Dissertation on

Anomaly Detection and Classification of IoT Attacks using Autoencoders and Multi-Output DNN

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE19CS390B – Capstone Project Phase - 2

Submitted by:

Maria Abraham Pynadath	**PES2UG19CS224**
K J Pavithra	**PES2UG19CS278**
Sahil Elton Lobo	**PES2UG19CS348**
Sanjana S Murthy	**PES2UG19CS364**

Under the guidance of
Dr. Bharathi R
Professor
PES University

June - Nov 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

Anomaly Detection and Classification of IoT Attacks using Autoencoders and Multi-Output DNN

is a bonafide work carried out by

Maria Abraham Pynadath	**PES2UG19CS224**
K J Pavithra	**PES2UG19CS278**
Sahil Elton Lobo	**PES2UG19CS348**
Sanjana S Murthy	**PES2UG19CS364**

In partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE19CS390B) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June 2022 – Nov. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature	Signature	Signature
Dr. Bharathi R	Dr. Sandesh B J	Dr. B K Keshavan
Professor	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ____________________________	____________________________
2. ____________________________	____________________________

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Anomaly Detection and Classification of IoT Attacks using Autoencoders and Multi-Output DNN"** has been carried out by us under the guidance of Dr. Bharathi R, Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June – Nov. 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2UG19CS224	**Maria Abraham Pynadath**
PES2UG19CS278	**K J Pavithra**
PES2UG19CS348	**Sahil Elton Lobo**
PES2UG19CS364	**Sanjana S Murthy**

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Bharathi R, Professor, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE19CS390B - Capstone Project Phase – 2.

I am grateful to the Capstone Project Coordinator, Dr. Sarasvathi V, Professor and Dr. Sudeepa Roy Dey, Associate Professor, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Sandesh B J, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Botnet attacks are responsible for the largest Distributed Denial-of-Service (DDoS) attacks on record. The detection of botnet attacks has become crucial now more than ever due to the emergence of newer botnets and botnet attacks . An increase in vulnerable connected devices, and continued growth in the DDOS-as-a-service industry ensures relevance of botnets as a threat.

IoT devices focus on maximizing functionality whilst reducing resource utilization. Hence, it becomes incredibly difficult for IoT device manufacturers and programmers to include complex cryptographic mechanisms and security measures in the framework of these devices, making IoT devices an easy target for attacks.

Thus, the purpose of the project is to detect unknown attack data from known data and further classify the known data into respective botnets and their attack types which is done in two phases. To perform the unknown attack detection, a dataset has been generated by simulating an unknown botnet attack in a virtualized environment. The first phase involves anomaly detection of unknown attacks using Autoencoders. The second phase consists of multi-output classification of the detected known data using Multi-Output DNN into botnet and attack types. Hence, the proposed approach overcomes the problem of detecting unknown or newer botnet attacks that may emerge. Moreover, the approach performs multi-output classification as well. Thus, proposed method provides a holistic approach to detect IoT attacks including unknown attacks.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	**INTRODUCTION**	**01**
2.	**PROBLEM STATEMENT**	**04**
3.	**LITERATURE REVIEW**	**06**
	3.1 Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection	
	3.1.1 Introduction	
	3.1.2 Characteristics and Implementation	
	3.1.3 Features	
	3.1.4 Evaluation	
	3.2 IoT Botnet Anomaly Detection Using Unsupervised Deep Learning	
	3.2.1 Introduction	
	3.2.2 Characteristics and Implementation	
	3.2.3 Features	
	3.2.4 Evaluation	
	3.3 N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders	
	3.3.1 Introduction	
	3.3.2 Characteristics and Implementation	
	3.3.3 Features	
	3.3.4 Evaluation	
	3.4 Network Flow based IoT Botnet Attack Detection using Deep Learning	
	3.4.1 Introduction	
	3.4.2 Characteristics and Implementation	
	3.4.3 Features	
	3.4.4 Evaluation	
	3.5 Unsupervised Anomaly Based Botnet Detection in IoT Networks	
	3.5.1 Introduction	

3.5.2 Characteristics and Implementation
3.5.3 Features
3.5.4 Evaluation
3.6 A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks
3.6.1 Introduction
3.6.2 Characteristics and Implementation
3.6.3 Features
3.6.4 Evaluation
4. PROJECT REQUIREMENTS SPECIFICATION 11
4.1 Product Perspective
4.1.1 Product Features
4.1.2 User Classes and Characteristics
4.2 Operating Environment
4.3 General Constraints, Assumptions and Dependencies
4.4 Software Requirements
4.4.1 Detection
4.4.2 Attack Dataset Generation
4.5 Non-Functional Requirements
4.5.1 Performance Requirements
4.5.2 Security Requirements
5. HLD Document 15
5.1 Design Considerations
5.1.1 Design Goals
5.1.2 Architecture Choices
5.1.3 Constraints, Assumptions and Dependencies
5.2 High Level System Design
5.3 Design Description
5.3.1 Data Flow Diagram
5.3.2 Swimlane Diagram
5.4 Design Details
5.4.1 Novelty
5.4.2 Performance

5.4.3 Reliability

5.4.4 Application Compatibility

6. LLD Document 21

7. SYSTEM DESIGN (Detailed) 22

7.1 Attack Dataset Generation

7.2 Detection

7.2.1 Autoencoder

7.2.2 Multi-Output Deep Neural Network

7.3 System Design Diagram

8. PROPOSED METHODOLOGY 24

8.1 Unknown Attack Dataset Generation

8.1.1 Attack Simulation

8.1.2 Dataset Generation

8.2 Detection Model

8.2.1 Anomaly Detection

8.2.2 Multi-Output Classification

9. IMPLEMENTATION AND PSEUDOCODE 26

9.1 Unknown Attack Dataset Generation

9.1.1 Simulated Network Setup

9.1.2 Breaking Point Cloud

9.1.3 Packet Capture Using Network Watcher

9.1.4 Packet Analysis Using Wireshark

9.1.5 Data Wrangling using Python Script

9.2 Detection Model

9.2.1 Data Preprocessing

9.2.2 Autoencoder

9.2.3 Multi-Output DNN

10. RESULTS AND DISCUSSION 39

10.1 Unknown Attack Dataset Generation

10.2 Detection Model

11. CONCLUSION AND FUTURE WORK 42

REFERENCES/BIBLIOGRAPHY 44

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS **46**

ANNEXURE I IEEE PAPER DRAFT

ANNEXURE II PROJECT POSTER

LIST OF FIGURES

Figure No.	Title	Page No.
Fig 5.1.	Layered Architecture Diagram	17
Fig 5.2.	Data Flow Diagram	18
Fig 5.3.	Swimlane Diagram	19
Fig 6.	Low Level Design Diagram	21
Fig 7.	System Design Diagram	23
Fig 9.1.	Network Topology Diagram	26
Fig 9.2.	Breaking Point Cloud	27
Fig 9.3.	Azure Network Watcher	28
Fig 9.4.	Wireshark Packet Analysis Tool	29
Fig 9.5.	EDA on N-BaIoT dataset – a. Distribution of Malicious, b. Botnet type and c. Attack type Records	33-34
Fig 9.6.	TSNE – Left: Normal and malicious, Right: Mirai and Bashlite	34
Fig 9.7.	Autoencoder Architecture	36
Fig 9.8.	Autoencoder Details	36
Fig 9.9.	Multi-Output DNN Architecture	38
Fig 9.10.	Multi-Output DNN Details	38
Fig 10.1.	Unknown Attack Dataset	39
Fig 10.2.	Autoencoder Threshold and Accuracy Output	40
Fig 10.3.	Multi-Output DNN accuracies output	40

LIST OF TABLES

Table No.	Title	Page No.
Table 9.1.	Dataset Features	30
Table 9.2.	Feature Calculation Formulae	31
Table 10.1.	Table with Results for Autoencoder and Multi-Output DNN	40

CHAPTER 1

INTRODUCTION

1.1 IoT Botnets

IoT Botnets are a specified attack class of botnets that target IoT devices specifically. An IoT malware infects vulnerable IoT devices in the network it is working in. The CNC (Command and Control) code is spread in these devices through peer to peer and Server-Bot sharing of the code.

1.1.1 Reasons for targeting IoT devices

IoT devices are built in such a way that they are low in performance and low in security. These reasons make it a favourable target for attacking. Introduction of cryptographic measures on these devices are not feasible because of the low computational power capabilities.

1.1.2 Bots

A Bot is any device that is a part of a Botnet, receiving commands from a CNC Server.

1.1.3 CNC Server

When a Botnet needs to start its formation, the first step it must take is identifying a single or multiple Command and Control(CNC) server(s) that act as centres for dissemination of CNC codes and bot formation.

1.1.4 Working of a Botnet

Botnets function based on the commands sent by the Bot Master via the CNC Server. When a CNC Server is created, it scans for vulnerable devices on the network to convert to bots. These devices are then fed CNC codes and then stay dormant until the Bot Master sends commands to attack a target device. On receiving the command, the bots all together deploy the attack in question, disrupting services.

1.2 Detection Model

1.2.1 Anomaly Detection

Anomaly detection is the process of training a model to baseline threshold with a certain type of values so that it is able to detect any other deviating value as an anomaly. Autoencoders is an unsupervised deep learning technique that is trained to reconstruct its output after compression of the input features and performs anomaly detection to identify anomalous inputs.

1.2.2 Multi-Output Classification

Multi-Output Classification is a type of machine learning that predicts multiple outputs simultaneously. Deep Neural Network is a supervised deep learning model used for classification. A typical DNN consists of input layer, hidden layers and a output layer and predicts one class at a time whereas a Multi-Output DNN consists of several output layers in order to predict multiple classes simultaneously.

1.3 Dataset Generation

The Model is to be tested on a dataset that is simulated. The dataset contains the same features as   N-BaIoT and is generated by simulating two types of attacks.

1.3.1 IoT Network Simulation

The Network with IoT devices is simulated on Azure IoT using virtual machines and a third-party software. Nodes were setup and communication between them tested.

1.3.2 Attack Simulation

Two types of attacks were simulated on the network and run for a set amount of time.

1.3.3 Network Data Capture

The Network data was captured as the attacks were underway, using Azure Network Watcher. The network data was exported as a PCAP file.

1.3.4 Feature Extraction

PCAP file is converted into a csv after adding all the required base features.

A Python3 script was written to extract all 115 features from the base csv and make the final dataset.

1.3.5 File Labelling

The final two files are manually labelled as benigndata.csv and tcpsynattackdata.csv to identify them as benign and malicious datasets.

CHAPTER 2

PROBLEM STATEMENT

With the emerging threats that arise as a result of botnets and the DDoS-as-a-Service Industry, there is a significant need to combat these threats and set a new scheme into implementation whereby newer botnets can be detected and the mitigation process can commence. New botnets are emerging everyday, and the current schemes aim at detecting and mitigating known botnet families, leaving a wide gap for these newer unknown botnet families to take over and cause mass damage to all sorts of devices, leading to a corruption of the network as a whole. This is a severe problem as, in today's world, the internet and the network that connects us all has become immensely valuable and furthermore, disruptions to the network can cause leakage of sensitive information, loss of communication and hampering the work of several important industries.

Another important fact to consider, with regard to botnets, is the frequent targeting of IoT devices. This targeting is due to the tendency of IoT devices to focus on maximizing functionality whilst reducing resource utilization. Due to this, it becomes incredibly difficult for IoT device designers and programmers to include complex cryptographic mechanisms and security measures in the framework of these devices. These reasons make IoT devices an easy target for botnets.

It is due to the reasons listed above that "Anomaly Detection and Classification of IoT Attacks using Autoencoders and Multi-Output DNN" was chosen as the problem statement. Furthermore, it was concluded that making use of Advanced DL Techniques would be beneficial for the project, as these are newer fields of learning and contain more scope for discovering newer methods of detection. This was especially due to the fact that, anomaly detection was selected, which would enable the discovery of newer, unknown botnets and DL techniques would help facilitate this. Furthermore, the project was implemented with the goal of multi-class classification, as opposed to merely focusing on binary classification. Binary Classification involves, classifying incoming botnets into malicious or benign classes, post which they are classified based on the kind of botnet family they belong to, whereas during multi-class classification, they are also classified based on mode of attack of botnets. Thus, both methods were incorporated into the project, in order to provide an extensive array of results regarding the incoming network traffic within an IoT network. Along with this, in order to test the anomaly detection capabilities of the project, an unknown attack dataset was generated.

CHAPTER 3

LITERATURE REVIEW

3.1. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection [1]

3.1.1. Introduction

The paper describes an Intrusion Detection System that flags off abnormal network traffic. It implemented a plug and play Network IDS that ran on a local network.

3.1.2. Characteristics and Implementation

The core algorithm used is an autoencoders. But before that a feature extraction framework is used.

The system was implemented using Raspberry Pi.

Kitsune’s feature extraction framework had multiple steps that helped extract 115 features.

3.1.3. Features

The Kitsune Framework used in the model helps extract 115 features, using the 5 decay windows just like N-BaIoT. Inspired by this approach, we have designed a script to calculate 115 features to match the chosen dataset.

3.1.4. Evaluation

The approach used in the paper is a powerful Network IDS for simple networks and devices.

3.2. IoT Botnet Anomaly Detection Using Unsupervised Deep Learning [2]

3.2.1. Introduction

The paper proposed unsupervised deep learning using anomaly detection for IoT botnet activity in a network.

3.2.2. Characteristics and Implementation

The approach used involved the use of a subset of an already available dataset called Bot-IoT.

The autoencoder model was trained on the benign traffic data after separation. Removing unnecessary features led to the input to being only 16 for the autoencoder.

Next a predictor was used to compare the reconstruction loss from the autoencoders to predict anomalies.

3.2.3. Features

Autoencoder is used to perform anomaly detection.

The predictor uses a high threshold value so that for frequent flags are observed than missing attacks with a low threshold value.

3.2.4. Evaluation

Setting a higher threshold value is a suboptimal optimization as even though it eliminates the risk of missing attacks it also runs the risk of reporting false positives.

3.3 N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders [3]

3.3.1. Introduction

The paper proposes a novel network-based anomaly detection method for the IoT called N-BaIoT that extracts behaviour snapshots of the network and uses deep autoencoders to detect anomalous network traffic from compromised IoT devices.

3.3.2. Characteristics and Implementation

Previous studies relied on emulated or simulated data. But they perform empirical evaluation with real traffic data, gathered from nine commercial IoT devices infected by authentic botnets from two families- Mirai and Bashlite. They capture the raw network traffic data using port mirroring on the switch.

3.3.3. Features

N-BaIoT is a network-based approach. First, they extract statistical features that capture behavioural snapshots of benign IoT traffic, and train a deep autoencoder (one for each device) to learn the IoT device’s normal behaviours. The autoencoders attempt to compress snapshots. When an autoencoder fails to reconstruct a snapshot, it is a strong indication that the observed behaviour is anomalous.

3.3.4. Evaluation

Using incremental statistics to perform the feature extraction, the training of the autoencoders can be performed in a semi-online manner. The training is practical, and there is no storage concern. Their method required only 174 ± 212 ms to detect the attacks.

- Raised the fewest false alarms. It demonstrated a mean FPR (false positive rate) of 0.007 ± 0.01.
- TPR(true positive rate) of 100 percent

3.4 Network Flow based IoT Botnet Attack Detection using Deep Learning [4]

3.4.1. Introduction

The objective of the paper is to provide advanced cyber security solutions to IoT devices and smart city applications. The paper proposes a deep learning-based botnet detection system.

3.4.2. Characteristics and Implementation

Using T-SNE visualization on N-BaIoT dataset, they find the reason why ml algorithms give very high performance on the dataset is because even though the data is non-linear, the data distributions are simple and almost separable. So, the existing anomaly detection based methods may not perform well in the real-world scenario as the pattern of attacks changes with time due to the changes in behaviours of the botnet attack. Thus, to avoid this issue, it has been modelled as intrusion detection in the work.

3.4.3. Features

They collect the network traffic flows from the connected devices in the packet capture (PCAP) format and convert them into connection records. During training, these connection records are labelled manually. ML algorithms and DNNs are used. The framework initially classifies the connection record into either legitimate or attack. Then, it classifies the attacks into their corresponding attack categories and as well as to corresponding botnet families.

3.4.4. Evaluation

The DT (decision tree) classifier showed the best overall performance. ML classifiers have achieved zero false positive rates in the first two experiments and DNN achieved the best performance with zero false positive rates in the 3rd experiment. So, both the DT and DNN models can be used for botnet detection and can be deployed in real-time IoT systems.

- Accuracy - DT: 98.5, DNN: 100
- Precision, recall, F-1 score - DT: 98.6, 98.5, 98.5, DNN: all 100
- Training time(DNN in secs) - 412, 111.150, testing time- 1.310

3.5 Unsupervised Anomaly Based Botnet Detection in IoT Networks [5]

3.5.1. Introduction

Three measurement approaches which require considerably less computational complexity are used for selecting the candidate feature sets. The first approach identifies the features with higher entropy. The second one selects the features demonstrating higher variance as they can properly identify isolation boundaries. The third one uses Hopkins statistics that measures the similarity of the corresponding feature values with the uniform distribution.

3.5.2. Characteristics and Implementation

The data with chosen feature sets are then evaluated with using the three most popular techniques for anomaly detection: local outlier factor (LOF), one class SVM and isolation forest (IF). Corresponding average accuracy scores and recall score values are computed by 10-fold cross-validation procedure

3.5.3. Features

Entropy and isolation forests give relatively higher results in both unbalanced and balanced datasets whereas entropy and SVM achieved better results only in the unbalanced dataset. The best accuracy and precision values, which are above 90%, are obtained by entropy and isolation forests with five features. Additionally, variance and isolation forests with ten features provide a similar result. As a feature selection method, entropy, and as an unsupervised learning method, isolation forests are superior to the others. The host-based features belonging to Host IP and Host MAC&IP categories constitute the most discriminating features (except one channel-based feature).

3.5.4. Evaluation

The main result of the study is that achieving low computational complexity by reducing the feature set is possible for an IoT botnet detection system in which the learning model is trained by only normal traffic. It is shown that such a trained model provides a reasonable accuracy and precision results.

3.6 A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks [6]

3.6.1. Introduction

Proposes a two-level anomalous activity detection system for IoT networks.

3.6.2. Characteristics and Implementation

The level-1 model classifies network traffic as normal or abnormal. If the level-1 model detects the flow as an anomaly, then the flow will be forwarded to the level-2 model for further classification to find the category or subcategory of the detected anomaly.

3.6.3. Features

The level-1 model is trained for binary classification. The proposed model uses a decision tree classifier at level-1 to classify the network traffic into normal traffic or anomalous traffic
The level-2 model is trained to identify the category or subcategory of malicious flow. Random forest classifier is used at level-2 to categorize the network flow received from the level-1 model.

3.6.4. Evaluation

The proposed level-1 model achieved 100% Accuracy, Precision, Recall, and F-1 score for the IoT Botnet Dataset.
Accuracy, Precision, Recall, and F-1 score for normal network traffic measured as 100%, DoS and theft as 99.90, DDoS as 99.60, and reconnaissance as 99.99
An average Accuracy, Precision, Recall, and F-1 score value of 99.80% was achieved for the category and 99% for the subcategory label.

CHAPTER 4

PROJECT REQUIREMENT SPECIFICATION

4.1 Product Perspective

The methodology used is a DL-based IoT botnet attack detection methodology which can be included in an intrusion detection system. It can be used in any software that is deployed in IoT networks to detect threats in the network.

The stated methodology can also be used by IoT device manufacturers as a part of their security features.

4.1.1 Product Features

1. Deployment in an IoT network -
   User can deploy the technique provided as part of an intrusion detection system to analyse and detect threats in an IoT network
2. Gather the data from the network -
   Analyse and collect information from the packets flowing in the network
3. Identify the malicious attributes using the anomaly detection-based methodology devised-
   This enables the detection of newer botnets
4. Detection of botnet attack -
   Alert the IoT device of the detection of a botnet attack

4.1.2 User Classes and Characteristics

The direct users of the product are:

- <u>Network Intrusion Detection and Prevention Systems and other software deployed in an IoT network to detect anomalies</u>
  Network IDS can use this methodology as a part of their system to detect newer botnet attacks and the respective attack types. They will need to identify the

required conditions in which to use the methodology based on whether they are deployed in an IoT network.

- <u>Antivirus software</u>

  Can be used by antivirus software employed by individual host IoT devices to also detect newer botnet attacks and the respective attack types.

The indirect users of the product are:

- <u>IoT device users</u>

IoT devices tend to be vulnerable as they lack the necessary built-in security tools to defend against threats. Since IoT devices are made to be compact and specific in use, it is not feasible to include detection of threats as part of the system. Hence the IoT devices and their users become indirect users of the methodology since it is deployed in the network and not the device itself. They will receive alerts on the detection of a botnet attack. They may not use the product as often as the other users.

- <u>Network administrators and other security professionals</u>

Network administrators and other security professionals are alerted on detecting a botnet attack and can then mitigate based on the classified attack types. They will use the product most often since they oversee protected IoT networks with sensitive information.

4.2 Operating Environment

This is security software that needs to be deployed in a network of IoT connected devices, hence the operating environment is a network used with the software and hardware components of existing security software.

4.3 General Constraints, Assumptions and Dependencies

- The project works on botnet attack data and thus an assumption is made that the network traffic in which it is deployed contains normal and botnet attack traffic

- Another assumption is that the botnet attack that has been simulated is like the attacks performed by actual known botnets
- The existing N-BaIoT dataset has a constraint that it has easily separable classes but combining the generated dataset with it resolves this
- An assumption is made that all IoT devices perform the same, if not similarly, on various wireless networks and that performance is not affected by change in networks for IoT devices.
- Another constraint is the scalability of a detection system is the decision of which layer of the network the system can be placed in. This plays a pivotal role in the type of detection that can be implemented. Based on the scalability of the model we can decide how big a network can be secured. A centralized structure system cannot stretch across a large network but is easier to implement. Whereas a partitioned system can be scaled to span a large network but is more expensive to implement.
- The main dependency that most botnet detection systems today rely on is network flow analysis (91% of systems).

4.4 Software Requirements

4.4.1 Detection

- Python
- TensorFlow for Model Building and evaluation
- Pandas for Data Pre-processing and Integration
- SkLearn for Model building and evaluation
- NumPy for EDA and Data Pre-Processing
- Matplotlib for EDA
- OpenDatasets (N-BaIoT dataset)
- Google Colab with GPU

4.4.2 Attack Dataset Generation

- Azure Cloud for Network Simulation

- BreakingPoint Cloud for TCP SYN Flood attack
- Wireshark for packet analysis.
- Python with Jupyter Notebook for the Data Wrangling

4.5 Non-Functional Requirements

4.5.1 Performance Requirements

- The approach consists of two models - Autoencoder and Multi-Output DNN which were built using the 32 GB RAM and GPU available on Google Colab
- The models were hyperparameter tuned by trial-and-error approach in order to improve the performance
- The combined dataset contains 1018298 records which can further impact the performance
- For the simulation of TCP SYN Flood attack, 3.75 GB of flooding data was required to attack the target device.

4.5.2 Security Requirements

- Proposed botnet detection method has high accuracy and low false negative rate.
- Furthermore, it can safeguard against newer and unknown botnet attacks
- The Supervised classification of attacks will help network professionals mitigate appropriately

CHAPTER 5

HIGH LEVEL DESIGN

5.1 Design Considerations

5.1.1 Design Goals

The significant difference between the newly proposed system and existing systems is the identification of unknown botnet attacks using Anomaly Detection with Deep Learning Techniques. As opposed to existing systems, wherein the main aim was to classify incoming botnet devices into known botnet families, the newly proposed system aims at identifying unknown botnet attacks as well as classifying the known malicious data into botnet and attack types.

Another significant novelty in the newly proposed system is the ability to perform binary (benign/malicious), botnet type and attack type classification simultaneously using Multi-Output Classification with DNN. This has not been explored in existing research papers.

5.1.2 Architecture Choices

- **Data Layer:** The project's architecture consists of the 3 Layers as shown in Fig 5.1. The Data Layer consists of Dataset curation and Data Preprocessing. The first consideration for the architecture was that the Data Layer and Logical Layer be in a single layer. Later it was deliberated that separating the Data Layer from the Logical Layer makes for an independent Data System so that new datasets that need to be tested can be integrated into the architecture without many changes to the models. The Data Layer needs to be dynamic in the sense that as and when a new network data needs to be scanned for malicious attacks, it can be simply replaced with the existing one. The advantage of this system is that testing each component in its performance is easier as there is no dependency.

- **Logic Layer:** The logic layer consists of deep learning models that consist of autoencoders and deep neural networks (DNN). First, an autoencoder is trained with all the known data (benign, Mirai and Bashlite attack data from N-BaIoT and benign from generated dataset). This way malicious records from the generated dataset are detected as anomalies. The known portion of the output from autoencoder is fed into the Multi-output DNN model that goes on to produce the three outputs. The model classifies if a particular record is benign/malicious, benign/Mirai/Bashlite, and benign/tcp/udp/udpplain/syn/junk/combo/scan/ack.
- **Presentation Layer:** The Presentation Layer has the ability to move control back to the Data Layer to enable users (network admins) to input their own datasets (with compatible attributes) to the model for detection. Thus, a level of abstraction is offered, as users don't directly interact with the Data Layer.

5.1.3 Constraints, Assumptions and Dependencies

- Interoperability requirements

  The dataset input given by the user, should be compatible with the N-BaIoT and unknown attack dataset and have all 115 attributes.
- Availability of Resources.

  Datasets like N-BaIoT which are used for training are available as open-source datasets.
- An assumption that the project makes is that different IoT devices don't have different behavior in different networks. The effect of different IoT devices is ignored.
- The network for attack dataset generation is a simulated one and not a real-world network as computational resources to process real network traffic is too high.

5.2 High Level System Design

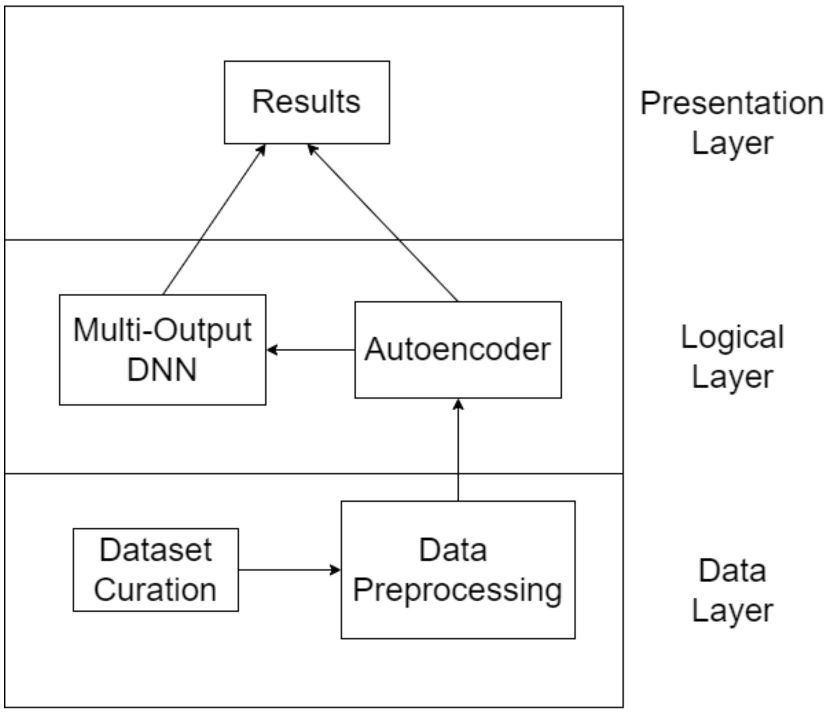


Fig 5.1 Layered Architecture Diagram

5.3 Design Description

5.3.1 Data Flow Diagram

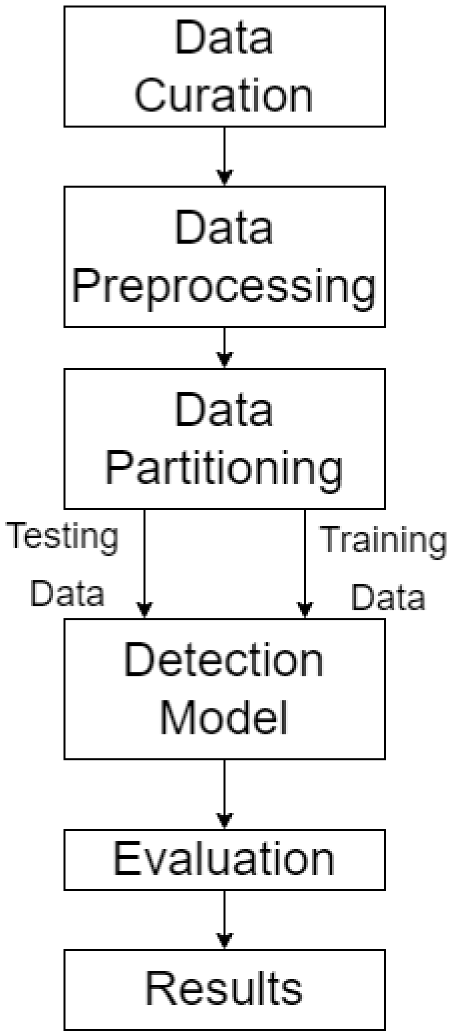


Fig 5.2 Data Flow Diagram

The data flow of the project starts from data curation of the unknown botnet attack dataset. This is followed by Data Pre-processing and Data Partitioning for the model building. The training and testing data is sent to the Detection model consisting of Autoencoder and DNN. The prediction data is then evaluated on metrics like accuracy and F1 score. This has been shown in Fig 5.2 and the same has been elaborated further in Fig 5.3.

5.4 Swimlane Diagram

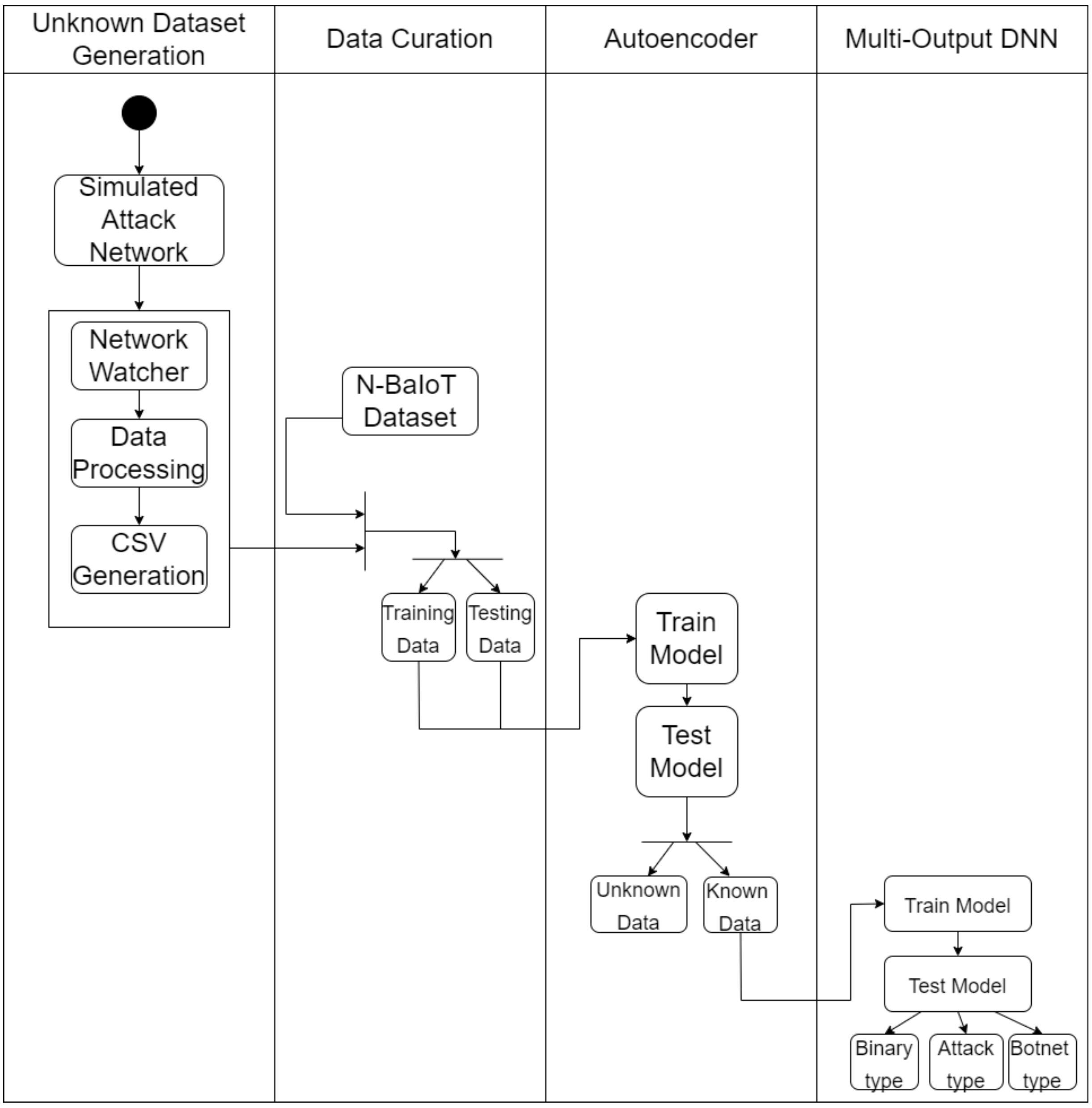


Fig 5.3. Swimlane Diagram

5.5 Design Details

5.5.1 Novelty

Existing multi-class classification approaches classify whether the device is infected or not and further classify the botnet family and the attack category. These approaches classify the botnet families according to the training dataset's botnet families so when a new botnet that is not part of the training dataset appears it will not get identified. In order to resolve this, we propose adding another label called unknown botnet attack. So, when a particular botnet family doesn't fall under normal behaviour or any of the other training dataset botnet family's behaviour, that botnet will be classified as unknown. It is done using anomaly detection.

Moreover, existing approaches have not used Multi-Output Classification to further classify the detected known data. So, instead of performing each of the classifications one by one, we use a Multi-output DNN to classify binary, botnet, and attack type simultaneously.

5.5.2 Performance

The performance is measured using the general model metrics like accuracy, recall, precision, TPR, FPR and F-1 score.

5.5.3 Reliability

Proposed botnet detection method will have a high accuracy and low false negative rate. It makes the model highly reliable.

5.5.4 Application Compatibility

The data provided by the user (network administrator) should be compatible with the model i.e., the features should be like the dataset on which the model was trained.

CHAPTER 6

LOW LEVEL DESIGN

6.1 Low Level Design Diagram

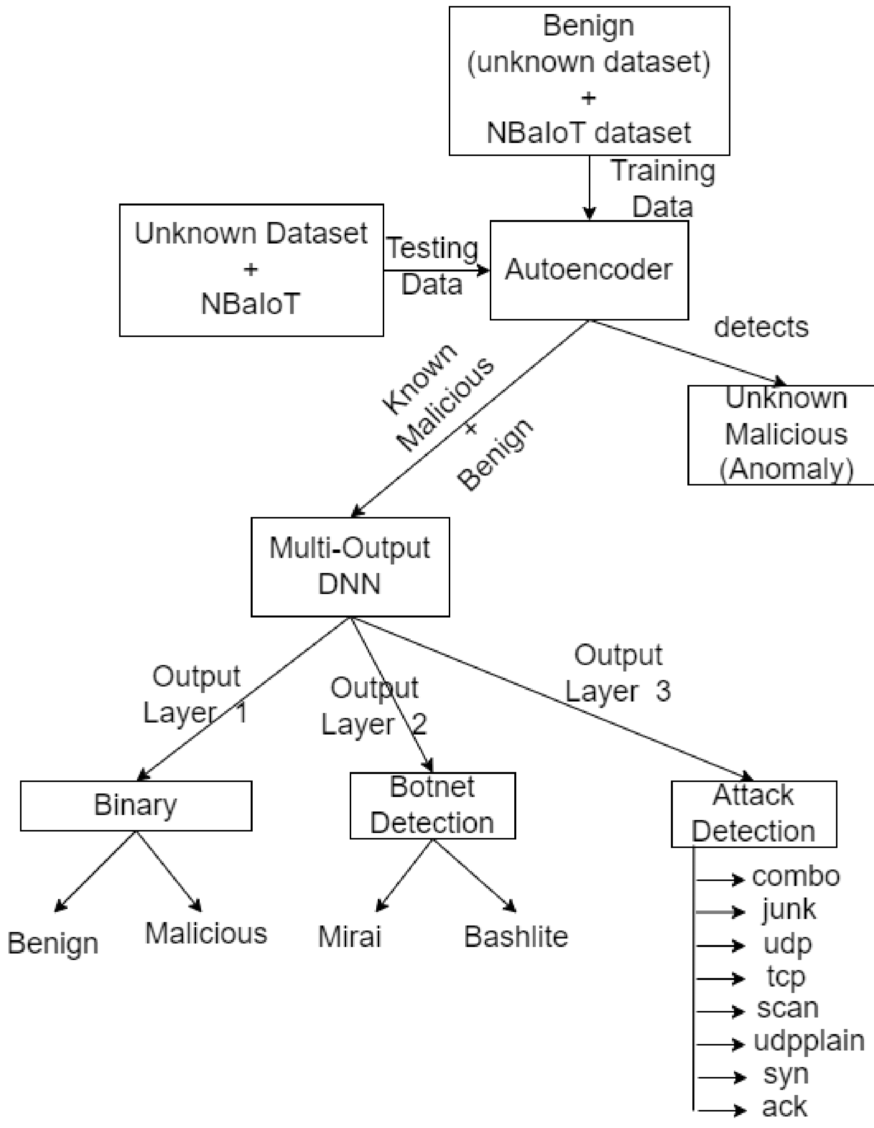


Fig 6. Low Level Design Diagram

CHAPTER 7

SYSTEM DESIGN (DETAILED)

7.1 Attack Dataset Generation

A VNet was set up using Microsoft Azure Cloud. A Target VM was deployed within the VNet and a TCP SYN Flood Attack was generated with the help of Breaking Point Cloud. The VNet was made accessible using public IPs. Azure Network Watcher was used to capture the packets sent during the attack. The packet capture was set up to monitor the target device and store the traffic data in a .cap file. In order to make the captured packet data compatible with the training dataset, the .cap file was converted to a .csv file and a python script was run. The script performs aggregations and wrangling to derive all 115 features from the training dataset. The new dataset was then given as input to the detection model for further testing purposes.

7.2 Detection

7.2.1 Autoencoder

The architecture of autoencoder consists of five encoder layers and four decoder layers. 115 features are given as input to the encoder layer, which compresses the features at each level and learns the underlying pattern, then decompresses to give back 115 features at the output layer. It helps the model detect anomalies in the testing phase. The model is trained on N-BaIoT and generated benign data. During testing, both the datasets (N-BaIoT and the generated dataset) are combined and used to detect anomalies (unknown malicious). The rest of the data is separated from anomaly, which is fed into the multi-output DNN.

7.2.2 Multi-Output Deep Neural Network

According to previous research done, existing papers have not explored multi-output classification with the dataset. The Multi-Output DNN consists of 1 input layer, 4 hidden layers and 3 output layers for binary type, botnet type and attack type classification respectively. The DNN model does simultaneous classification of binary type (benign or malicious), botnet type (benign or mirai or bashlite), and attack type (benign or one of the known eight attacks). The system design diagram is shown in Fig 7.3

7.3 System Design Diagram

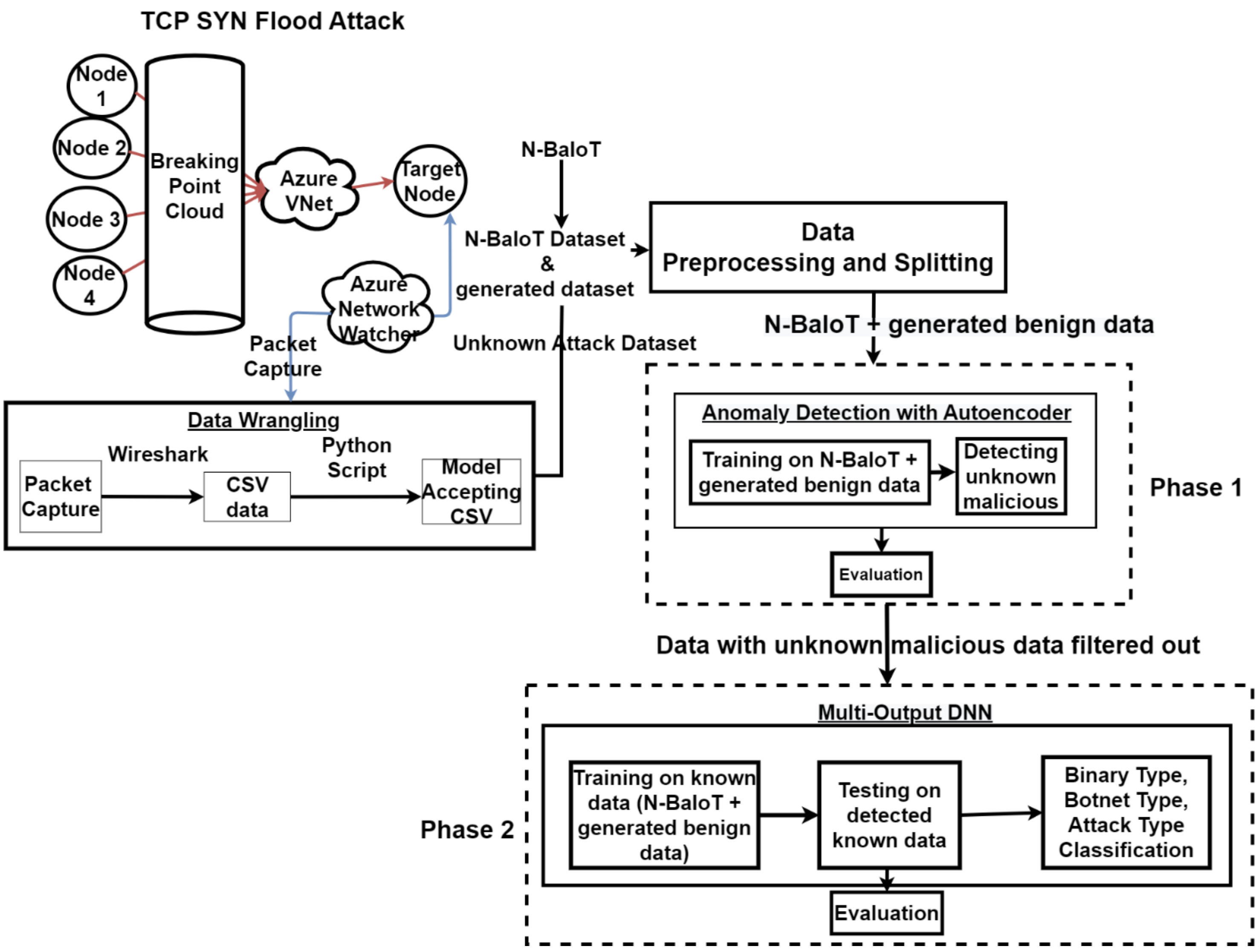

TCP SYN Flood Attack
Node 1
Node 2
Node 3
Node 4
Breaking Point Cloud
Azure VNet
Target Node
Azure Network Watcher
Packet Capture
N-BaloT
N-BaloT Dataset & generated dataset
Unknown Attack Dataset
Data Preprocessing and Splitting
N-BaloT + generated benign data
Data Wrangling
Wireshark
Python Script
Packet Capture
CSV data
Model Accepting CSV
Anomaly Detection with Autoencoder
Training on N-BaloT + generated benign data
Detecting unknown malicious
Evaluation
Phase 1
Data with unknown malicious data filtered out
Multi-Output DNN
Phase 2
Training on known data (N-BaloT + generated benign data)
Testing on detected known data
Binary Type, Botnet Type, Attack Type Classification
Evaluation

Fig 7.3 System Design Diagram

CHAPTER 8

PROPOSED METHODOLOGY

8.1 Unknown Attack Dataset Generation

The proposed methodology includes the simulation of an unknown attack and the generation of a dataset encapsulating the same.

8.1.1 Attack Simulation

The simulated attack type chosen was a DDoS attack, in order to pose similarities to that of a botnet attack. The attack was to be simulated on a target device, the packets of which would be used to generate the new dataset. The attack and environment were proposed to be simulated using a cloud-based platform. It was decided, to generate all 115 features in the N-BaIoT dataset, without using physical infrastructure. The packets were to be captured in .cap format and converted to .csv for further processing.

8.1.2 Dataset Generation

The dataset generated was made to be compatible with the N-BaIoT dataset. This is because the model constructed to perform detection was trained using the dataset. The dataset consisted of 115 features and all 115 features were to be generated for the new dataset. It would be implemented using data processing and wrangling techniques, along with various statistics for aggregation of the existing features.

8.2 Detection Model

The next part of the proposed methodology involves anomaly detection, accompanied by multi-output classification of known malware and benign records.

8.2.1 Anomaly Detection

An anomaly detection model was constructed to detect unknown malicious records. It is done using an autoencoder. The model would be capable of detecting unknown or newer botnet attacks, thus, safeguards against emerging attacks. These abnormalities were to be decided, based on a threshold. The threshold would be set based on the training data, consisting of both benign and known malware. Records going beyond the threshold would be detected as anomalies. This would imply that these records were neither benign, nor known malware. The anomalies detected would then be singled out and flagged.

8.2.2 Multi-Output Classification

A Multi-Output Classification model would be constructed in order to further classify the known data detected by the anomaly detection model as known benign/known malware. Multi-Output Classification is a type of machine learning that predicts multiple outputs simultaneously. This saves the time taken for classification and is therefore efficient. A typical DNN consists of input layer, hidden layers and a output layer and predicts one class at a time whereas a Multi-Output DNN consists of several output layers in order to predict multiple classes simultaneously. This would enable, not only detection of unknown malware, but also classification of known benign and malicious data. The known malware detected would further be classified into benign/malicious, the type of botnet they come under, along with the mode of attack taken by them to infiltrate the target device. After classification, these devices would be flagged based on their type of botnet and mode of attack.

CHAPTER 9

IMPLEMENTATION AND PSUEDOCODE

9.1 Unknown Attack Dataset Generation

9.1.1 Simulated Network Setup

The simulation environment used to generate the attack was Microsoft Azure Cloud.

The attack was performed within an Azure VNet (Virtual Network) with public IP addresses attached to its components. The components of the network included a target VM which would be the victim of the attack performed, a public Network Interface Card to navigate incoming traffic to the victim, a Network Security Group to manage the DDoS Protection Standards Azure has in place and a public IP address to make the target accessible to incoming traffic. The file was then wrangled and transformed into the resultant dataset to test the anomaly detection aspect of the model. The network topology is shown in Fig 9.1.

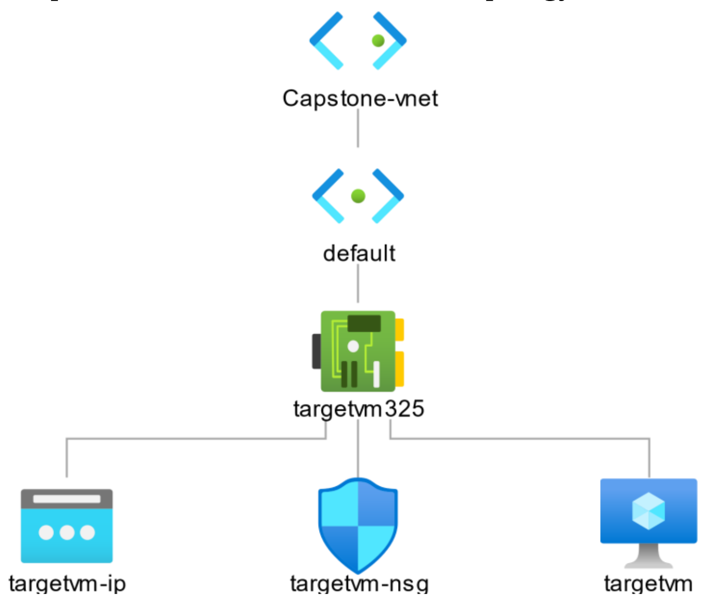
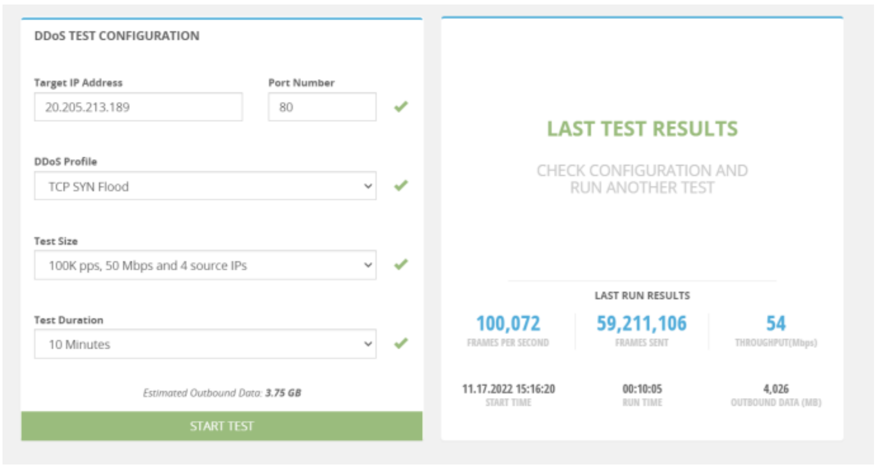


Fig 9.1. Network Topology Diagram

9.1.2 Breaking Point Cloud

Due to the DDoS Protection Standards enforced by Azure, an application known as BreakingPoint Cloud was used to bypass these Standards and perform the attack.

In order to do this, the network was made accessible through public endpoints and IPs. It was linked to the Azure VNet created. Port 53 and the public IP address of the target VM were given to specify the target of the attack. The attack performed was a TCP SYN flood attack, a common mode of attack taken by various botnet families. The attack was performed by flooding the target VM with SYN packets. This would cause the target VM to send ACK packets as a part of the three-way TCP handshake. It was followed by sending further SYN Packets, instead of responding to the previous ACK packet.



DDoS TEST CONFIGURATION

Target IP Address: 20.205.213.189
Port Number: 80 ✓

DDoS Profile: TCP SYN Flood ✓

Test Size: 100K pps, 50 Mbps and 4 source IPs ✓

Test Duration: 10 Minutes ✓

Estimated Outbound Data: ***3.75 GB***

LAST TEST RESULTS

CHECK CONFIGURATION AND RUN ANOTHER TEST

LAST RUN RESULTS

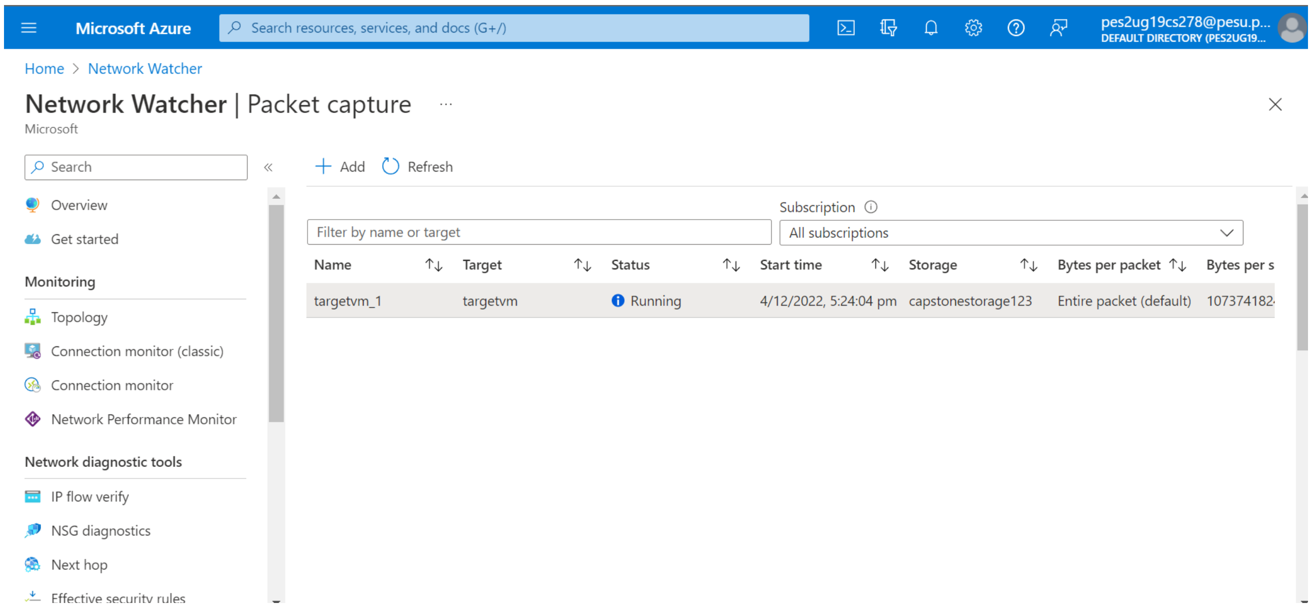
100,072	59,211,106	54
FRAMES PER SECOND	FRAMES SENT	THROUGHPUT(Mbps)
11.17.2022 15:16:20	00:10:05	4,026
START TIME	RUN TIME	OUTBOUND DATA (MB)

Fig 9.2. Breaking Point Cloud

This caused the sessions to stay open until the session timeout occurred, thus rendering the target unable to respond to legitimate traffic. During the attack, 5,92,11,106 frames were sent, summing up to 3.75 GB of data and four attack nodes were used to perform the DDoS attack. The Breaking Point Cloud setup is shown in Fig 9.2.

9.1.3 Packet Capture using Network Watcher

Azure Network Watcher was used to monitor the network. Azure Network Watcher enabled an overall view of the topology and had a feature within it, known as Azure Packet Capture. Azure Packet Capture was used to capture the network traffic in the form of packets. The packets were logged, throughout the duration of the attack, into a .cap file in the cloud storage location. The screenshot of Azure Network Watcher is shown in Fig 9.3.

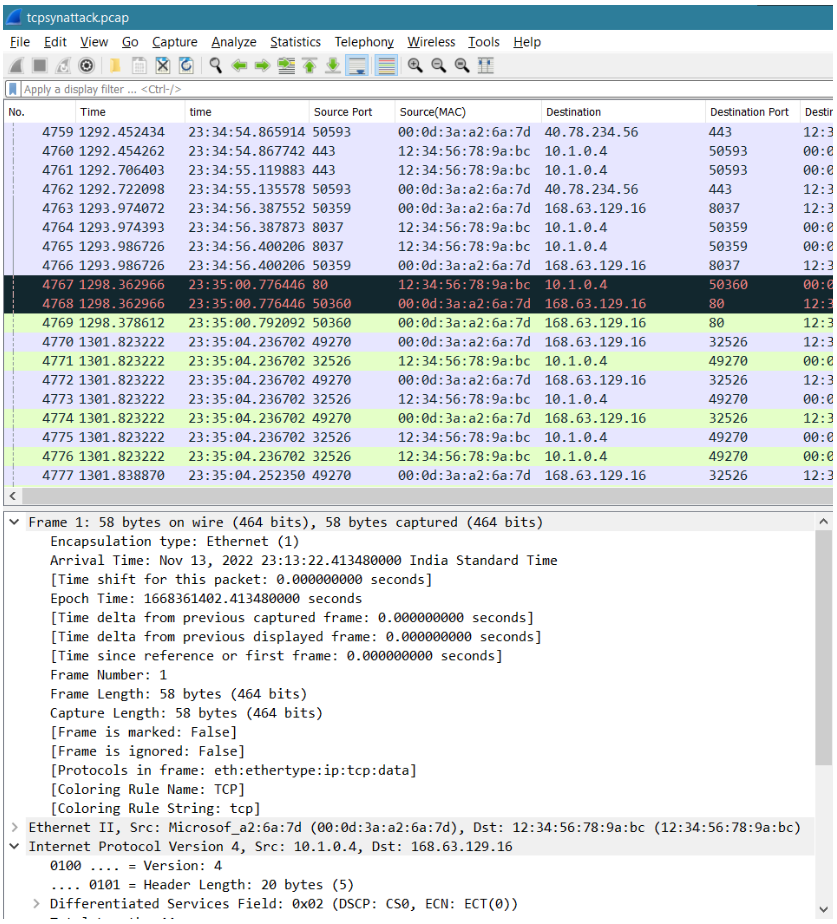


Name	Target	Status	Start time	Storage	Bytes per packet	Bytes per s
targetvm_1	targetvm	Running	4/12/2022, 5:24:04 pm	capstonestorage123	Entire packet (default)	107374182

Fig 9.3. Azure Network Watcher

9.1.4 Packet Analysis using Wireshark

The .cap file was analysed using Wireshark to add various features, relevant to the generation of the dataset. The features added were MAC addresses of the source nodes and destination nodes along with the jitter of the packets. These features were added to the .pcap file which was then exported as a CSV file to be fed into the python script for data wrangling. The wireshark packet analysis tool screenshot is shown in Fig 9.4.


tcpsynattack.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time         time             Source Port Source(MAC)        Destination     Destination Port Destir
4759 1292.452434  23:34:54.865914  50593       00:0d:3a:a2:6a:7d  40.78.234.56    443              12:3
4760 1292.454262  23:34:54.867742  443         12:34:56:78:9a:bc  10.1.0.4        50593            00:0
4761 1292.706403  23:34:55.119883  443         12:34:56:78:9a:bc  10.1.0.4        50593            00:0
4762 1292.722098  23:34:55.135578  50593       00:0d:3a:a2:6a:7d  40.78.234.56    443              12:3
4763 1293.974072  23:34:56.387552  50359       00:0d:3a:a2:6a:7d  168.63.129.16   8037             12:3
4764 1293.974393  23:34:56.387873  8037        12:34:56:78:9a:bc  10.1.0.4        50359            00:0
4765 1293.986726  23:34:56.400206  8037        12:34:56:78:9a:bc  10.1.0.4        50359            00:0
4766 1293.986726  23:34:56.400206  50359       00:0d:3a:a2:6a:7d  168.63.129.16   8037             12:3
4767 1298.362966  23:35:00.776446  80          12:34:56:78:9a:bc  10.1.0.4        50360            00:0
4768 1298.362966  23:35:00.776446  50360       00:0d:3a:a2:6a:7d  168.63.129.16   80               12:3
4769 1298.378612  23:35:00.792092  50360       00:0d:3a:a2:6a:7d  168.63.129.16   80               12:3
4770 1301.823222  23:35:04.236702  49270       00:0d:3a:a2:6a:7d  168.63.129.16   32526            12:3
4771 1301.823222  23:35:04.236702  32526       12:34:56:78:9a:bc  10.1.0.4        49270            00:0
4772 1301.823222  23:35:04.236702  49270       00:0d:3a:a2:6a:7d  168.63.129.16   32526            12:3
4773 1301.823222  23:35:04.236702  32526       12:34:56:78:9a:bc  10.1.0.4        49270            00:0
4774 1301.823222  23:35:04.236702  49270       00:0d:3a:a2:6a:7d  168.63.129.16   32526            12:3
4775 1301.823222  23:35:04.236702  32526       12:34:56:78:9a:bc  10.1.0.4        49270            00:0
4776 1301.823222  23:35:04.236702  32526       12:34:56:78:9a:bc  10.1.0.4        49270            00:0
4777 1301.838870  23:35:04.252350  49270       00:0d:3a:a2:6a:7d  168.63.129.16   32526            12:3
```

```
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 13, 2022 23:13:22.413480000 India Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1668361402.413480000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 58 bytes (464 bits)
    Capture Length: 58 bytes (464 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:data]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
Ethernet II, Src: Microsof_a2:6a:7d (00:0d:3a:a2:6a:7d), Dst: 12:34:56:78:9a:bc (12:34:56:78:9a:bc)
Internet Protocol Version 4, Src: 10.1.0.4, Dst: 168.63.129.16
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x02 (DSCP: CS0, ECN: ECT(0))
```

Fig 9.4. Wireshark Packet Analysis Tool

9.1.5 Data Wrangling using Python Script

The dataset required to have 115 features in total, consisting of, 23 features captured over 5 different time windows with a decay factor, λ (5,3,1,0.1,0.01) (as seen in Eq 9.1.). These 23 features were derived from the existing features using various statistics.

The time windows were calculated using Equation 9.1,

$$d\lambda(t) = 2^{-\lambda t} \qquad (9.1)$$

Therefore, based on the different decay factors, the time windows chosen were 100ms, 500ms, 1.5sec, 10sec and 1min.

The .csv file with the IP addresses(source and destination), MAC addresses(source and destination), packet jitter, length of packet, Port Number (source and destination) is fed into the python script. The script first reads the csv and writes the data into 2D arrays. The arrays are iterated through using the calculated time windows for each of the features mentioned below. The table below (*Table 9.1.*) contains the features of the dataset generated.

Table 9.1. Dataset Features

Feature	**Description**	**Statistic**
MI	Stat summarizing traffic from a specific MAC+IP address	MI_weight
		MI_mean
		MI_variance
H	Stat summarizing traffic from a specific IP address(source)	H_weight
		H_mean
		H_variance
HH	Stat summarizing the traffic from a specific source IP to a destination IP (Channel)	HH_weight
		HH_mean
		HH_std
		HH_mag
		HH_radius
		HH_covariance
		HH_pcc
HH_jit	Stat summarizing jitter from a specific source IP to a destination IP (Channel jitter)	HH_jit_weight
		HH_jit_mean
		HH_jit_std
		HH_jit_mag
		HH_jit_radius
		HH_jit_covariance
		HH_jit_pcc
HpHp	Stat summarizing the traffic from a specific source IP+Port No. to a destination IP+Port No.	HpHp_weight
		HpHp_mean
		HpHp_std
		HpHp_mag
		HpHp_radius
		HpHp_covariance
		HpHp_pcc

The following formulae are used to calculate the features and then write them into arrays, over the different time windows. The table below (*Table 9.2*) contains all the formulae for the features.

Table 9.2. Feature Calculation Formulae

Feature	Formula
Weight	$\boldsymbol{weight(W) = \Sigma X_i}$
Mean	$\boldsymbol{mean(M) = \frac{W}{N}}$
Variance	$\boldsymbol{variance(S^2) = \frac{\Sigma(X_i - M)^2}{N-1}}$
Standard deviance	$\boldsymbol{std(S) = \sqrt{variance}}$
Magnitude	$\boldsymbol{mag = \sqrt{{M_1}^2 + {M_2}^2}}$
Radius	$\boldsymbol{rad = \sqrt{(\frac{\Sigma(X_i - M_1)^2}{n-1})^2 + (\frac{\Sigma(X_i - M_2)^2}{m-1})^2}}$
Covariance	$\boldsymbol{cov = \frac{\Sigma(X_i - M_1).(X_i - M_2)}{N-1}}$
Pearson Correlation Coefficient (pcc)	$\boldsymbol{pcc = \frac{\Sigma(X_i - M_1).(X_i - M_2)}{\sqrt{\Sigma(X_i - M_1)^2 .\Sigma(X_i - M_2)^2}}}$

The arrays for each of the features are then put into one array with the attribute names and the zip() function is used to flip the arrays such that each tuple in the final csv file is an array in the 2D array. Finally, the 2D array is iterated through and written
into the csv file, to be further given as input to the Autoencoders for anomaly detection.

9.2.1 Data Preprocessing

- The generated unknown botnet dataset has been combined with N-BaIoT
- Data pre-processing includes data cleaning, data integration, data labelling, label encoding and normalization.
- Data cleaning is done to remove complex numbers from the unknown botnet dataset
- Data integration is done to combine the data device-wise for N-BaIoT since the dataset has been provided as multiple files and to combine N-BaIoT dataset with the generated dataset.
- The integrated dataset has been labelled with three columns: binary_type for binary classification (benign /malicious); botnet_type for multiclass classification into (benign/ mirai/bashlite/unknown malicious); attack type for multiclass classification (benign/ 8 attacks/unknown malicious) and another label bin_type for known/unknown malicious data.
- The classes are categorically encoded, and the data has been normalized using Min-Max Scaling.
- Exploratory Data Analysis is also performed on N-BaIoT dataset
- During EDA, it was noticed that the N-BaIoT dataset consists of significantly more malicious records and mirai records as shown in Fig 9.5a, Fig 9.5b and Fig 9.5c. Furthermore, the binary and botnet type classes are easily separable as seen in the t-SNE in Fig 9.6. Thus, on combining with the generated dataset, this is resolved.
- After EDA, the dataset is split into training-set (80%) and testing-set (20%).

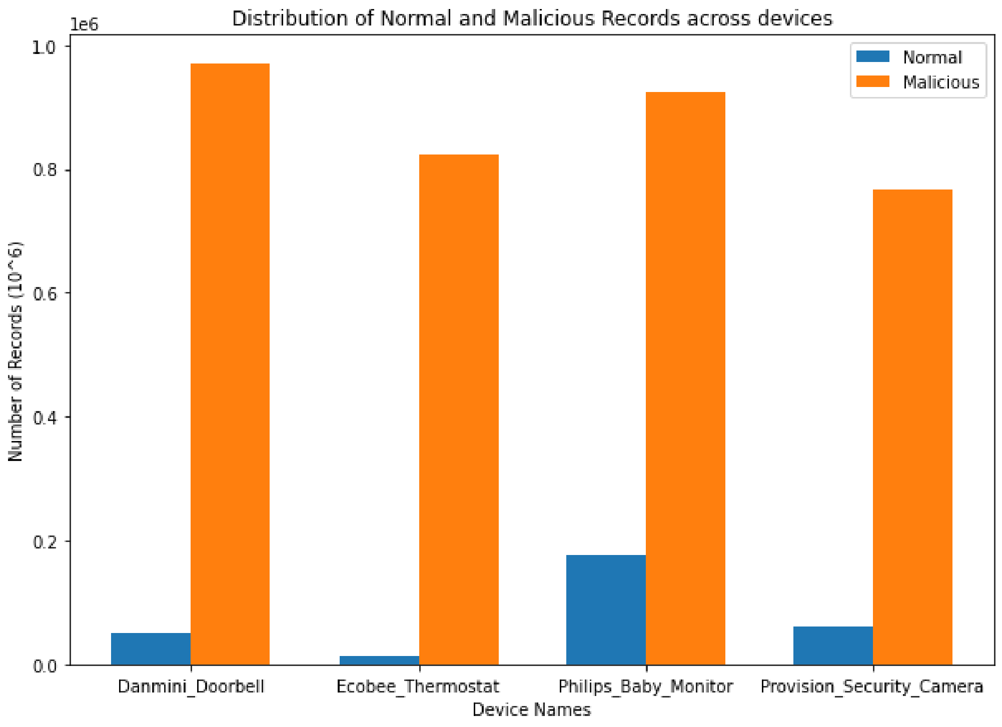


Fig 9.5a. EDA on N-BaIoT dataset – Distribution of Malicious record

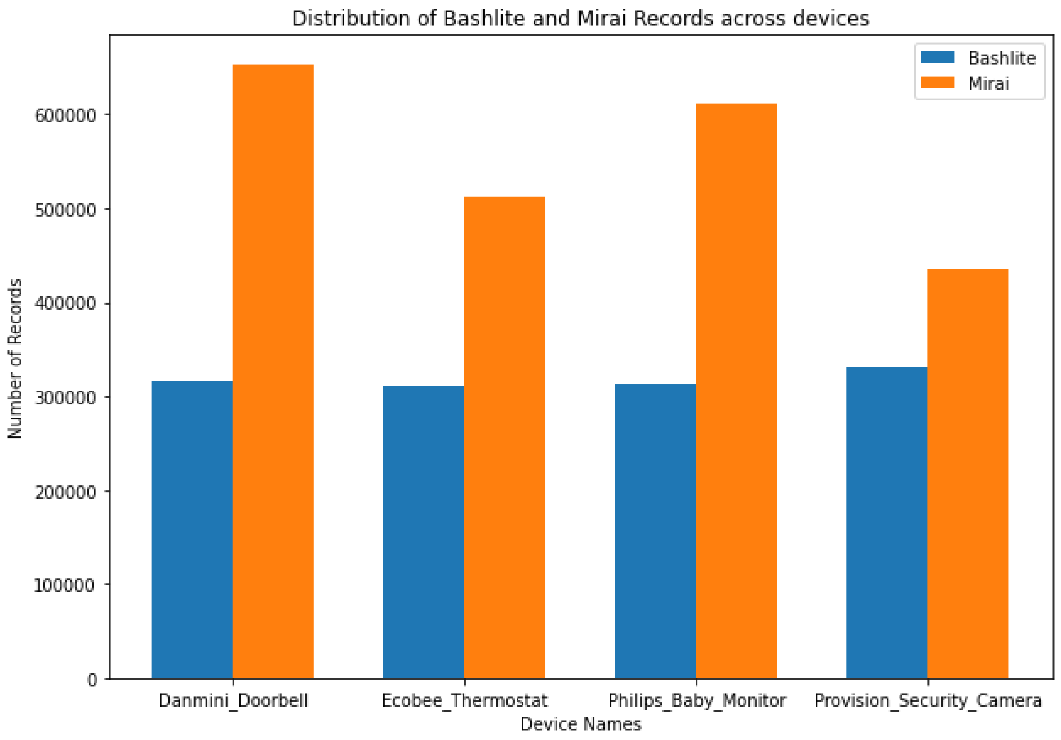


Fig 9.5b. EDA on N-BaIoT dataset – Distribution of Botnet type records

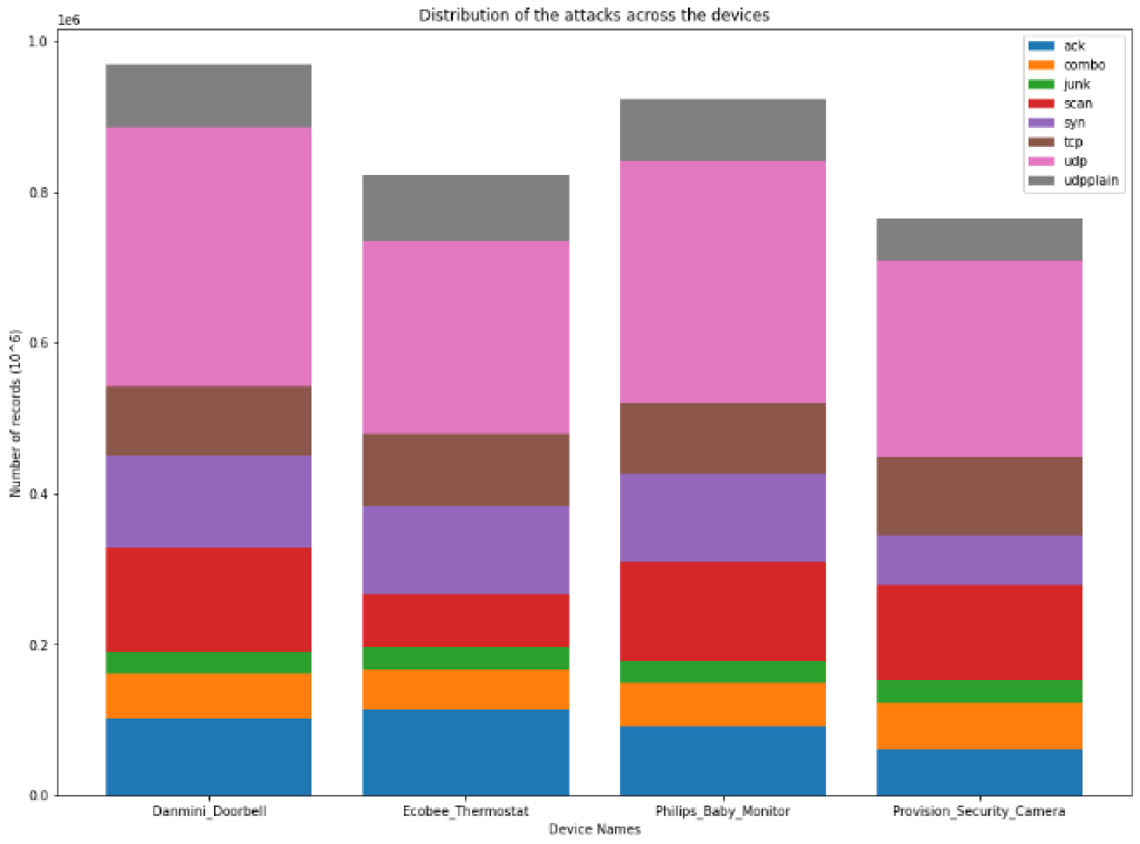


Fig 9.5c. EDA on N-BaIoT dataset – Distribution of Attack Type Records

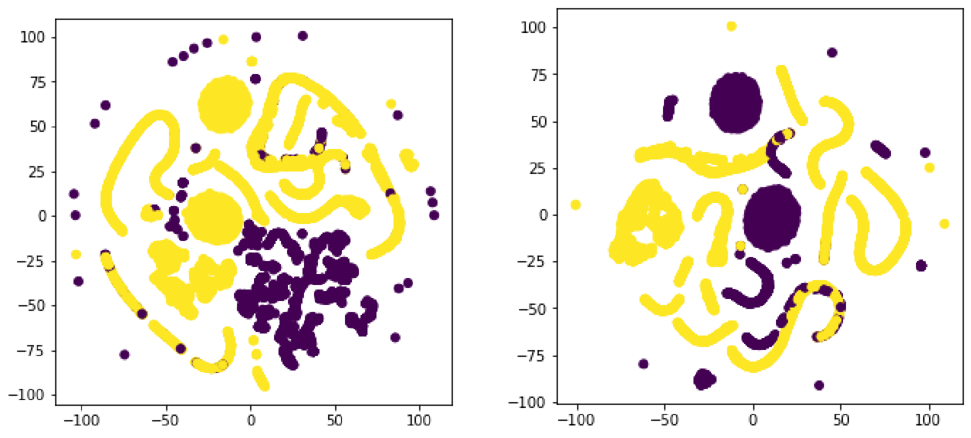


Fig 9.6. TSNE – Left: Normal and malicious, Right: Mirai and Bashlite

9.2.2 Autoencoder

- Part of the novelty of the project comes from detecting unknown botnet attacks using anomaly detection.
- Anomaly detection is achieved by using autoencoders.
- The autoencoder is used to detect unknown botnet attacks using the column with ‘known’ and ‘unknown malicious’ labels
- To do this, it is trained with N-BaIoT and generated benign data. The architecture of the autoencoder is shown in Fig 9.7.
- Testing is done with data containing both benign and malicious records from N-BaIoT and generated dataset. The model summary is shown in Fig 9.8
- It finally detects the generated unknown botnet attack data
- Pseudocode:

```
o Input: Two-dimensional array A consisting of 10,93,070 records and 115 features sent to the encoder, where each value is normalized and scaled in the range(0, 1)
o Fix a number h for the units of neurons at each dense layer ($h \in N, h < 115$), and a number d of hidden layers ($d < 2$ x size of input layer)
o Training: for each feature ai of A, where $i \in [1, 115]$:
    ➢ for each training iteration:
        ▪ for each hidden layer:
            ❖ encoder: compress the data effectively using fixed units so that the encoder learns expected patterns
            ❖ decoder: reconstruct the output from the lower-dimensional data obtained from previous step
o Testing: for each feature ai of A, where $i \in [1, 115]$:
    ➢ for each training iteration:
        ▪ for each d hidden layer:
            ❖ compute losses of individual instances using (Equation 1)
```

- compute threshold for anomaly score dynamically using (Equation 2)
- if loss > threshold, mark the instance as anomaly
- [where Equation 1: loss = mean(square(log(y_true + 1) - log(y_pred + 1)), axis=-1); Equation 2: mean(individual losses) + standard deviation(individual losses)]

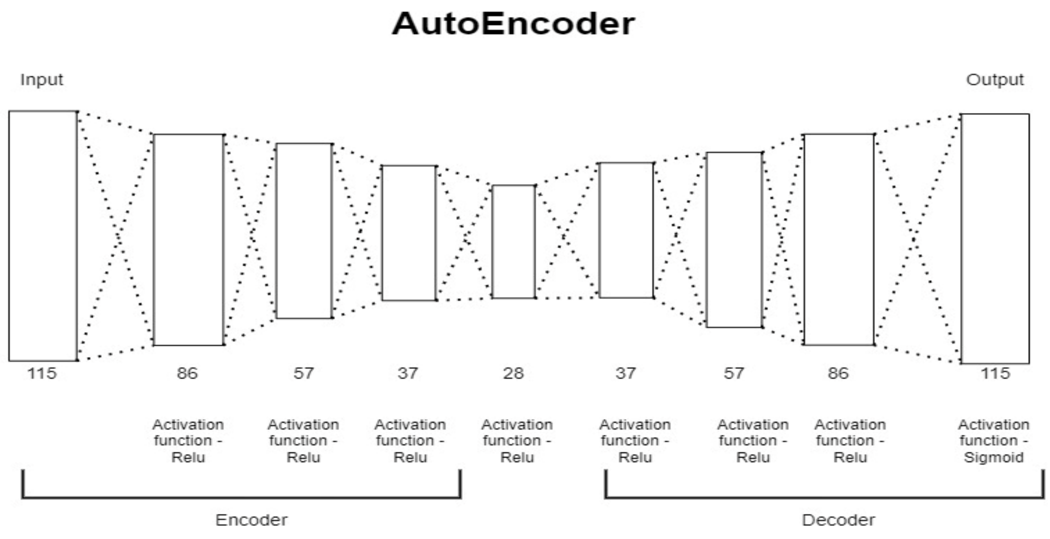
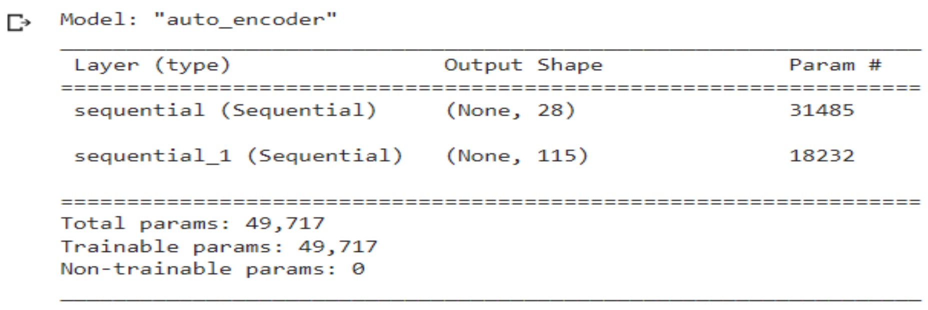

AutoEncoder
Input
Output
115
86
57
37
28
37
57
86
115
Activation function - Relu
Activation function - Relu
Activation function - Relu
Activation function - Relu
Activation function - Relu
Activation function - Relu
Activation function - Relu
Activation function - Sigmoid
Encoder
Decoder

Fig 9.7. Autoencoder Architecture



```
Model: "auto_encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 28)                31485

 sequential_1 (Sequential)   (None, 115)               18232

=================================================================
Total params: 49,717
Trainable params: 49,717
Non-trainable params: 0
_________________________________________________________________
```

Fig 9.8. Autoencoder Details

9.2.3 Multi-Output DNN

- The detected unknown botnet data is removed from the malicious data which results in the known data (N-BaIoT and generated benign data)
- This will be passed into a Multi-output DNN to classify all the three classes (binary type, botnet type, attack type)
- Multi-output classification is a type of machine learning that predicts multiple outputs simultaneously
- To the best of our knowledge, existing papers have not explored multi-output classification with the dataset
- The Multi-Output DNN consists of 1 input layer, 4 hidden layers and 3 output layers for binary type, botnet type and attack type classification respectively. The architecture and summary of the model is shown in Fig 9.9 and Fig 9.10.
- Pseudocode:

```
Input: Pre-processed N-BaIoT dataset is combined with generated benign and unknown botnet data which is split into dataframe_train_1 and dataframe_test_1 which are the training and testing sets respectively. The columns in the data frame are:  binary_type, botnet_type and attack_type, bin_type. The detected unknown botnet attack data is removed using the predicted bin_type column from autoencoder.
1. Train the DNN using dataframe_train_1. The DNN has an input layer that consists of 115 neurons, the four hidden layers consist of 115, 256, 128, 64 neurons. The first output layer uses sigmoid activation function. The second output layer uses softmax activation function and consists of 3 neurons. The third output layer uses softmax activation function and consists of 9 neurons
2. Test the DNN using dataframe_test_1
3. Evaluate using accuracy, precision, f1 score, recall, TPR and FPR
```

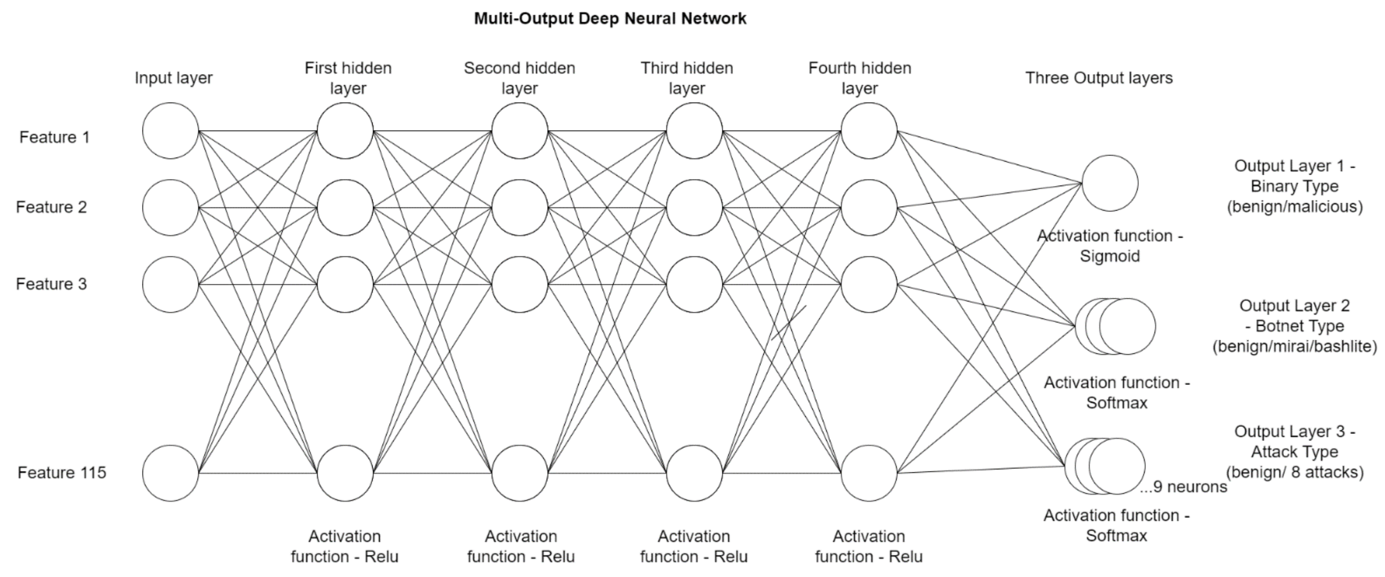
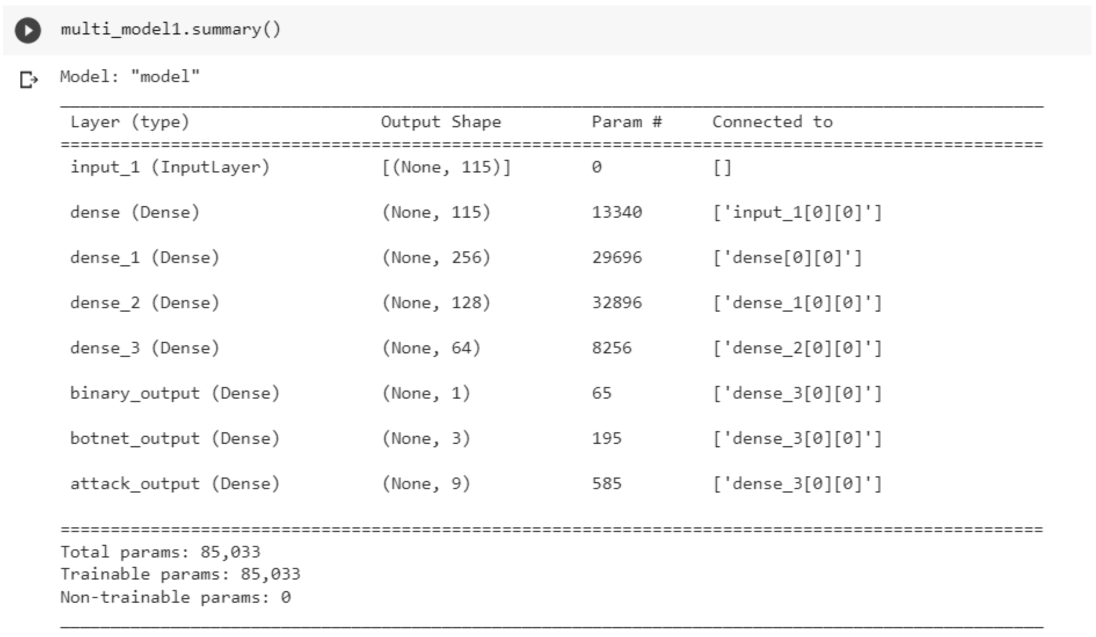

Multi-Output Deep Neural Network
Input layer
First hidden layer
Second hidden layer
Third hidden layer
Fourth hidden layer
Three Output layers
Feature 1
Feature 2
Feature 3
Feature 115
Output Layer 1 - Binary Type (benign/malicious)
Activation function - Sigmoid
Output Layer 2 - Botnet Type (benign/mirai/bashlite)
Activation function - Softmax
Output Layer 3 - Attack Type (benign/ 8 attacks)
...9 neurons
Activation function - Softmax
Activation function - Relu
Activation function - Relu
Activation function - Relu
Activation function - Relu

Fig 9.9. Multi-Output DNN Architecture



```
multi_model1.summary()

Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 115)]        0           []

 dense (Dense)                  (None, 115)          13340       ['input_1[0][0]']

 dense_1 (Dense)                (None, 256)          29696       ['dense[0][0]']

 dense_2 (Dense)                (None, 128)          32896       ['dense_1[0][0]']

 dense_3 (Dense)                (None, 64)           8256        ['dense_2[0][0]']

 binary_output (Dense)          (None, 1)            65          ['dense_3[0][0]']

 botnet_output (Dense)          (None, 3)            195         ['dense_3[0][0]']

 attack_output (Dense)          (None, 9)            585         ['dense_3[0][0]']

==================================================================================================
Total params: 85,033
Trainable params: 85,033
Non-trainable params: 0
__________________________________________________________________________________________________
```

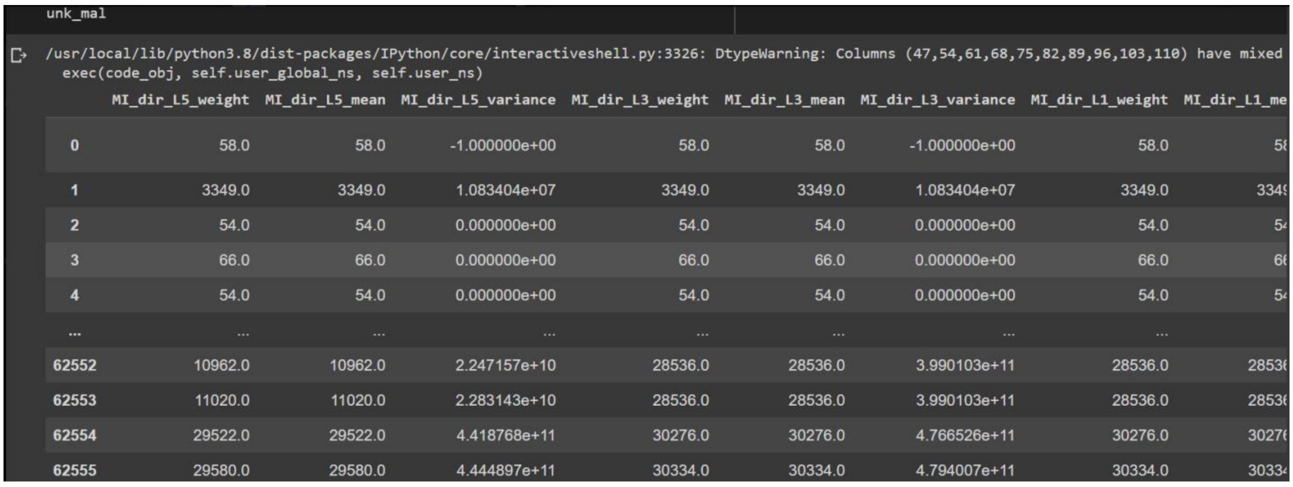
Fig 9.10. Multi-Output DNN Details

CHAPTER 10

RESULTS AND DISCUSSION

10.1 Unknown Attack Dataset Generation

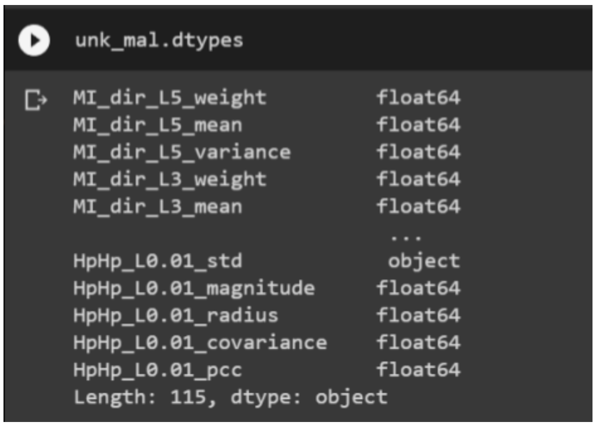
A TCP SYN Flood Attack was successfully performed on a target node within Azure, with a total of 5,92,11,106 frames being sent, summing up to 3.75 GB of data. The data was captured and processed by the python script to generate the required dataset. The dataset generated contained all 115 features, required by the model. The generated dataset and its data types are shown in Fig 10.1.



```
unk_mal
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (47,54,61,68,75,82,89,96,103,110) have mixed
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	MI_dir_L3_variance	MI_dir_L1_weight	MI_dir_L1_me
0	58.0	58.0	-1.000000e+00	58.0	58.0	-1.000000e+00	58.0	58
1	3349.0	3349.0	1.083404e+07	3349.0	3349.0	1.083404e+07	3349.0	334
2	54.0	54.0	0.000000e+00	54.0	54.0	0.000000e+00	54.0	54
3	66.0	66.0	0.000000e+00	66.0	66.0	0.000000e+00	66.0	66
4	54.0	54.0	0.000000e+00	54.0	54.0	0.000000e+00	54.0	54
...	...	...	...	...	...	...	...	
62552	10962.0	10962.0	2.247157e+10	28536.0	28536.0	3.990103e+11	28536.0	2853
62553	11020.0	11020.0	2.283143e+10	28536.0	28536.0	3.990103e+11	28536.0	2853
62554	29522.0	29522.0	4.418768e+11	30276.0	30276.0	4.766526e+11	30276.0	3027
62555	29580.0	29580.0	4.444897e+11	30334.0	30334.0	4.794007e+11	30334.0	3033



```
unk_mal.dtypes
```

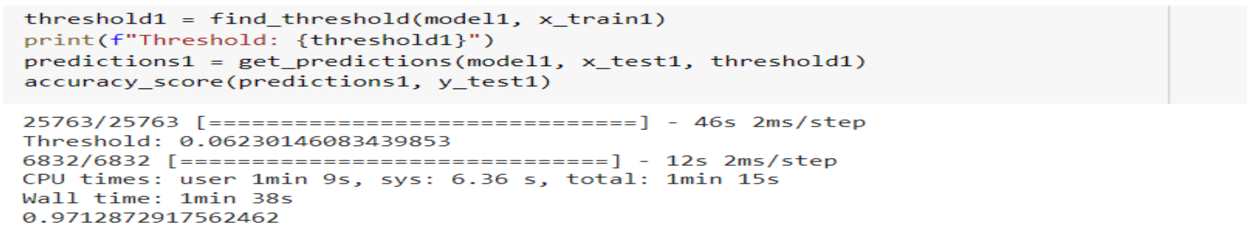
```
MI_dir_L5_weight           float64
MI_dir_L5_mean             float64
MI_dir_L5_variance         float64
MI_dir_L3_weight           float64
MI_dir_L3_mean             float64
                             ...
HpHp_L0.01_std              object
HpHp_L0.01_magnitude       float64
HpHp_L0.01_radius          float64
HpHp_L0.01_covariance      float64
HpHp_L0.01_pcc             float64
Length: 115, dtype: object
```

Fig 10.1. Unknown Attack Dataset

10.2 Detection Model

The Autoencoder performed anomaly detection to detect unknown botnet attacks and the Multi-Output DNN was able to classify the known data into their respective sub classes successfully. For evaluation, both the models are evaluated on metrics such as accuracy, precision, recall, F1-Score and Inference time.

The Autoencoder resulted in a testing accuracy of 97% with a threshold of 0.062.

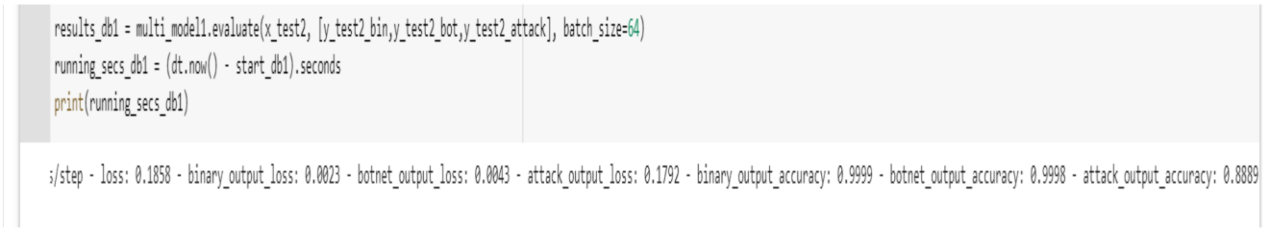


```
threshold1 = find_threshold(model1, x_train1)
print(f"Threshold: {threshold1}")
predictions1 = get_predictions(model1, x_test1, threshold1)
accuracy_score(predictions1, y_test1)
```

```
25763/25763 [==============================] - 46s 2ms/step
Threshold: 0.06230146083439853
6832/6832 [==============================] - 12s 2ms/step
CPU times: user 1min 9s, sys: 6.36 s, total: 1min 15s
Wall time: 1min 38s
0.9712872917562462
```

Fig 10.2. Autoencoder Threshold and Accuracy Output

The Multi-Output DNN resulted in a training accuracy of 99.9%, 99.98%, 88.63% and testing accuracy of 99.99%, 99.98%, 88.89% for binary type (benign/malicious), botnet type (benign/mirai/bashlite) and attack type (benign/ 8 attacks) classification respectively.



```
results_db1 = multi_model1.evaluate(x_test2, [y_test2_bin,y_test2_bot,y_test2_attack], batch_size=64)
running_secs_db1 = (dt.now() - start_db1).seconds
print(running_secs_db1)
```

```
s/step - loss: 0.1858 - binary_output_loss: 0.0023 - botnet_output_loss: 0.0043 - attack_output_loss: 0.1792 - binary_output_accuracy: 0.9999 - botnet_output_accuracy: 0.9998 - attack_output_accuracy: 0.8889
```

Fig 10.3. Multi-Output DNN accuracies output

The below Table 10.1 summarizes the results of both the models:

Table 10.1. Table with Results for Autoencoder and Multi-Output DNN

		Accuracy	**Precision**	**Recall**	**F1-Score**	**TPR**	**FPR**	**Inference time**
Autoencoder	**Unknown Botnet Attack detection**	97.12%	0.99	0.66	0.79	0.60	0.39	75 sec
Multi-Output DNN	**Binary Type classification**	99.99%	0.99	0.99	0.99	0.99	0.0005	8 sec
	Botnet Type classification	99.98%	0.99	0.99	0.99	0.99	5.22 e-05	
	Attack Type classification	88.89%	0.90	0.86	0.85	0.86	0.018	

The autoencoder ,being an unsupervised deep learning model, takes longer time for inference unlike the Multi-Output DNN model. Moreover, both the models produce high accuracies and F1-scores resulting in a reliable and efficient approach of IoT botnet attacks detection and classification.

CHAPTER 11

CONCLUSION AND FUTURE WORK

The motive behind the project was to develop a model that would not only detect and classify known IoT attacks, but also detect unknown attacks. The approach chosen to fulfil the motive was an anomaly detection model, accompanied by a multi-output classifier. The model can be used in the future as a part of network intrusion detection systems, in order to enable detection of all kinds of IoT attacks. IoT devices, due to their limited resource consumption, are characterised by a lack of complex cryptographic capabilities. This deems them vulnerable to all kinds of network attacks, allowing them to be an easy target. The project helps to ensure a reliable and legitimate process of attack detection in these IoT devices. IoT provides a large landscape for attacks and therefore is an area that needs attention in terms of anomaly and intrusion detection.

The project included the simulation of a virtual network, in order to generate an attack on a target device within the network. The attack would represent the unknown traffic data to be detected as an anomaly. The data from the attack was wrangled to form a new dataset, which would be used to test the model. The model used for anomaly detection was an autoencoders model. The N-BaIoT dataset was used for training purposes as the dataset contains attack data pertaining to two very popular IoT botnets. The model was trained using the dataset along with the generated benign data, and tested on all the benign and malicious data, to ensure that it could detect unknown or newer botnet attacks. The detected known data was further classified into various categories based on its binary type, botnet type and mode of attack using a Multi-Output DNN model.

The completion of the project involved:

- Simulating a DDoS attack within a virtual network, in order to test the anomaly detection aspect of the model
- Wrangling the data from the simulation, in order to use it as a testing dataset for the model

- Constructing an anomaly detection model using autoencoders to detect unknown botnet attacks
- Constructing a Multi-Output DNN model for further classification of the detected known data

All of the steps detailed above, have been completed successfully. For future work, scores can be improved and the model can be deployed at a network level, in order to facilitate real-time detection.

REFERENCES/BIBLIOGRAPHY

[1] Mirsky, Yisroel & Doitshman, Tomer & Elovici, Yuval & Shabtai, Asaf. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. 10.14722/ndss.2018.23211.

[2] Apostol, Ioana, Marius Preda, Constantin Nila, and Ion Bica. 2021. "IoT Botnet Anomaly Detection Using Unsupervised Deep Learning" Electronics 10, no. 16: 1876. https://doi.org/10.3390/electronics10161876

[3] Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.

[4] S. Sriram, R. Vinayakumar, M. Alazab and S. KP, "Network Flow based IoT Botnet Attack Detection using Deep Learning," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020, pp. 189-194, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162668.

[5] S. Nõmm and H. Bahşi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 1048-1053, doi: 10.1109/ICMLA.2018.00171.

[6] Ullah, Imtiaz, and Qusay H. Mahmoud. 2020. "A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks" Electronics 9, no. 3: 530. https://doi.org/10.3390/electronics9030530.

[7] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa and F. T. H. d. Hartog, "ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets," in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 485-496, 1 Jan.1, 2022, doi: 10.1109/JIOT.2021.3085194.

[8] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, Benjamin Turnbull,
Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, Future Generation Computer Systems, Volume 100, 2019, Pages 779-796, ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.05.041.

[9] Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. A Novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks. *Electronics* **2019**, *8*, 1210. https://doi.org/10.3390/electronics8111210.

[10] W. Zhou, Y. Jia, A. Peng, Y. Zhang and P. Liu, "The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1606-1616, April 2019, doi: 10.1109/JIOT.2018.2847733.

[11] Chunduri, Hrushikesh & Kumar, T. & Putrevu, Venkata Sai Charan. (2021). A Multi Class Classification for Detection of IoT Botnet Malware. 10.1007/978-3-030-76776-1_2.

[12] Ullah, I.; Mahmoud, Q.H. A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks. *Electronics* **2020**, *9*, 530. https://doi.org/10.3390/electronics9030530

[13] R. Gandhi and Y. Li, "Comparing Machine Learning and Deep Learning for IoT Botnet Detection," 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 234-239, doi: 10.1109/SMARTCOMP52413.2021.00053.

[14] Prokofiev, Anton & Smirnova, Yulia & Surov, Vasiliy. (2018). A method to detect Internet of Things botnets. 105-108. 10.1109/EIConRus.2018.8317041.

[15] Wazzan, M.; Algazzawi, D.; Bamasaq, O.; Albeshri, A.; Cheng, L. Internet of Things Botnet Detection Approaches: Analysis and Recommendations for Future Research. *Appl. Sci.* **2021**, *11*, 5713. https://doi.org/10.3390/app11125713

APPENDIX A: DEFINITIONS, ACRONYMS AND ABBREVIATIONS

IoT – Internet of Things
DDoS – Distributed Denial of Service
DL – Deep Learning
AI – Artificial Intelligence
IDS – Intrusion Detection System
FPR – False Positive Rate
TPR – True Positive Rate
t-SNE - t-Distributed Stochastic Neighbour Embedding
PCAP – Packet Capture
DT – Decision Tree
DNN – Deep Neural Network
IF – Isolation Forest
LOF – Local Outlier Factor
SVM – Support Vector Machine
GPU – Graphics Processing Units
TCP – Transmission Control Protocol
ACK – Acknowledgement
RAM – Random Access Memory
UDP – User Datagram Protocol
VM – Virtual Machine
IP – Internet Protocol
VNet – Virtual Network
CSV – Comma Separated Values
MAC - Media Access Control Address
TPR – True Positive Rate
FPR – False Positive Rate

Anomaly Detection and Classification of IoT attacks using Autoencoders and Multi-Output DNN

Maria Abraham Pynadath
Student, Undergraduate
Department of Computer Science
PES University, Bangalore, India
empynadath@gmail.com

K J Pavithra
Student, Undergraduate
Department of Computer Science
PES University, Bangalore, India
kjpavithra2000@gmail.com

Sahil Elton Lobo
Student, Undergraduate
Department of Computer Science
PES University, Bangalore, India
elton.lobo21@gmail.com

Sanjana S Murthy
Student, Undergraduate
Department of Computer Science
PES University, Bangalore, India
sanjana2000murthy@gmail.com

Dr. Bharathi R
Professor
Department of Computer Science
PES University, Bangalore, India
rbharathi@pes.edu

***Abstract*—Botnet attacks are responsible for the largest Distributed Denial-of-Service (DDoS) attacks on record. The detection of botnet attacks has become crucial now more than ever due to the frequent emergence of newer botnets and botnet attacks An increase in vulnerable connected devices, and continued growth in the DDOS-as-a-service industry ensures relevance of botnets as a threat. IoT devices tend to focus on maximizing functionality whilst reducing resource utilization. Due to this, it becomes incredibly difficult for IoT device designers and programmers to include complex cryptographic mechanisms and security measures in the framework of these devices. These reasons make IoT devices an easy target for botnets.**

Thus, the purpose of the project is to detect unknown botnet attack mode data from known data and further classify the known data into respective botnet and attack types. One of the benefits of the approach is that it performs anomaly detection using Autoencoders, therefore overcoming the problem of detecting unknown or newer botnet attacks that may emerge. To perform the unknown botnet attack detection, a dataset has been generated by simulating an unknown botnet attack in a virtualized environment. Moreover, this approach further classifies known data into benign/malicious, botnet type and attack types using a Multi-Output DNN which is a model that existing papers have not explored. Thus, this project provides a holistic approach to IoT botnet attack detection.

***Index Terms*—IoT, Botnet, Autoencoder, Multi-Output DNN, Dataset Generation, Anomaly Detection, Classification**

I. Introduction

With the emerging threats that arise as a result of botnets and the DDoS-as-a-Service Industry, there is a significant need to combat these threats and set a new scheme into implementation whereby newer botnets can be detected and the mitigation process can commence. New botnets are emerging everyday, and the current schemes aim at detecting and mitigating known botnet families, leaving a wide gap for these newer unknown botnet families to take over and cause mass damage to all sorts of devices, leading to a corruption of the network as a whole. This is a severe problem as, in today's world, the internet and the network that connects us all has become immensely valuable and furthermore, disruptions to this network can cause leakage of sensitive information, loss of communication and hampering the work of several important industries. Another important fact to consider, with regard to botnets, is the frequent targeting of IoT devices. This targeting is due to the tendency of IoT devices to focus on maximizing functionality whilst reducing resource utilization. Due to this, it becomes incredibly difficult for IoT device designers and programmers to include complex cryptographic mechanisms and security measures in the framework of these devices. These reasons make IoT devices an easy target for botnets. It is due to the reasons listed above that "Anomaly detection and Classification of IoT attacks using Autoencoders and Multi-Output DNN" was chosen as the problem statement. Furthermore, it was concluded that making use of Advanced DL Techniques would be beneficial for this project, as these are newer fields of learning and contain more scope for discovering newer methods of detection. This was especially due to the fact that, anomaly detection was selected, which would enable the discovery of newer, unknown botnets and DL techniques would help facilitate this. Furthermore, this project was implemented with the goal of multi-output classification, as opposed to merely focusing on single output classification since predictions are required for three different classes. Typical classification approaches involves predicting one class at a time whereas during multi-output classification, multiple classes can be predicted simultaneously. This enables the binary, botnet and attack type classification to happen simultaneously thereby improving the classification efficiency. Thus, both methods were incorporated into this project, in order to provide an extensive array of results regarding the incoming network traffic within an IoT network. Along with this, in order to test the anomaly detection capabilities of the project, an unknown botnet attack dataset was generated.

II. Literature Survey

Authors Mirsky, Yisroel, et al. [1] have described an Intrusion Detection System that flags off abnormal network traffic. It implements a plug and play Network IDS that runs on a local machine. The core algorithm used is autoencoders that uses feature extraction framework prior to training the model. The entire system is implemented using Raspberry Pi. The Kitsune Framework used in the paper helps extract 115 features, using the decay windows just like N-BaIoT.

Apostol, Ioana et al. [2] have proposed unsupervised deep learning using anomaly detection for IoT botnet activity in a network. The approach involves the use of a subset of an already available dataset called Bot-IoT. The autoencoder model is trained on the benign traffic data after separation. Removal of unnecessary features has resulted in the input being only 16 features for the autoencoder. Next, a predictor is used to compare the reconstruction loss from the autoencoders to predict anomalies. The predictor uses a high threshold value so as to observe frequent flags than to miss attacks with a lower threshold value.

Y. Median et al., propose a novel metwork-based anomaly detection method using the N-BaIoT dataset to extract behaviour snapshots of the network and use deep autoencoders to detect anomalous network traffic from compromised IoT devices. Studies so far relied on emulated or simulated data. But the authors here perform empirical evaluation with real traffic data, gathered from nine commercial IoT devices infected by authentic botnets from two families-Mirai and BASHLITE. Raw network traffic data using port mirroring on the switch is used for the same. Using incremental statistics to perform the feature extraction, the training of the autoencoders is performed in a semi-online manner. The training is practical, and there is no storage concern.

S. Sriram., et al.[4] propose advanced cybersecurity solutions to IoT devices and smart city applications using deep learning-based botnet detection system. Network traffic flows is collected from the all connected devices in the packet capture (PCAP) format and converted into connection records. During training, these connection records are labelled manually. ML algorithms and DNNs are used. The framework initially classifies the connection record into either legitimate or attack. Then, it classifies the attacks into their corresponding attack categories and as well as to corresponding botnet families.

S. Nõmm and H. Bahsi [5] make use of three measurement approaches that results in considerably less computational complexity while selecting the candidate feature sets.The first approach identifies the features with higher entropy. The second one selects the features demonstrating higher variance as they can properly identify isolation boundaries. The third one uses Hopkins statistics that measures the similarity of the corresponding feature values with the uniform distribution. The data with chosen feature sets is then evaluated using the three most popular techniques for anomaly detection: local outlier factor (LOF), one class SVM and isolation forest (IF). Corresponding average accuracy scores and recall score values are computed by 10-fold cross-validation procedure. The main result of this study is that achieving low computational complexity by reducing the feature set is possible for an IoT botnet detection system in which the learning model is trained by only normal traffic. It is shown that such a trained model provides a reasonable accuracy and precision results.

Ullah, Imtiaz, and Qusay H. Mahmoud [6] proposes a two-level anomalous activity detection system for IoT networks. The level-1 model classifies network traffic as normal or abnormal. If the level-1 model detects the flow as an anomaly, then the flow will be forwarded to the level-2 model for further classification to find the category or subcategory of the detected anomaly. The level-1 model is trained for binary classification. The proposed model uses a decision tree classifier at level-1 to classify the network traffic into normal traffic or anomalous traffic. The level-2 model is trained to identify the category or subcategory of malicious flow. Random forest classifier is used at level-2 to categorize the network flow received from the level-1 model. The proposed level-1 model achieved 100 percent Accuracy, Precision, Recall, and F-1 score for the IoT Botnet Dataset. Accuracy, Precision, Recall, and F-1 score for normal network traffic measured as 100 percent, DoS and theft as 99.90, DDoS as 99.60, and reconnaissance as 99.99 An average Accuracy, Precision, Recall, and F-1 score value of 99.80 percent was achieved for the category and 99 percent for the subcategory label.

III. Proposed Methodology

The proposed methodology includes the simulation of an unknown attack and the generation of a dataset encapsulating the same. The generated dataset is combined with the N-BaIoT to train and test the autoencoder model. The detected known data derived from this model is given to the multi-output DNN.

A. Attack Simulation

The simulated attack type chosen is a DDoS attack, in order to pose similarities to that of a botnet attack. This attack is simulated on a target device, the packets of which is used to generate the new dataset. The attack and environment is simulated using a cloud-based platform. This is done to generate all 115 features in the N-BaIoT dataset, without using physical infrastructure. The packets are captured in .cap format and converted to .csv for further processing.

B. Dataset Generation

The dataset generated is made compatible with the N-BaIoT dataset. This is because the model constructed to perform detection was trained using this dataset. This dataset consisted of 115 features and all 115 features were to be generated for the new dataset. This would be implemented using data processing and wrangling techniques, along with various statistics for aggregation of the existing features.

C. Autoencoder

An anomaly detection model was constructed in order to detect unknown malicious records. This is done using an autoencoder. This model would be capable of detecting unknown or newer botnet attacks, thus, safeguards against emerging attacks. These abnormalities were to be decided, based on a threshold. This threshold would be set based on the training data, consisting of both benign and known malware. Records going beyond this threshold would be detected as anomalies. This would imply that these records were neither benign, nor known malware. The anomalies detected would then be singled out and flagged.

D. Deep Neural Network Model

A Multi-Output Classification model would be constructed in order to further classify the known data detected by the anomaly detection model as known benign/known malware. Multi-Output Classification is a type of machine learning that predicts multiple outputs simultaneously. This saves the time taken for classification and is therefore efficient. A typical DNN consists of input layer, hidden layers and a output layer and predicts one class at a time whereas a Multi-Output DNN consists of several output layers in order to predict multiple classes simultaneously. This would enable, not only detection of unknown malware, but also classification of known benign and malicious data. The known malware detected would further be classified into benign/malicious, the type of botnet they come under, along with the mode of attack taken by them to infiltrate the target device. After classification, these devices would be flagged based on their type of botnet and mode of attack.

E. Figures and Tables

IV. Implementation

A. Simulation Of Network Setup

The simulation environment used to generate the attack was Microsoft Azure Cloud. The attack was performed within an Azure VNet (Virtual Network) with public IP addresses attached to its components. The components of this network included a target VM which would be the victim of the attack performed, a public Network Interface Card to navigate incoming traffic to this victim, a Network Security Group to manage the DDoS Protection Standards Azure has in place and a public IP address to make the target accessible to incoming traffic. This file was then wrangled and transformed into the resultant dataset to test the anomaly detection aspect of the model.

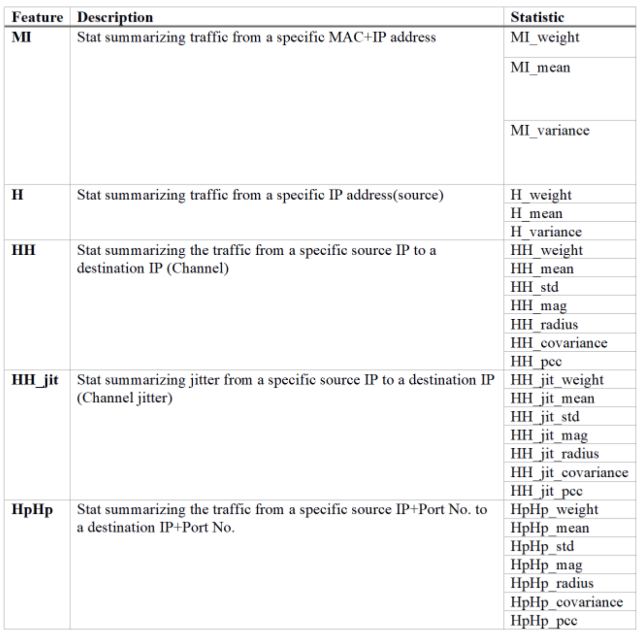
B. Breaking Point Cloud

Due to the DDoS Protection Standards enforced by Azure, an application known as BreakingPoint Cloud was used to bypass these Standards and perform the attack. In order to do this, the network was made accessible through public endpoints and IPs. It was linked to the Azure VNet created. Port 53 and the public IP address of the target VM were given to specify the target of the attack. The attack performed was a TCP SYN flood attack, a common mode of attack taken by various botnet families. The attack was performed by flooding the target VM with SYN packets. This would cause the target VM to send ACK packets as a part of the three-way TCP handshake. This was followed by sending further SYN Packets, instead of responding to the previous ACK packet. This caused the sessions to stay open until the session timeout occurred, thus rendering the target unable to respond to legitimate traffic. During the attack, 5,92,11,106 frames were sent, summing up to 3.75 GB of data and four attack nodes were used to perform this DDoS attack.

C. Packet Capture using Network Watcher

Azure Network Watcher was used to monitor the network. Azure Network Watcher enabled an overall view of the topology and had a feature within it, known as Azure Packet Capture. Azure Packet Capture was used to capture the network traffic in the form of packets. The packets were logged, throughout the duration of the attack, into a .cap file in the cloud storage location.

Table 1.1. Dataset Features



Feature	Description	Statistic
MI	Stat summarizing traffic from a specific MAC+IP address	MI_weight
		MI_mean
		MI_variance
H	Stat summarizing traffic from a specific IP address(source)	H_weight
		H_mean
		H_variance
HH	Stat summarizing the traffic from a specific source IP to a destination IP (Channel)	HH_weight
		HH_mean
		HH_std
		HH_mag
		HH_radius
		HH_covariance
		HH_pcc
HH_jit	Stat summarizing jitter from a specific source IP to a destination IP (Channel jitter)	HH_jit_weight
		HH_jit_mean
		HH_jit_std
		HH_jit_mag
		HH_jit_radius
		HH_jit_covariance
		HH_jit_pcc
HpHp	Stat summarizing the traffic from a specific source IP+Port No. to a destination IP+Port No.	HpHp_weight
		HpHp_mean
		HpHp_std
		HpHp_mag
		HpHp_radius
		HpHp_covariance
		HpHp_pcc

D. Packet Analysis using Wireshark

The .cap file was analysed using Wireshark to add various features, relevant to the generation of the dataset. The features added were MAC addresses of the source nodes and destination nodes along with the jitter of the packets. These features were added to the .pcap file which was then exported as a CSV file to be fed into the python script for data wrangling.

E. Data Wrangling using Python Script

The dataset required to have 115 features in total, consisting of, 23 features captured over 5 different time windows with a decay factor of (5,3,1,0.1,0.01). These 23 features were derived from the existing features using various statistics. Therefore, based on the different decay factors, the time windows chosen were 100ms, 500ms, 1.5sec, 10sec and 1min. The .csv file with the IP addresses(source and destination), MAC addresses(source and destination), packet jitter, length of packet, Port Number (source and destination) is fed into the python script. The script first reads this csv and writes the data into 2D arrays. The arrays are iterated through using the calculated time windows for each of the features mentioned below.

Table 1.2. Feature Calculation Formulae

Feature	Formula
Weight	$weight(W) = \Sigma X_i$
Mean	$mean(M) = \frac{W}{N}$
Variance	$variance(S^2) = \frac{\Sigma(X_i - M)^2}{N-1}$
Standard deviance	$std(S) = \sqrt{variance}$
Magnitude	$mag = \sqrt{{M_1}^2 + {M_2}^2}$
Radius	$rad = \sqrt{(\frac{\Sigma(X_i - M_1)^2}{n-1})^2 + (\frac{\Sigma(X_i - M_2)^2}{m-1})^2}$
Covariance	$cov = \frac{\Sigma(X_i - M_1)(X_i - M_2)}{N-1}$
Pearson's Correlation Coefficient (pcc)	$pcc = \frac{\Sigma(X_i - M_1)(X_i - M_2)}{\sqrt{\Sigma(X_i - M_1)^2\,\Sigma(X_i - M_2)^2}}$

F. Data Preprocessing

The generated unknown botnet dataset has been combined with N-BaIoT Data pre-processing includes data cleaning, data integration, data labeling, label encoding and normalization. Data cleaning is done to remove complex numbers from the unknown botnet dataset Data integration is done to combine the data device-wise for N-BaIoT since the dataset has been provided as multiple files and to combine N-BaIot dataset with unknown botnet dataset The integrated dataset has been labelled with three columns: binary type for binary classification (benign /malicious); botnet type for multiclass classification into (benign/ mirai/ bashlite/unknown malicious); attack type for multiclass classification (benign/ 8 attacks/unknown malicious) and another label bin type for known/unknown malicious data. The classes are categorically encoded and the data has been normalized using Min-Max Scaling. Exploratory Data Analysis is performed followed by the splitting of the dataset into training-set (80 percent) and testing-set (20 percent).

G. Autoencoder

Part of the novelty of the project comes from detecting unknown botnet attacks using anomaly detection. Anomaly detection is achieved by using autoencoders. The autoencoder is used to detect unknown botnet attacks using the column with 'known' and 'unknown malicious' labels To do this, it is trained with N-BaIoT and generated benign data Testing is done with data containing both benign and malicious records from N-BaIoT and generated dataset It finally detects the generated unknown botnet attack data

H. Multi-output DNN

The detected unknown botnet data is removed from the malicious data which results in the known data (N-BaIoT and generated benign data) This will be passed into a Multi-output DNN to classify all the three classes (binary type, botnet type, attack type) Multi-output classification is a type of machine learning that predicts multiple outputs simultaneously To the best of our knowledge, existing papers have not explored multi-output classification with this dataset The Multi-Output DNN consists of 1 input layer, 4 hidden layers and 3 output layers for binary type, botnet type and attack type classification respectively

V. Results and Discussion

A TCP SYN Flood Attack was successfully performed on a target node within Azure, with a total of 5,92,11,106 frames being sent, summing up to 3.75 GB of data. This data was captured and processed by the python script to generate the required dataset. The dataset generated contained all 115 features, required by the model.

The Autoencoder performed anomaly detection to detect unknown botnet attacks and the Multi-Output DNN was able to classify the known data into their respective sub classes successfully. For evaluation, both the models are evaluated on metrics such as accuracy, precision, recall, F1-Score and Inference time. The Autoencoder resulted in an testing accuracy of 97 percent with a threshold of 0.062.

The Multi-Output DNN resulted in a training accuracy of 99.9 percent, 99.98 percent, 88.63 percent and testing accuracy of 99.99 percent, 99.98 percent, 88.89percent for binary type (benign/malicious), botnet type (benign/mirai/bashlite) and attack type (benign/ 8 attacks) classification respectively.

		Accuracy	**Precision**	**Recall**	**F1-Score**	**TPR**	**FPR**	**Inference time**
Autoencoder	**Unknown Botnet Attack detection**	97.12%	0.99	0.66	0.79	0.60	0.39	75 sec
Multi-Output DNN	**Binary Type classification**	99.99%	0.99	0.99	0.99	0.99	0.0005	8 sec
	Botnet Type classification	99.98%	0.99	0.99	0.99	0.99	5.22 e-05	
	Attack Type classification	88.89%	0.90	0.86	0.85	0.86	0.018	

VI. Conclusion

The motive behind this project was to develop a model that would not only detect and classify known IoT attacks, but also detect unknown attacks. The approach chosen to fulfil this motive was an anomaly detection model, accompanied by a multi-output classifier. This model can be used in the future as a part of network intrusion detection systems, in order to enable detection of all kinds of IoT attacks. IoT devices, due to their limited resource consumption, are characterised by a lack of complex cryptographic capabilities. This deems them vulnerable to all kinds of network attacks, allowing them to be an easy target. This project helps to ensure a reliable and legitimate process of attack detection in these IoT devices. IoT provides a large landscape for attacks and therefore is an area that needs attention in terms of anomaly and intrusion detection.

The dataset known as N-BaIoT was used for training purposes as this dataset contains attack data pertaining to two very popular IoT botnets. The model was trained using this dataset and generated benign data, and tested on all the benign and malicious data, to ensure that it was capable of detecting unknown or newer botnet attacks. The detected known data was further classified into various categories based on it's binary type, botnet type and mode of attack using a Multi-Output DNN model.

References

[1] Mirsky, Yisroel Doitshman, Tomer Elovici, Yuval Shabtai, Asaf. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. 10.14722/ndss.2018.23211.

[2] Apostol, Ioana, Marius Preda, Constantin Nila, and Ion Bica. 2021. ”IoT Botnet Anomaly Detection Using Unsupervised Deep Learning” Electronics 10, no. 16: 1876. https://doi.org/10.3390/electronics10161876

[3] Y. Meidan et al., ”N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders,” in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.

[4] S. Sriram, R. Vinayakumar, M. Alazab and S. KP, ”Network Flow based IoT Botnet Attack Detection using Deep Learning,” IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020, pp. 189-194, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162668.

[5]] S. Nõmm and H. Bahşi, ”Unsupervised Anomaly Based Botnet Detection in IoT Networks,” 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018, pp. 1048-1053, doi: 10.1109/ICMLA.2018.00171.

[6] Ullah, Imtiaz, and Qusay H. Mahmoud. 2020. ”A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks” Electronics 9, no. 3: 530. https://doi.org/10.3390/electronics9030530.

Project ID:18

Title: Anomaly Detection and Classification of IoT Attacks using Autoencoders and Multi-Output DNN
Domain: Networks, IoT and DL

Abstract: Botnet attacks are responsible for the largest Distributed Denial-of-Service (DDoS) attacks on record. The detection of botnet attacks has become crucial now more than ever due to the frequent emergence of newer botnets and botnet attacks . An increase in vulnerable connected devices, and continued growth in the DDOS-as-a-service industry ensures relevance of botnets as a threat.

IoT devices focus on maximizing functionality whilst reducing resource utilization. Due to this, it becomes incredibly difficult for IoT device manufacturers and programmers to include complex cryptographic mechanisms and security measures in the framework of these devices. These reasons make IoT devices an easy target for attacks.

Thus, the purpose of the project is to detect unknown attack data from known data and further classify the known data into respective botnet and attack types which is done in two phases. To perform the unknown attack detection, a dataset has been generated by simulating an unknown botnet attack in a virtualized environment. The first phase involves anomaly detection of unknown attacks using Autoencoders. The second phase consists of multi-output classification of the detected known data using Multi-Output DNN into botnet and attack types. Hence, one of the benefits of the approach is that it overcomes the problem of detecting unknown or newer botnet attacks that may emerge. Moreover, this approach performs multi-output classification which is a technique that existing papers have not explored. Thus, this project provides a holistic approach to detect IoT attacks including unknown attacks.

Team:

Maria Abraham Pynadath PES2UG19CS224

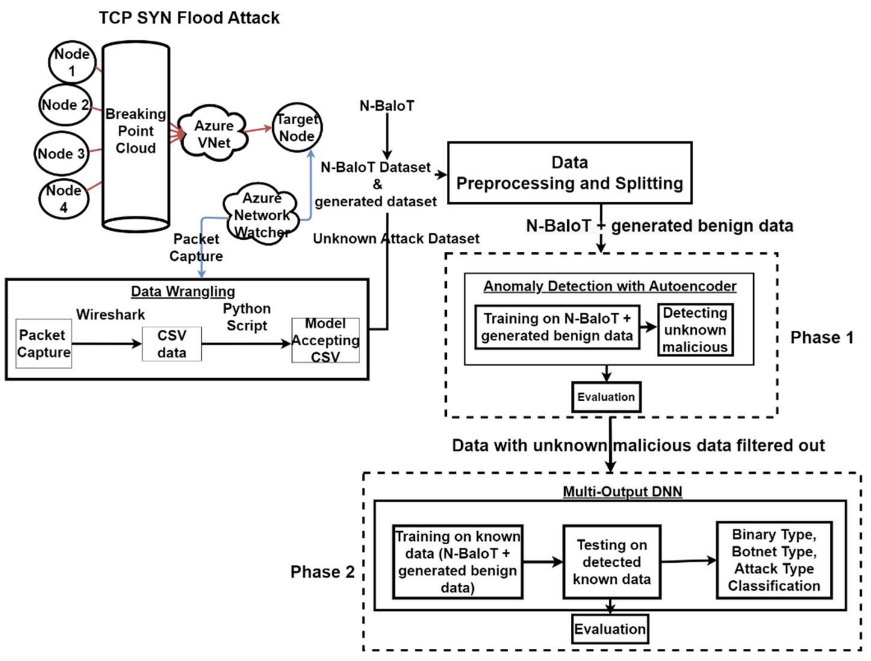
K J Pavithra PES2UG19CS278

Sahil Elton Lobo PES2UG19CS348

Sanjana S Murthy PES2UG19CS364

Supervisor: Dr. Bharathi R

Architecture diagram


TCP SYN Flood Attack
Node 1
Node 2
Node 3
Node 4
Breaking Point Cloud
Azure VNet
Target Node
Azure Network Watcher
Packet Capture
N-BaIoT
N-BaIoT Dataset & generated dataset
Unknown Attack Dataset
Data Preprocessing and Splitting
N-BaIoT + generated benign data
Data Wrangling
Packet Capture
Wireshark
CSV data
Python Script
Model Accepting CSV
Anomaly Detection with Autoencoder
Training on N-BaIoT + generated benign data
Detecting unknown malicious
Evaluation
Phase 1
Data with unknown malicious data filtered out
Multi-Output DNN
Phase 2
Training on known data (N-BaIoT + generated benign data)
Testing on detected known data
Binary Type, Botnet Type, Attack Type Classification
Evaluation