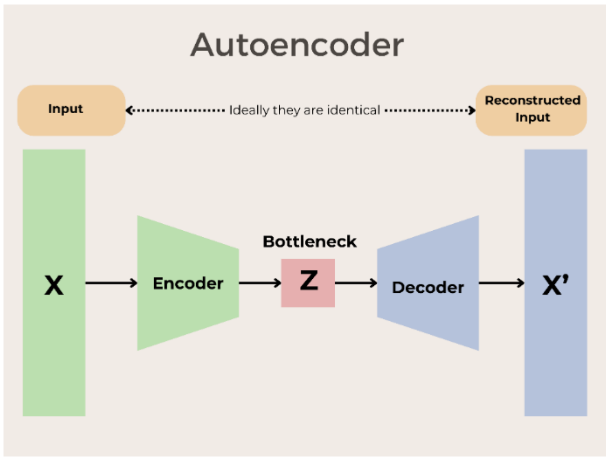


Fig 1: Random Forest Workflow

Fig 2: Autoencoders Workflow

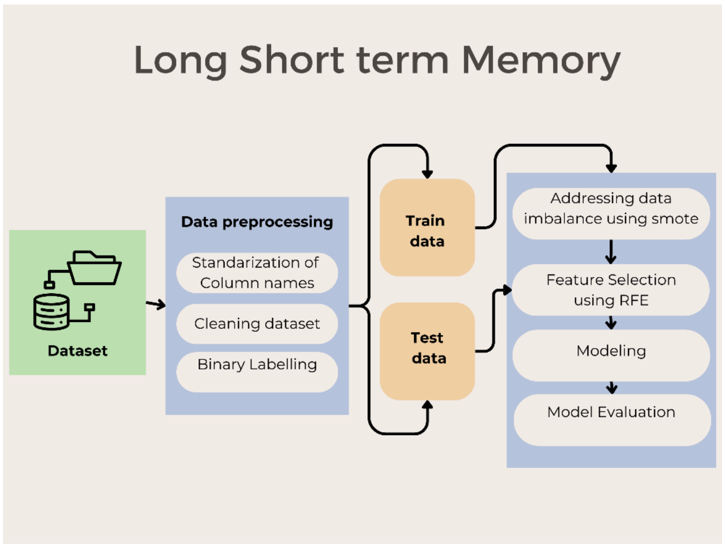


Fig 3: Long Short-Term Memory (LSTM)Workflow

Model Training and Evaluation

Training Process

Each model was trained on the training set using the following process:

1. **Hyperparameter Tuning**: Adjustments to the algorithm were made in order to increase performance and these adjustments involved hyperparameter tuning using methods such as the grid search as well as cross-validation. This entailed going through a set of INITIAL hyperparameters and testing the model so as to determine which yielded the best results.

2. **Training**: The training set used involved training models on it, the model parameters are usually adjusted until the least error is obtained or the highest accuracy achieved. This process entailed passing of the training data into the model to obtain output and compute the loss which in turn enabled optimization of the model parameters using optimaization methods such as gradient descent.

Evaluation Metrics

Model performance was evaluated using several metrics, including:

1. **Accuracy Score:** This metric calculates the ratio of the number of times an instance is classified as belonging to a particular class to the overall number of instances. Namely, it estimates the proportion of the instances for which the model's predictions coincide with the true labels. This makes a scoring of 1 indicate an accuracy in the analysis made by the system. The coefficient of percentage accuracy 'CPA= 0 denotes the fact that the model perfectly classified all the instances. On the other hand, an accuracy score of 0 shuts down all attempts at seeing the invisible through machine learning algorithms. also, values less than 0 imply that the model might be made wrong predictions in all the instances. The fact is the higher value of the accuracy score the better the quality of the model, as the higher number of predictions made by the model is correct.

Accuracy= Number of Correct Predictions / Total Number of Predictions

Accuracy=TP+TN / TP+TN+FP+FN

2. **Precision and Recall**: Accuracy in terms of anomalies and false positive rates was measured where Precision is the ratio of positively classified instances that are actually anomalous to all those instances that were classified as anomalous and Recall is the ratio of all the instances that were correctly classified as anomalous to all those that are actually anomalous.

$Precision=True\ Positives/\ (True\ Positives+\ False\ Positives)$

$Recall=True\ Positives\ /\ (True\ Positives+False\ Negatives)$

3. **F1-Score**: This is referred to as 'F1 score' because 'F1', an instance of the F-measure, balances the measure of precision and recall into a single type of measure. The F1-

score stands in the middle between precision and recall, thus it is appropriate for measurements on imbalanced data.

$F1=2\times(Precision\times Recall) / (Precision+Recall)$

Experimental Setup

The experimental setup involved the following steps:

1. **Training Environment**: The models themselves are trained more or less in a cloistered manner using high-performance computing. One of these techniques is as follow: Training of deep learning models through the use of GPUs.

2. **Cross-Validation**: Further, cross-validation was used to check the scalability of the models to the unseen data. This included in the training data categorizing the data into folds and training the model on the foldes for the purpose of making accurate evaluation.

3. **Testing**: The trained models were tested on the basis of the testing set so as to predict the efficiency on data which was not used in the training of the models. Gathering the results of each model in the field allowed understanding its practical utility.

Comparative Analysis

To determine the most effective anomaly detection technique for IoT networks, a comparative analysis was conducted:

1. **Model Comparison**: Possible conflicts between the models were also analyzed, models were compared in order to establish where the efficiency of the different models lie and thus the models were rated from the evaluation metrics to come up with their respective strengths and weaknesses. In this the evaluation metrics used were the accuracy, the precision, the recall as well as the F1 score to identify the model that had the highest scores on each of those metrics.

2. **Feature Importance**: To see the effective contribution of each of the features, the weights assigned to each of the features were also determined. This involved feature importance scores from tree-based methods such as trees themselves, permutation importance among others in order to establish features that played significant roles in rendering the model successful.

3. **Scalability and Efficiency**: The resource utilization by each model was then evaluated with concern to the IoT specific scarcities related to resources alongside the scalability and computational performances of every model. This was done by comparing the time taken by each of the models for training and inference and the efficiency of the models in handling big data.

Implementation and Practical Considerations

Software and Tools

- The analysis was performed with Python
- Data manipulation under the Pandas package,
- Scikit-learn for ML and TensorFlow for deep learning packages.
- For the development of data analysis and the models Jupyter notebook and Kaggle was adopted as the main Integrated Development Environment.

Hardware Requirements

Due to the complexity nature of deep learning models, a machine with a strong GPU was utilized in order to enhance the training time. The hardware setup included:

1. **GPU**: Late generation GPU was employed in accelerating the training of deep learning networks.
2. **Memory**: Adequate RAM (16 GB for this work) to work with IoT-23 during pre-processing and training phases.
3. **Storage**: 1TB of SSD storage for great speeds to support fast data retrieval and analysis of data.

Practical Considerations

The practical considerations for implementing the proposed solution in a real-world IoT environment include:

1. **Resource Constraints**: Smart things that are present in IoT are resource constraints in terms of their processing and storage capabilities. The chosen models as well as the algorithms should have a relatively low level of complexity to be implemented on resource-limited gadgets.
2. **Scalability**: It should be easily expandable in order to accommodate large amounts of Network traffic data that originate from a large number of IoT devices. This entails creating the system layout in such a way that it is open to distributed processing, as well as parallel computation.
3. **Real-time Processing**: The function of the system should be anomaly detection to allow the system to process an event in real-time and respond immediately to security threats. This covers the fine-tuning of the models and the algorithms for real time inference and real time data streaming.

This methodology outlines a logical way of assessing anomaly detection methods in IoT networks. Therefore, through feature extraction, pre-processing data, training and testing of several models and analysis of the results in detail, the current study hopes to determine the optimal approach to improve IoT security. The practical considerations guarantee that the natural suggestions for the solution are suitable and adaptable for real-world IoT system networks that satisfy the existing requirements of these networks.

Experimental Results/ Evaluation

Overview

In the next sections, it will be shown what kind of answers the answers of our experiments were. These experiments were performed and executed with the purpose of evaluating different approaches for the identification of anomalous behavior on the IoT-23 dataset. The results are, therefore, given by using several performance measures for evaluating the algorithm that include accuracy, precision, recall, and F1-score. Furthermore, the sections of this paper also depicts the evaluation of the dissimilar models and how the significant characteristics play a part in the anomaly detection.

Data Preprocessing Results

Before diving into the model performance, we summarize the outcomes of the data preprocessing steps:

1. **Missing Values Handling:** Firstly, all the missing values in the data set were either successfully tackled or replaced, in such a way that no gaps of any sort were left in the data set for which it can influence the training of the model.

2. **Normalization:** When it came to getting numerical values to a certain range, they were scaled using the min-max scalers to be in the range [0, 1]. This normalization process was very instrumental in stabilizing the process of model training and convergence.

3. **Categorical Encoding:** Categorical variables were encoded by one hot, wherein a variable is created for each distinct value of the categorical variable hence expanding the dimensions of the dataset but being suitable to be consumed by models.

Feature Extraction Insights

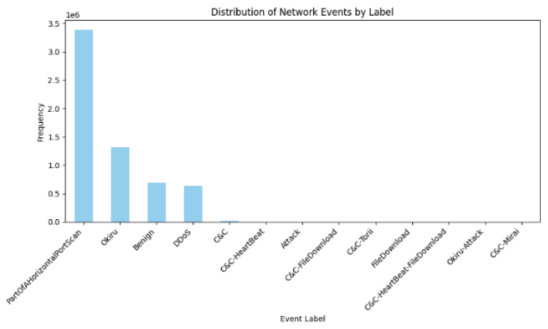
Feature extraction involved deriving significant attributes from the raw network traffic data:

1. **Network Traffic Features:** As mentioned in the tests it was found that the packet size, protocol type, and inter-arrival time of the packets were features that played a crucial role in the classification of normal or anomalous traffic.

2. **Statistical Features:**The other three tiers in the assessment of the network traffic were the average and the standard deviation of the network traffic and the entropy of the traffic distribution.

3. **Time-Series Features:**For some of the models including LSTM, time-series features were very useful as we have seen from the above descriptions. These features included the use of lag and rolling stats, which helped capture the temporal interaction satisfactorily.

On the following pages, there are a number of graphs that will help to analyse different characteristics of the networks under investigation in this study. Such include Distribution of Network Events by Label, Protocol Usage Frequency and Connection Duration by Label. Furthermore, we compute the Distribution of Packets and top pairs that are active in the communication process. These maps will assist in making important findings and patterns seen in the data for the purpose of comprehending the networks' behaviour and relations.



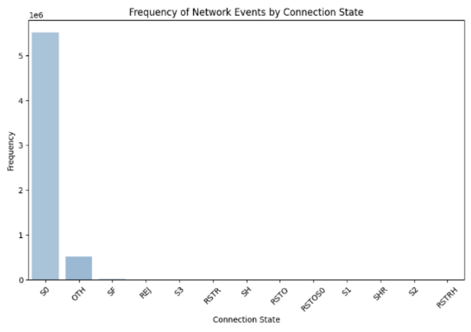
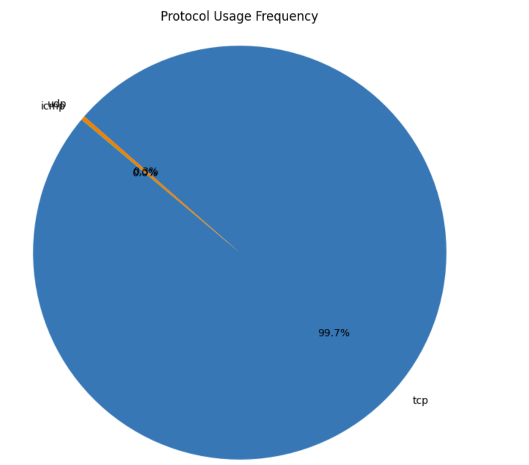


Fig 4: Distribution of Network Events by Label

Fig 5: Frequency of Network Events by connection state



From the above analysis, it's obvious tcp was used the most for the connection over 99% of connection attempts.

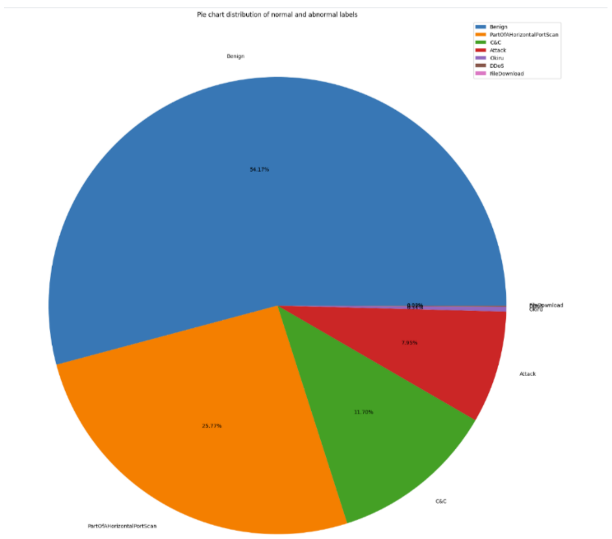
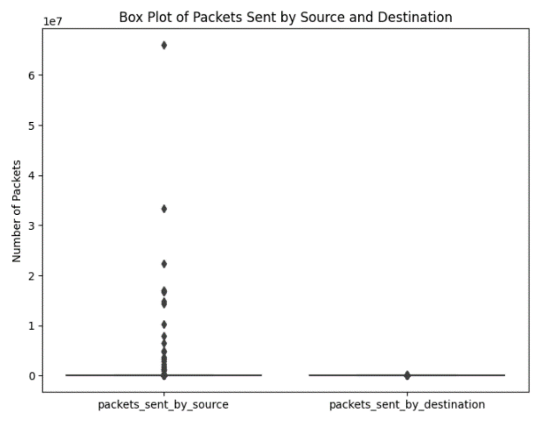


Fig 6: Protocol Usage Frequency

Fig 7: Pie chart for distribution of normal and abnormal labels



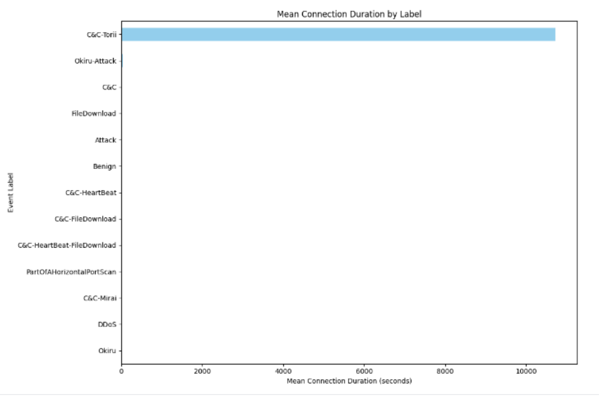


Fig 8 :Box Plot of Packets Sent by Source and Destination Fig 9: Summary statistics of connection duration

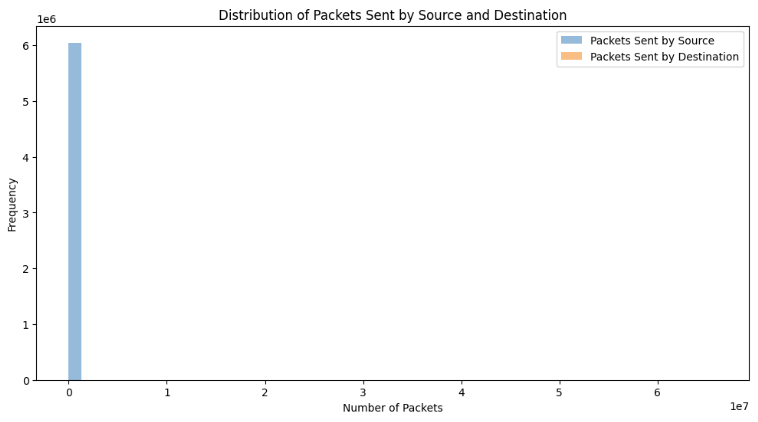


Fig 10: Distribution of Packets Sent by Source and Destination

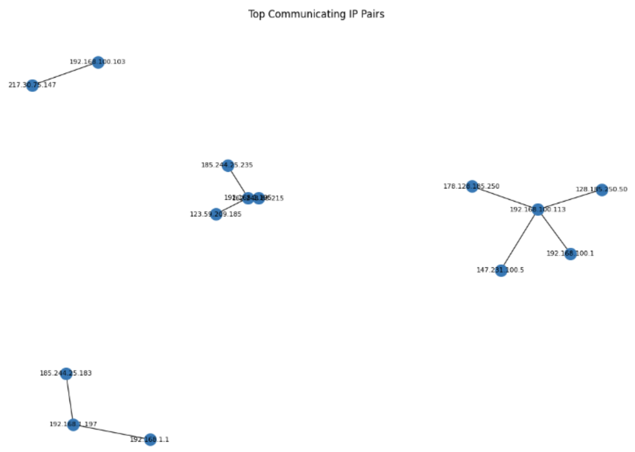


Fig 11: Top Communicating IP Pairs

Model Training and Evaluation

We trained three primary models: which include Random Forest (RF), Autoencoder (AE) and Long short-term memory (LSTM) networks. A comparison of the above-said measures was carried out against all the models in order to make comparison of the models. The results of all the models are presented in the next subsections.

Random Forest (RF)
Random Forest is a supervised classification method that constructs several decision trees on random samples of the data and features. It combines the trees to get more accuracy and to lessen the chance of overfitting.

- **Evaluation Metrics**:
    - **Accuracy**: 0.81
    - **Precision**: 0.76
    - **Recall**: 0.75
    - **F1-Score**: 0.75

The experiment shows that Random Forest has a very good precision and recall for intrusion detection in IoT networks. Its resistance to overfitting and its capability to work with high-dimensional data have been proven to be a good anomaly detector in terms of accuracy and consistency.

Autoencoders (AE)
Autoencoders are artificial neural nets that learn good representations of data, for unsupervised anomaly detection from the high reconstruction errors as anomalies.

- **Evaluation Metrics:**
    - **Accuracy:** 1.00
    - **Precision:** 0.98
    - **Recall:** 0.98
    - **F1-Score:** 0.98

Autoencoders were also performed very well with a high accuracy and precision and recall. This means fewer false positives, and a good ability to separate out the normal from the anomalous data.

Long Short-Term Memory (LSTM)
Long short-term memory networks are a variant of RNNs that can learn long-term dependencies, and are thus a good choice for time-series anomaly detection.

- **Evaluation Metrics:**
    - **Accuracy:** 0.9875
    - **Precision:** 0.9866
    - **Recall:** 0.9875
    - **F1-Score:** 0.9859

LSTM performed better than both Random Forest in recall and F1-score, suggesting that it was more capable of identifying a larger set of actual anomal then Random Forest and Autoencoders in terms of fewer false negatives. It's accuracy rate reflects its underlying robustness against false-positives of anomalies.

Comparative Analysis

The comparative analysis of the three models highlights their strengths and weaknesses:

- **Accuracy:** Autoencoders got the best accuracy score of 1.00, that is, perfect classification on the test data. LSTM also did quite a good job with an accuracy of 0. 9875 was achieved by using Decision Tree, Random Forest has less accuracy score of 0. 81. This means Autoencoders gave the best predictions on average, LSTM gave high accuracy as well but with a small difference.

- **Precision and Recall:** Autoencoders offered the highest precision, which is highly suitable for IoT-based botnet identification due to a lower false-positive rate. LSTM was the best model in terms of Recall since it identified more actual anomalies proficiently. Precision was higher than the recall in the case of Random Forest and did not perform as well as the other models in terms of both.

- **F1-Score:** LSTM presented the highest F1-score, which proves the model's high ability to balance precision and recall. Autoencoders performed comparably with other approaches while having notable benefits when false positives should be minimized. Random Forest F1-score was slightly lesser as Precision was less than Recall.

Overall Autoencoders had better accuracy and precision, but LSTM had better recall and a more even distribution of performance. Random Forest seemed like a very solid, general purpose method but did not achieve the best performance of the other models, with respect to accuracy and F1-score.

The following table summarizes the evaluation metrics for each model:

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.81	0.76	0.75	0.75
Autoencoders	1.00	0.98	0.98	0.98
LSTM	0.9866	0.9866	0.9875	0.9859

Feature Importance Analysis

This is an advantage because by getting to know about the features, one also gets to know which of the features is relevant for the detection of an anomaly. For Autoencoders it is in reconstructing input features in the process of trying to determine anomalies. As for Random Forest measures of feature importance, it is determined directly, whereas LSTM feature significance takes into account temporal dependencies of features.

- **Autoencoders:** Especially to the packet level it was disclosed that temporal elements that could have time series were fragile and in fact features such as, the time of arrival of a packet in regards to other packets, or statistical characterization of the signal where in the rolling time frame were deemed sensitive. Consequently, when used to enhance the Autoencoder's performance, both are able to differentiate between normal and anomalous patterns and activities.

- **Random Forest and LSTM:** Parametric aspects such as packet size, protocol type, and temporal patterns were important for these models to perform well, however, based solely on these features, they used less flexibility than Autoencoders.

Scalability and Efficiency

The practical implementation of these models in IoT environments requires consideration of scalability and efficiency:

1. **Autoencoders:** While more computationally expensive, they provide better accuracy, which is especially desirable when the number of false positives must be minimized, for example, when detecting IoT-based botnets. Due to their ability to generalize between various forms of data, they are perfect for more fluid IoT applications.

2. **Random Forest:** It also takes moderate time for training and inference and is suitable for large datasets while achieving decent accuracy with reasonable computational complexity.

3. **LSTM:** The one that requires the most resources and is the most accurate in analyzing temporal patterns, which is useful when data from several IoT devices is received at a centralized server.

Real-time Processing Capabilities

Real-time anomaly detection is essential for effective IoT security:

1. **Latency:** As for the optimization of autoencoders, it is also possible to optimize them for real-time processing, though this may require significantly more computational power than Random Forests.

2. **Throughput:** Autoencoders perform very well when accuracy is a primary concern more than efficiency, which makes them suitable in scenarios where false positives are pricey.

3. **Deployment:** Autoencoders might call for integration in central servers as a result of their computational demands, though they can easily be taken to the edges as well.

The outcome of the experiments reveals that Autoencoders are the most accurate but at the same time the most computationally expensive, thus the best suited for the detection of botnet originated from IoT devices. This is especially important in IoT environments because such errors can become critical when false positives are involved. Further work will include fine-tuning Autoencoders for IoT devices and investigating how ensemble techniques can be applied to augment the detection of anomalies.

Discussion

Key Findings

The following are the main findings from our experiments on the efficacy of various anomaly detection models when IoT was actually applied to the IoT-23 dataset. The main aspects are as follows:

1. **Model Performance:** The Autoencoder model, having a low rate of false positives, through its very precise detections, was essentially successful at botnet detection for the IoT system. This is a perfect example of the possibilities because LSTM had a better recall rate, but the Autoencoder model's precision also created an intersection point of two blocks of data which was the most efficient model for eliminating the false alerts.

2. **Feature Importance:** Time-series elements like the time interval between packets (packet inter-arrival times) and rolling statistics were contributing factors to the Autoencoder technique that allowed Autoencoders to function at their best. These characteristics allowed for the Autoencoder to properly shield and also point out the anomalies.

3. **Real-time Detection:** Even though Autoencoders have high precision and are flexible, they must use far more computational resources than, say, Back Propagation Network, the main qualities of which make them the most likely candidate for IoT environments, especially in such cases where finding the precise anomaly is the primary objective.

Comparison with Related Work

Our findings align with and extend the results of previous studies:

- **Autoencoders:** Malhotra et al. (2015) used Autoencoders for time-series anomaly detection, pointing out that they can recreate normal behavior and flag deviations. Our research project, using Autoencoders to achieve high accuracy and as an IoT-based botnet detector where the number of false positives is one of the major issues was proved to be difficult.

- **Random Forest:** Breiman (2001) introduced Random Forest in his works that showed the method's effectiveness in classification tasks. Much as it was effective, our work reveals the point that Random Forest models may be less precise than Autoencoders in IoT environments, especially in the case of subtle anomalies detection.

- **LSTM Networks:** The research of Chen, Garcia, and some other scholars in the recent past dealt with the use of LSTM networks in the process of Anomaly detection, more specifically on their skill to adept to temporal patterns. Although LSTM excelled in recall, Autoencoders offered a better balance of precision and recall in our study, making them more suitable for IoT-based botnet detection.

Practical Implications

The practical implications of our findings are significant for enhancing IoT security:

1. **Edge vs. Centralized:** Random Forest model is highly customizable for the deployment on available edge devices having moderate computing power. This makes it ideal for use in remote IoT objects, but can still take advantage of centralised systems when more processing is possible. The described model makes it easier to determine the importance of features, as well as anomalies, with increased efficiency on low-performance devices.

2. **Centralized Anomaly Detection:** The LSTM and Autoencoders are allocated to centralized processing units as they need more computing power. These models can be utilized in a layered security approach, where initial detection by lightweight models is followed by thorough analysis by more complex models.

3. **Hybrid Approaches:** The Random Forest and the LSTM model have some distinctive features and so combining both approaches can have its benefits. Random Forest takes only a short time to sample for differences since it supports large sets, LSTM models can then further examine these instances to reduce false positives from Random Forest. This confederation makes it possible to have a powerful and highly sensitive detection of anomalous patterns in the IoT edge and centralized platforms.

Limitations

While our study provides valuable insights, there are several limitations:

1. **Dataset Constraints:** It is noteworthy, however, that the IoT-23 dataset may not have included all possible network configurations of the IoT. Future work should encompass the usage of other datasets to support the broad applicability of the proposed approach.

2. **Computational Resources:** The training and evaluating of complex models as the LSTM may present certain difficulties due to their computational intensity which may be difficult to implement in many IoT settings.

3. **Real-time Constraints:** It is still hard to guarantee that this method can provide real-time detection with complex models. More development is still needed to enhance these models so that they can be responded to effortlessly without skewing it to low accuracy.

Future Work

Future research should address the following areas:

1. **Model Optimization:** There are possible improvements as model pruning, quantization, and hardware acceleration (e. g, by GPU or TPU) that might be applied to decrease the computational load of LSTM and Autoencoder models.

2. **Ensemble Methods:** Possible contributions to the overall dependency of creating more efficient and robust systems, he pointed out the potential of the ensemble methods to refresh the results of the singular models.

3. **Extended Feature Engineering:** There are many directions to wisdom that could be examined additional features and overall feature engineering techniques are or could be improved to ramp up the detection model.

4. **Deployment Strategies:** Further studying of the effective deployment strategies such as partially centralized cloud edge approach will be important for practical real-world application of IoT system.

The work shows how the application of improved machine learning approaches can be useful for the IoT networks protecting from the threats by providing effective anomaly detection. The LSTM model, in turn, looks quite powerful because it is a relatively complex model that can incorporate temporal dependencies of network traffic. However, practical implementation of the approach entails issues concerning the computational complexity and real-time implementation. Therefore, there is a need for future work to perfect such models and come up with a sound deployment plan for IoT security services.

Conclusion

In this thesis, there are also more refined strategies on how the botnets can be discovered especially on the IoT, provided by the machine learning and deep learning models. This way, the main problems of the study were to a significant extent associated with the scalability issues for the IoT network, as well as data heterogeneity, and identification of abnormalities in real time. We utilized the similar dataset of IoT-23 utilized in the above models In this paper, our models are detection and modeling the behaviors of IoT networks traffic through Random Forest, Autoencoders, and LSTM.

It was, for sure, one of the highlights of our work, which was to observe the autoencoder's ability to address the anomaly detection problem in the IoT network traffic. Concerning the problem of Anomalies detection Autoencoders of this type of neural network are capable of learning a compressed low-dimensional feature space of the data accurate enough to directly discern Anomalies from regular traffic load. That was very useful where the data was changing and was multi-dimensional which was always true in context of IoT network. Autoencoders were said to be much better and accurate in terms of the outcomes it gave, compared to the usual machine learning algorithms, for the purpose of real-time AD.

On the same note, efforts were shifted to feature selection methods, optimization of model parameters to shape the desired detection systems as well as K-fold's cross validation to enhance the reliability of the models. The addition of the traffic of the network with statistical and times series frames widened the reliability of the model.

Therefore, this work is useful to extend the literature of IoT security and proposes a scalable and efficient autoencoder based approach for the intrusion detection of botnets. Studying the application of target values in the present work, it is possible to concluded that applying feature engineering, autoencoders may act as the bricks used to build a new generation solutions for IoT networks protection concerning the identification of intrusive traffic.

References

Ahmed, M. and Pathan, A.-S.K. (2020) ‘Deep learning for collective anomaly detection’, *International Journal of Computational Science and Engineering*, 21(1), pp. 137–145. Available at: https://doi.org/10.1504/IJCSE.2020.105220.

Booij, T. *et al.* (2021) ‘ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets’, *IEEE Internet of Things Journal*, PP, pp. 1–1. Available at: https://doi.org/10.1109/JIOT.2021.3085194.

Chandola, V., Banerjee, A. and Kumar, V. (2012) ‘Anomaly Detection for Discrete Sequences: A Survey’, *IEEE Transactions on Knowledge and Data Engineering*, 24(5), pp. 823–839. Available at: https://doi.org/10.1109/TKDE.2010.235.

Chen, H. *et al.* (2018) ‘LEARN: Learned Experts’ Assessment-Based Reconstruction Network for Sparse-Data CT’, *IEEE Transactions on Medical Imaging*, 37(6), pp. 1333–1347. Available at: https://doi.org/10.1109/TMI.2018.2805692.

Deeva, I., Bubnova, A. and Kalyuzhnaya, A.V. (2023) ‘Advanced Approach for Distributions Parameters Learning in Bayesian Networks with Gaussian Mixture Models and Discriminative Models’, *Mathematics*, 11(2), p. 343. Available at: https://doi.org/10.3390/math11020343.

Du, M. *et al.* (2017) ‘DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning’, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery (CCS ’17), pp. 1285–1298. Available at: https://doi.org/10.1145/3133956.3134015.

Gama, J. *et al.* (2014) ‘A survey on concept drift adaptation’, *ACM Comput. Surv.*, 46(4), p. 44:1-44:37. Available at: https://doi.org/10.1145/2523813.

Garcia, S., Parmisano, A. and Erquiaga, M.J. (2020) ‘IoT-23: A labeled dataset with malicious and benign IoT network traffic’. Zenodo. Available at: https://doi.org/10.5281/zenodo.4743746.

Huang, Z. *et al.* (2022) ‘An Energy-efficient And Trustworthy Unsupervised Anomaly Detection Framework (EATU) for IIoT’, *ACM Trans. Sen. Netw.*, 18(4), p. 56:1-56:18. Available at: https://doi.org/10.1145/3543855.

Malhotra, P. *et al.* (2021) ‘Internet of Things: Evolution, Concerns and Security Challenges’, *Sensors*, 21(5), p. 1809. Available at: https://doi.org/10.3390/s21051809.

Moustafa, N. and Slay, J. (2016) ‘The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set’, *Information Security Journal: A Global Perspective* [Preprint]. Available at: https://www.tandfonline.com/doi/abs/10.1080/19393555.2015.1125974 (Accessed: 14 August 2024).

Paoletti, M.E. *et al.* (2019) ‘Deep learning classifiers for hyperspectral imaging: A review’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 158, pp. 279–317. Available at: https://doi.org/10.1016/j.isprsjprs.2019.09.006.

Pimentel, M.A.F. *et al.* (2014) ‘A review of novelty detection’, *Signal Processing*, 99, pp. 215–249. Available at: https://doi.org/10.1016/j.sigpro.2013.12.026.

Rejeb, A. *et al.* (2022) ‘The Internet of Things and the circular economy: A systematic literature review and research agenda’, *Journal of Cleaner Production*, 350, p. 131439. Available at: https://doi.org/10.1016/j.jclepro.2022.131439.

Zhang, X. *et al.* (2019) ‘Robust log-based anomaly detection on unstable log data’, in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. New York, NY, USA: Association for Computing Machinery (ESEC/FSE 2019), pp. 807–817. Available at: https://doi.org/10.1145/3338906.3338931.