A PROJECT REPORT ON

ANOMALY BASED INTRUSION DETECTION SYSTEM FOR IOT NETWORKS USING RANDOM FOREST

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Of

SAVITRIBAI PHULE PUNE UNIVERSITY

By

SHRUTI HOUJI   **Exam   Seat No: 71707035C**

RHISHABH HATTARKI   **Exam   Seat No: 71707027B**

SAHIL DIXIT   **Exam   Seat No: 71706924K**

SANIKA PATIL   **Exam   Seat No: 71707451L**

Under the guidance of

PROF. M. R. DHAGE

Sinhgad Institutes
(Two blank spaces)

DEPARTMENT OF COMPUTER ENGINEERING
SINHGAD COLLEGE OF ENGINEERING, PUNE-41
Accredited by NAAC

2019-20

Date:

CERTIFICATE

This is to certify that the project report entitled

"Anomaly based intrusion detection system for IoT networks using Random Forest"

Submitted by

Shruti Houji	**Exam Seat No : 71707035C**
Rhishabh Hattarki	**Exam Seat No : 71707027B**
Sahil Dixit	**Exam Seat No : 71706924K**
Sanika Patil	**Exam Seat No : 71707451L**

is a bonafide work carried out by him/her under the supervision of Prof. M. R. Dhage and it is approved for the partial fulfillment of the requirements of Savitribai Phule Pune University , Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2019-20.

Prof. M. R. Dhage
Guide
Department of Computer Engineering

Prof. M. P. Wankhade
Head
Department of Computer Engineering

Dr. S. D. Lokhande
Principal
Sinhgad College of Engineering

Acknowledgements

Abstract

IoT devices are becoming popular day-by-day. Several vulnerabilities in IoT present the need for IoT security. The number of attacks on these devices keep increasing and most of them are slight variations of the previously known attacks, which can bypass the conventional firewall systems.

The existing systems are not suitable for IOT devices as IOT devices have low computational power. Those that use signature-based intrusion detection work only on known patterns and attacks, hence they cannot recognize newer attacks with unknown pattern. Also, many systems use cloud computing, which has a downfall that it needs access to internet at all times, also the cloud services are most often paid.

In this system, we have used Random Forest ML model to achieve a real time anomaly-based detection system. The anomaly-based intrusion detection system comes into effect when detecting newer attacks, that are not filtered by the firewall. It is capable of handling newer/unknown attacks, which signature based cannot. Also we are setting up the IDS on a local higher powered device rather than on cloud. The model has been trained on the IoT network traffic dataset created from the IoT node in consideration.

TABLE OF CONTENTS

List of Figures

List of Tables

Abbreviations

1. IoT – Internet of Things
2. ML – Machine Learning
3. IDS – Intrusion Detection System
4. CCTV – Closed Circuit Television
5. HIDS – Host-based Intrusion Detection System
6. NIDS – Network-based Intrusion Detection System
7. GUI – Graphical User Interface
8. Npm – Node Package Manager
9. Csv – Comma Separated Values
10. HTTP – HyperText Transfer Protocol
11. COCOMO – Constructive Cost Model
12. KLOC – Thousands (Kilos) of Lines Of Code
13. NDT – Nominal Development Time
14. IP – Internet Protocol
15. P – Polynomial time solvable algorithm
16. NP – Non-deterministic polynomial time solvable algorithm
17. FE – Front End
18. BE – Back End
19. UML – Unified Modeling Language
20. TCP – Transmission Control Protocol
21. DDoS – Distributed Denial of Service
22. MITM – Man In The Middle
23. WS – Web Socket
24. KNN – K Nearest Neighbours
25. CNN – Convolutional Neural Network
26. RNN – Recurrent Neural Network
27. Node MCU – Node MicroController Unit

ACM Keywords

• Security and privacy ~ Intrusion/anomaly detection and malware mitigation ~ Intrusion detection systems

• Security and privacy ~ Network security ~ Denial-of-service attacks

• Security and privacy ~ Network security ~ Mobile and wireless security

• Security and privacy ~ Security in hardware ~ Embedded systems security

• Computing methodologies ~ Machine learning ~ Machine learning algorithms ~ Ensemble methods ~ Bagging

• Computing methodologies ~ Machine learning ~ Learning paradigms ~ Supervised learning ~ Supervised learning by classification

• Computing methodologies ~ Machine learning ~ Machine learning approaches ~ Classification and regression trees

<div align="right">

Chapter 1
Introduction

</div>

1.1 Background and Basics

Internet of things, or IoT, is a system of interrelated "things" that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. These "things" could be computing devices, mechanical and digital machines such as built in sensors, monitors etc.

A good example of a network of IoT devices is the security system implemented by various locations which include, CCTV cameras, motion sensors, automatic locks, smoke detectors, temperature sensors etc.

IOT devices are becoming pervasive. They are used extensively in a lot of fields and their utility is just going to keep increasing. IOT devices help to automate things, reduce labour costs and facilitate smart living.

Hence, it is important to build an optimal system that can provide proper safety and security measures for IOT devices.

An intrusion detection system is one such system that can be a building block in making the IOT network as secure as possible. An intrusion detection system is a system that passively monitors the data exchange in the network and of the network with externals entities and looks for malicious activities that can be classified as an intrusion or attack. It then notifies the user or sends a notification to some other system which may or may not take action against the detected intruder. Simply put, if the network of devices is a home, an intrusion detection system is a CCTV camera.

Also, a step up from just an intrusion detection system is an anomaly based intrusion detection system. This type of system creates a profile of normal behaviour, and any

---

activity that falls outside of the normal category is marked as anomalous. Anomaly based system is better suited to defend the system against zero-day attacks.

There are certain challenges while building this system which are specific to IOT devices. For example, the low computational powers and limited amount of resources (such as space/memory). The IDS system should be build keeping in mind all these challenges and they should be overcome in the most efficient manner.

1.2 Literature Survey

Table 1.1 Literature Survey

Title	Year	Publication	Data -set	HIDS /NIDS	Anomaly /Signa-ture	Model used	Drawbacks
Anomaly based intrusion de-tection system through feature selection analysis and building hybrid efficient model	2018	Journal of computational science 25(2018) 152-160	NSL-KDD	NIDS	Anomaly	Hybrid	Not imple-mented for iot Devices

Distributed at- tack scheme deep approach for Internet of Things detection using learning	2017	Future Genera-tion Computer Systems (2017)	NSL-KDD	NIDS, fog	Signature	Deep learning	signature based, new attacks cannot be recog-nized
Towards Ma-chine Learning Based IoT Intrusion Detection Service	2018	Springer In-ternational Publishing AG, part of Springer Nature 2018	UNSW-NB15	NIDS, Cloud	Anomaly	Neural net-work, random forest	uses cloud com-puting, which increases the resource require-ments
Host based intrusion de-tection system with combined CNN/RNN model	2019	Springer Na-ture Switzer-land AG 2019	ADFA-LD	HIDS	Anomaly	RNN with GRU	HIDS, and not NIDS Perfor-mance doesnt match ensemble models

Anomaly-based intrusion detection of iot device sensor data using provenance graphs	2018	1st International Workshop on Security and Privacy for the Internet- of-Things (IoTSec)	longi - tudinal data on the thermal	HIDS	Anomaly	Prov - enance graphs	Works with offline data, doesn't work with real time data

1.3 Project Undertaken

1.3.1 Problem Definition

Design and implement an anomaly Based Intrusion Detection System in IoT Networks using Random Forest which will passively monitor the IoT network traffic and alerts the user for any intrusion detected.

1.3.2 Scope Statement

Product can be used in any IOT network for the purpose of the security. Since use of IOT devices is increasing, day by day, in all fields and sector, there is an increasing need for security mechanisms designed specifically for IOT devices or networks. The main function of the IDS system is to detect anomalous behavior that can be caused by the attacks. So it can be deployed anywhere where we need security measures in place to protect IOT devices.

1.4 Organization of Project Report

The project report is organized as follows:

1.4.1 Chapter 1

Chapter 1 is Introduction. It gives the background and basics of the project. It is followed by a detailed literature survey of similar works in the past. Problem statement and scope of the project are defined as well.

1.4.2 Chapter 2

Chapter 2 is Project Planning and Management. It has details of the system requirement specifications which include functional and non-functional requirements, system overview, deployment environment, external interface and other requirements. The project process model applicable to this project is also mentioned. Cost estimate analysis and time line scheduling is done as well.

1.4.3 Chapter 3

Chapter 3 is Analysis and Design. It consists of idea matrix, mathematical model and feasibility analysis. All the analytical and design diagrams are also included in this chapter. These diagrams include use case diagram, activity diagram, architecture diagram, class diagram, ER diagram, sequence diagram, state transition diagram and deployment diagram.

1.4.4 Chapter 4

Chapter 4 is Implementation and coding. This explains the details of the working of the system as a whole along with module wise operations. It provides an insight of how each module works individually. It has important code snippets of all the modules. It shows screenshots that help visualize the system better.

1.4.5 Chapter 5

Chapter 5 is Testing. In this chapter, test cases regarding different types of testing are given. The types of testing included are unit testing, integration testing and acceptance testing.

Chapter 2
Project Planning and Management

2.1 Introduction

This chapter covers the project planning and management details. It also covers System Requirement specifications. SRS is considered as the base for the effort estimations and project scheduling.

2.2 System Requirement Specification (SRS)

2.2.1 Detail System Requirement Specification

2.2.1.1 System Overview

Product Perspective

The system has been inspired by the already existing intrusion detection systems which attempt to protect the home-network against attacks and intrusions. We are taking these pre-existing models and combining them to get additional advantages and reduce the downsides as much as possible. Till date, there have been very few attempts at making an IDS for IOT devices. Most of the IDS present in the market cater to non-IDS networks. We are taking the principles of these IDS systems and modifying them as per requirement of an IOT network.

Product Functions

1) User authentication
2) Data connection establishment
3) Intrusion detection analysis

4) Attack Notification

5) Dashboard

6) Network log analysis

7) Visualizations

User Classes and Characteristics

Factories and companies employing a network of IOT devices. Usually the IOT networks for these users are larger and require higher level security systems as economic factors are involved. Residential habitants employing network of IOT devices. Includes home-IOT networks (smart homes), or residential society IOT networks like CCTVs, smoke detectors, fire alarms etc. These networks are usually smaller than the previous.

Operating Environment

The system will be implemented on Linux. Python is used for programming the backend using Django Framework. Languages such as Python and Sqlite and platform like apache server are used for database and connection. Libraries like sklearn, pandas etc. will be used. For real life implementation, IOT data will be connected from Tshark.

Design and Implementation Constraints

• The user will be notified through a web app, which should be installed on their system.

• All the nodes of the IOT network should be connected to the main server where the processing is to take place.

• The device to be notified also has to be part of the given network.

• Processing time should be as low as possible.

• Backend should be connected to database.

• Wifi (802.11) supported on all the IOT devices.

User Documentation

• User manual.

Assumptions and Dependencies

• We are assuming that the machine has the required resources (memory and processing power etc.) and capabilities to run the system.

• The user has updated system. We are assuming that the system has the required packages and dependencies (such as Django and Xampp) to run the system.

2.2.1.2 Functional Requirements(System Features)

2.2.1.2.1 Sensing of the data using ultrasonic sensor

Description and Priority

First, sensors or devices collect data from their environment. The benefit of this feature is much larger and the cost of this feature is reasonably low. Only risk of this feature is that it sometimes may not work as intended.

Stimulus/Response Sequences

Once the IOT device is started, distance data is processed and sent from the node to the apache server. The network traffic is monitored using Tshark and dumped into a csv file. The python script in the backend reads this csv file, preprocesses it and feeds it to the ML model. If an intrusion is detected, it is notified to the user on the front end.

Functional Requirements

IOT sensor node equipped with Arduino and nodemcu.

2.2.1.2.2 Connectivity

Description and Priority

The sensors/devices are to be connected to the network through wifi. This feature is of high priority because the sensed data needs to be processed further. The data will be processed only if it gets proper way to reach to the server. Only risk of this feature is that it can sometimes not send the data to the server.

Stimulus/Response Sequences

Data sensed is sent to the server.

Functional Requirements

Apache server of the network along with wifi module.

2.2.1.2.3 Data Processing

Description and Priority

Once the data gets to the server, software performs processing on it. Here IDS software is used to determine the category of the attack. This feature is of high priority because this is the main objective behind selecting this project. Only risk of this feature is that it can sometimes not categorize the attack perfectly.

Stimulus/Response Sequences

In response, it will determine whether the attack was normal or malicious.

Functional Requirements

IDS software.

2.2.1.2.4 User Interface

Description and Priority

Next, the information is made useful to the end-user in some way. This is an alert to the user (text, notification, etc.). This feature is of medium priority. Only risk of this feature is that it can sometimes not work as intended.

Stimulus/Response Sequences

In response it will give notification of the attack on the GUI.

Functional Requirements

GUI

2.2.1.3 Non- Functional Requirements

Performance Requirements

The data transfer from the IOT node to the apache server should take as little time as possible for proper real time functioning of the system. Also, the processing of the data should be fast enough so as to give proper result as soon as the attack happens, i.e. the time between the occurrence and detection of the attack should be minimum.

Safety Requirements

The backend could crash resulting in failure of the whole system. If the system takes up all the processing power, that machine could crash. Regular maintenance of the networking components, such as checking the proper functioning of the Arduino, ultrasonic sensor and nodemcu should be done.

Security Requirements

User need to signup first and then login to get access to the system, in this case users data should be protected. Systems should be secure against unauthorized access to any of their data, unauthorized use of them or any of their components. Details regarding IOT devices and their data should also be protected.

Software Quality Attributes

Our software has many quality attribute that are given below: -

• Availability: This software is freely available to all users. The availability of the software is easy for everyone.

• Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the software developer.

• Reliability: The performance of the software is better which will increase the reliability of the Software.

• User Friendly: Since, the software is a GUI application, the output generated is much user friendly in its behavior.

• Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.

• Security: Users are authenticated using many security phases so reliable security is provided.

2.2.1.4 Deployment Environment

Software Requirements-

1) Python 3 - Coding language used in our project

2) Django Framework - Used for backend work

3) Node - Used for executing JavaScript code outside of a web browser

4) Npm - Used as package manager for the Node JavaScript platform

5) React.js - Used for frontend work

6) Sqlite - Used for database

7) Xampp server - Used for storing network data

8) Tshark - Used for network data analysis

Hardware Requirements-

1) Arduino - For reading inputs of ultrasonic sensor and turn it into an output as distance data

2) Nodemcu - Node Microcontroller Unit used to build IOT network

3) Ultrasonic sensor - Used for collecting sensor data

4) Connecting wires - Used for connecting nodemcu to Arduino and ultrasonic sensor

2.2.1.5 External Interface Requirements

User Interfaces

1) User Signup page

• Username

• Email id

• Password

• Signup(Button)

2) User login page

• Username

• Password

• Login(Button)

3) User Notification page

Notifies when the network is under attack or some anomalous activity is detected.

4) User dashboard page

Displays ultrasonic data from php script in animated format on web page.

5) User Network Logs page

Displays the real time Tshark data in table format.

6) User Visualisations page

Displays graphs and confusion matrix for various types of data.

Hardware Interfaces

Hardware interfaces for a web application will run on Linux, Windows or Mac. For connection of IOT node to the apache server will require a network. For a larger model, the network and servers will increase. Here IOT network consists of ultrasonic sensor, Arduino and Nodemcu which sends data to the apache server using wifi module.

Software Interfaces

1)   **Operating system** –

We have chosen Linux operating system because of its user friendly features, benefits of security and control without complexity and unrealistic costs.

2)   **Database** –

Sqlite using apache server

3)   **Python** –

Django is used to program and implement the backend and React.js used to create web pages for frontend. IDS is designed to run in the background of an IoT network, where distance data is processed and sent from the node to the apache server. The network traffic is monitored using Tshark and dumped into a csv file. The python

script in the backend reads this csv file, preprocesses it and feeds it to the ML model. If an intrusion is detected, it is notified to the user on the front end.

Communications Interfaces

1) Web application runs on a web browser

2) HTTP is used for notification system

3) Wifi module is used for data transfer from IOT node to the apache server.

2.2.1.6 Other Requirements

1) **Configuration** –

Systems should be configurable for detection and reaction, as feasible for alerts, updates, protocol and port coverage, and detection-threshold levels. Systems should be configurable treat identified IP or other network addresses exceptionally; for example, configurable to never block or shun network activity from one or more IP addresses.

2) **System Updates** –

These capabilities should be engineered in such a manner as to allow updates during operational use of the products without disruption.

3) **Ease of use** –

Operator console should not require undue expertise to operate; experts may reside external to operator staff.

2.2 Project Process Modeling

Iterative waterfall model is the best suited for this project. Iterative waterfall model can be thought of as incorporating the necessary changes to the classical waterfall model to make it usable in practical software development projects. It is almost same as the classical waterfall model except some changes are made to increase the efficiency of the software development.

The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model. In our project, every phase is well defined and to be executed one after the other sequentially, like the waterfall model. But if need be, there is space for going back to previous stages making changes. Hence, iterative waterfall is to be used for this project.

2.3 Cost & Efforts Estimates

As per the basic COCOMO cost estimation formula projected cost for our product,

Development Effort = a1 x (KLOC) a2 PM

$\qquad$ = 2.4 * (4) 1.05

$\qquad$ = 10 Person Months (approx)

Nominal development time   = b1 x (Effort) b2 Months

$\qquad$ = 2.5 * (10) 0.38

$\qquad$ = 6 months (approx)

Cost required to develop the product $\qquad$ = NDT x Average salary per month x

$\qquad$ members

$\qquad$ = 6 * 5000 * 4

$\qquad$ = Rs 120,000/-

Where,

KLOC is the estimated size of the software product expressed in Kilo Lines of Code

a1, a2, b1, b2 are constants for each category of software products.

2.4 Project Scheduling

Time line chart
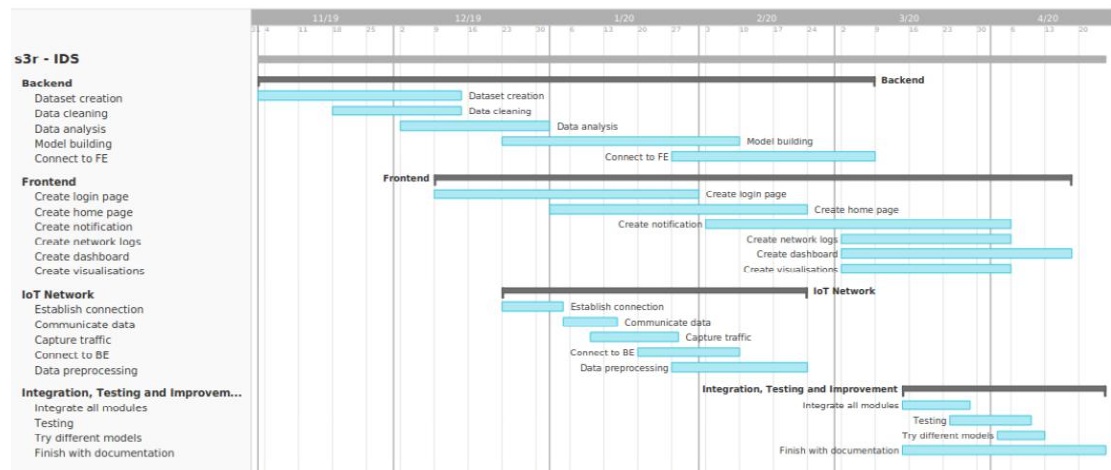
Fig 2.1 Time line chart

Chapter 3
Analysis and Design

3.1 Introduction

This chapter covers the analysis and design of the considered system.

3.2 Idea Matrix

Table 3.1 Idea Matrix

Idea Matrix		
Increase	1)It increases Network security 2)It increases network monitoring	1)Network security 2)Network monitoring
Improve	1)It improves zero-day attack detection 2)Improves IoT security 3)Improves awareness of user about network intrusions	1)Unknown attacks 2)IoT security 3)User awareness
Ignore	1)It ignores the normal behavior in the network	1)Normal behavior
Deliver	1)It delivers efficient intrusion detection system for IoT devices	1)IoT security

Document	1)Documentation of the project for researchers to understand and improve on the project in the future.	1)Project documentation
Decrease	1)It decreases efforts taken to create signatures	1)Effort
Educate	1) Educate project members with security concepts, machine learning, Tshark, web sockets, scripting, UI development 2)Educate user	1)Project members 2)User
Evaluate	1)Continuously monitor network traffic for intrusions 2)Evaluate packets to determine if there is an intrusion	1)Network Traffic 2)Packets
Experiment	1)Experiment on different types of attacks 2)Experiment on different types of ML models	1)Attacks 2)ML Models
Accelerate	1)It increases the processing speed by using a separate device instead of the IoT device itself	1)Processing speed
Analysis	1)It analyses detected anomalies to determine what type of attack it is	1)Category of attack

Advertise	1)Advertise about IoT security and need of an IDS 2)Create a GitHub page, which will include code, documentation and process	1)Security awareness 2) GitHub

3.3 Mathematical Model

Let S be the solution set for the given problem statement.

S = { Input, Function, Output, Terminate, Success, Failure }.

Where,

Input = Input to the System.

Function = Functions of the system.

Output = Output of the system.

Terminate = Terminating Condition of the System.

Success = Success cases for the System.

Failure = Failure cases for the system.

1. Input = { UserName, Password, Data Packets }

a. UserName :- user id

b. Password : user password

c. Data Packets:- A data packet is a unit of data made into a single package that travels along a given network path.

2. Function = { Login auth, Network connection, train test, intrusion detection, Notification }

a. Login auth: This function will take Username and Password as the input and gives the authorization rights to the user accordingly.

IF Y = F(X) is the login auth function then

X: - Username and Password taken as an input.

Y: - Authorization rights of a user.

b. Network connection = This function will take port number, IP address as an input to give the status of the connection network.

IF Y=F(X) is the Network Connection Function, then

X:- Inputs to setup a network, connecting to a device, routing the incoming data and device configuration.

Y:- Status of the network connection.

c. train test: This function will take features as an input to give the accuracy of the model and further it will be turned to improve the accuracy of the model. Random Forest has been used as a Machine learning algorithm for building the model.

IF Y=F(X) is the function to test and train the model then,

X:- Features of the self created dataset as an input.

Y:- Accuracy of the model

d. intrusion detection : This function will take features like frame number, frame time, frame length, source mac address, destination mac address, source IP address, destination IP address, IP protocol, IP length, TCP length, TCP source port, TCP destination port, packet info as an input and notifies the user the system for the intrusion.

• Random Forest Classifier has been used for the building and implementing the model.

Basically, a random forest is an average of tree estimators. As with nonparametric regression, simple and interpretable classifiers can be derived by partitioning the range of X.

Let n = A1, . . . , AN be a partition of X. Let Aj be the partition element that contains x. Then h(x) = 1 if XiAj Yi XiAj (1 Yi) and h(x) = 0 otherwise.

We conclude that the corresponding classification risk satisfies R(h) R(h) = O(n1/(d+2)).

Trees are useful for their simplicity and interpretability. But the prediction error can be reduced by combining many trees.

These are bagged trees except that we also choose random subsets of features for each tree. The estimator can be written as

$$\widehat{m}(x) = \frac{1}{M} \sum_j \widehat{m}_j(x)$$

where mj is a tree estimator based on a subsample (or bootstrap) of size a using p randomly selected features. The trees are usually required to have some number k of observations in the leaves. There are three tuning parameters: a, p and k. You could also think of M as a tuning parameter but generally we can think of M as tending to . For each tree, we can estimate the prediction error on the un-used data. (The tree is built on a subsample.)

Averaging these prediction errors gives an estimate called the out-of-bag error estimate.

> IF Y=F(X) is the function then,
>
> X: frame number, frame time, frame length, source mac address, destination mac address, source IP address, destination IP address, IP protocol, IP length, TCP length, TCP source port, TCP destination port, packet info as an input
>
> Y: Classification – attack/normal.

e. Notification: This function takes the input which is provided by the function (intrusion detection) and notifies the user.

> IF Y=F(X) is the function then,
>
> X: Output of the function(intrusion detection).
>
> Y: Intrusion message to the user.

3. Output = { display intrusion msg }

> a. display intrusion msg : display error message if any intrusion occurs.

4. Intermediate Results

> a. Successful working of module.
>
> b. Successful Working of Network.
>
> c. Successful User authentication.

5. Terminate = { Invalid details, Network failure, Timeout}

> a. Invalid User Authentication.
>
> b. Network failure
>
> c. timeout

6. Success

> a. Successful user login.
>
> b. Successful connection establishment of nodes and ids.
>
> c. Successful detection of intrusion.

    d. Displaying the results.

    e. Appropriate error messages in case of invalid input.

7. Failure

    a. Web app Failure.

    b. Hardware faults.

    c. Network establishment failure.

    d. Not displaying required results.

3.4 Feasibility Analysis

The problem is to detect an intrusion in a IoT network. The main objective of our project is to provide a real-time solution for any possible intrusions. In the proposed system, we are planning to use anomaly-based detection system. The anomaly-based intrusion detection system comes into effect when detecting newer attacks, that are not filtered by the firewall. Random Forest is an ensemble model of decision trees.

Training Complexity: $O((n^2)pntrees)$ Prediction Complexity: $O(pntrees)$

Calling n the number of training sample, p the number of features, ntrees the number of trees (for methods based on various trees), where, n= 21000 (Approx) p = 17 ntress= 4

Hence $O(pntrees)$ runs in polynomial time complexity. The algorithm falls in P. Hence, the given problem is NP.

3.5 Architecture Diagram
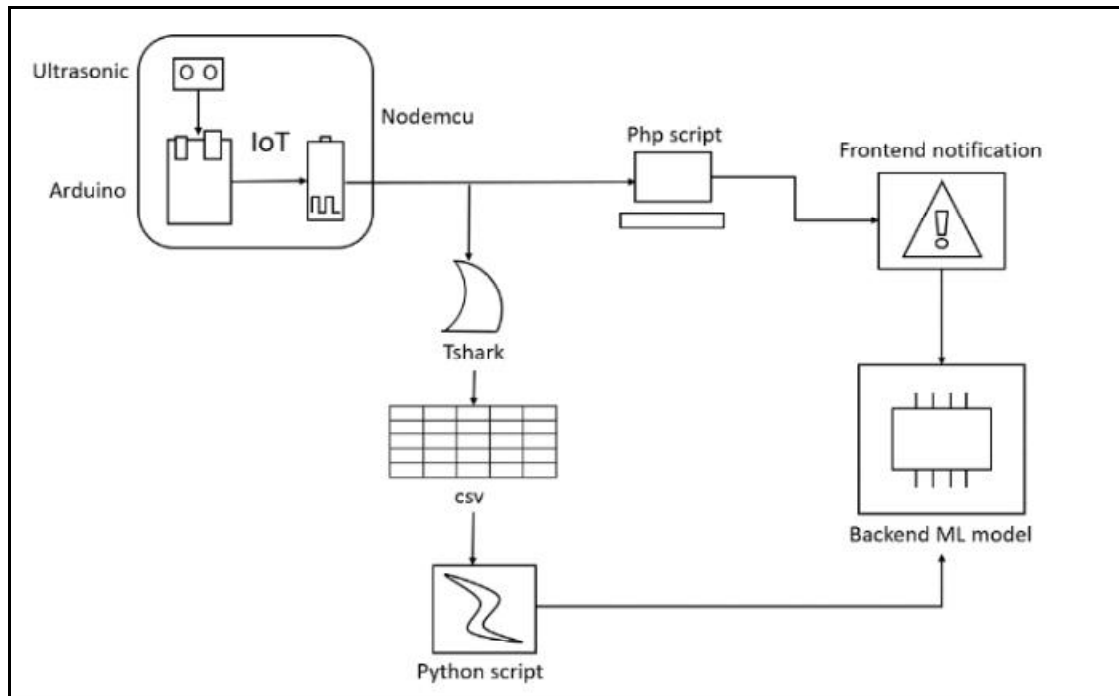
Fig 3.1 Architecture Diagram

3.6 UML Diagrams

3.6.1 Use case diagram

A use case diagram shows the relationship between the user and the different use cases in which the user is involved. The following diagram shows how the user, IoT device and the backend interacts with the system. The user can login/register and view the network details, notifications, etc. while other details are hidden.
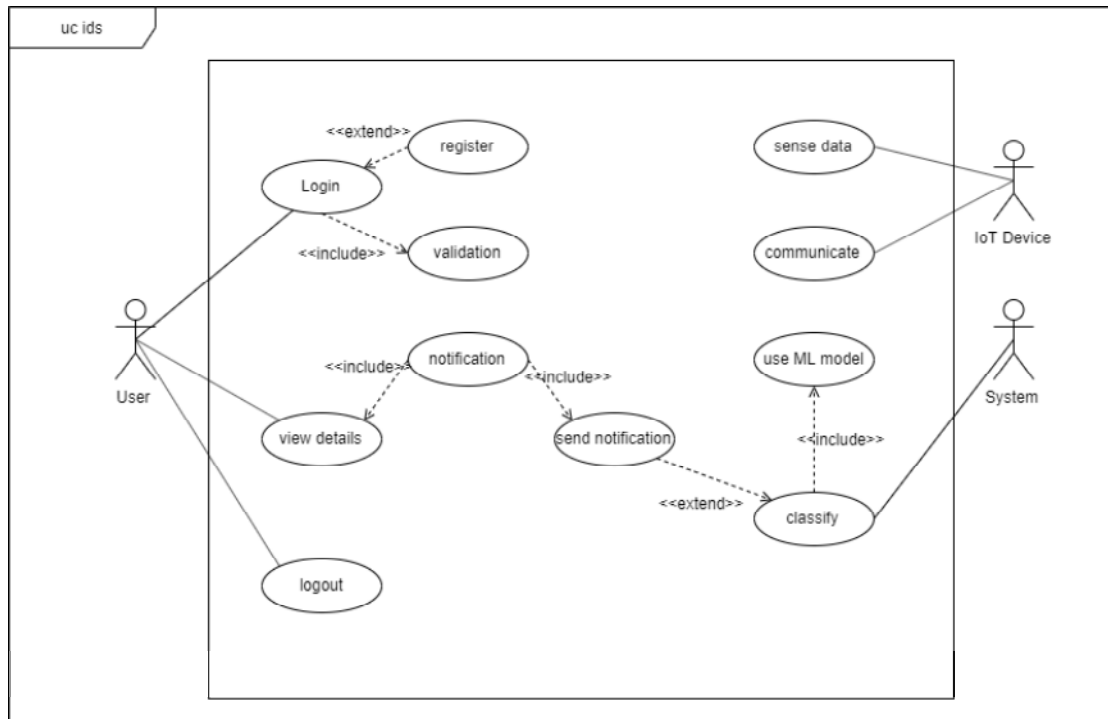
Fig 3.2 Use Case Diagram

3.6.2 Activity Diagram

The activity diagram shows the stepwise flow of activities and actions. Here it explains the flow of the system, how it starts with login/registration. Then it user can perform their regular tasks while the IDS keeps working in the background and notifies the user when an attack is detected.
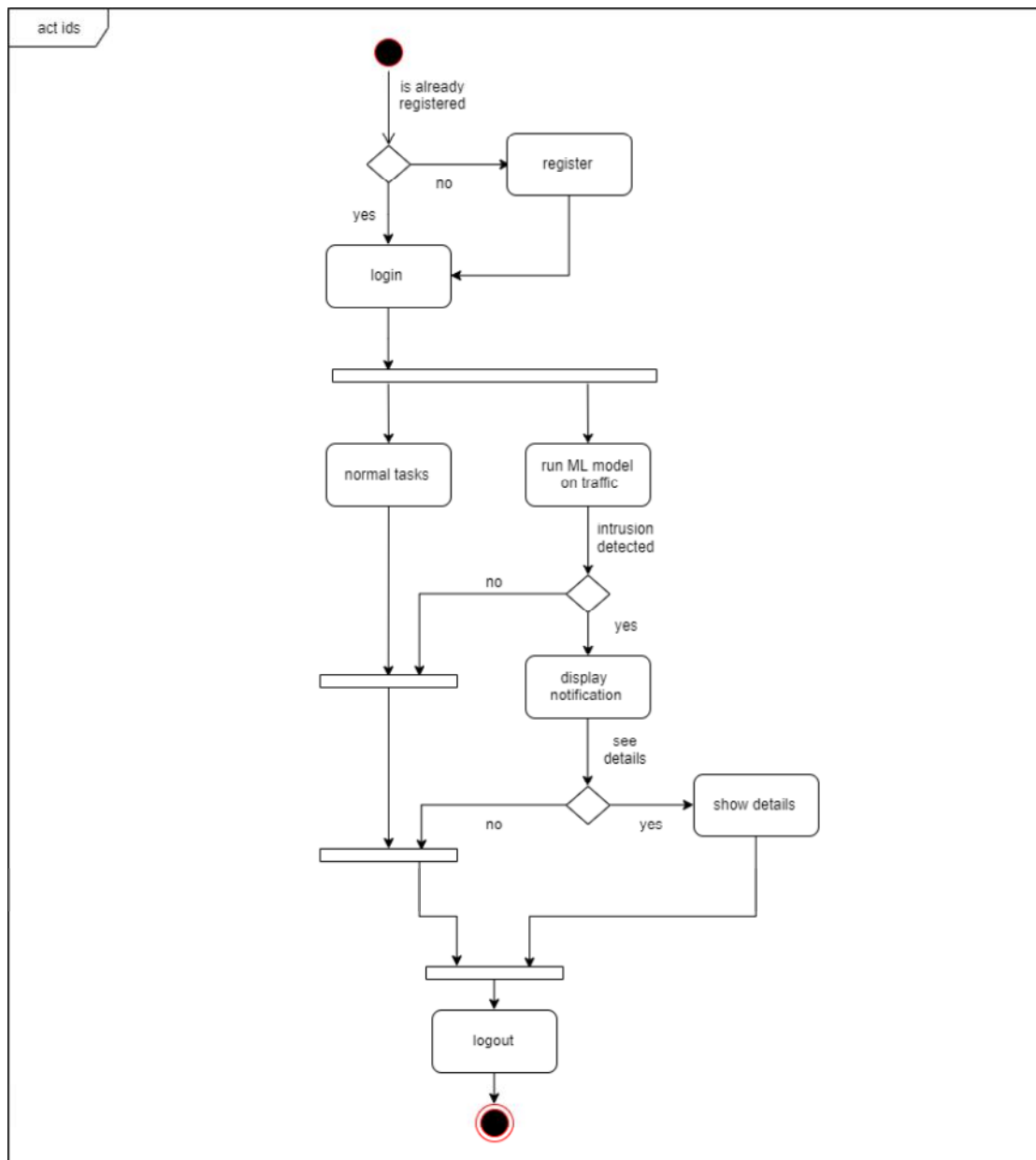
Fig 3.3 Activity Diagram

3.6.3 Class Diagram

A class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Here it shows the structure of objects such as user, notification, model, etc.
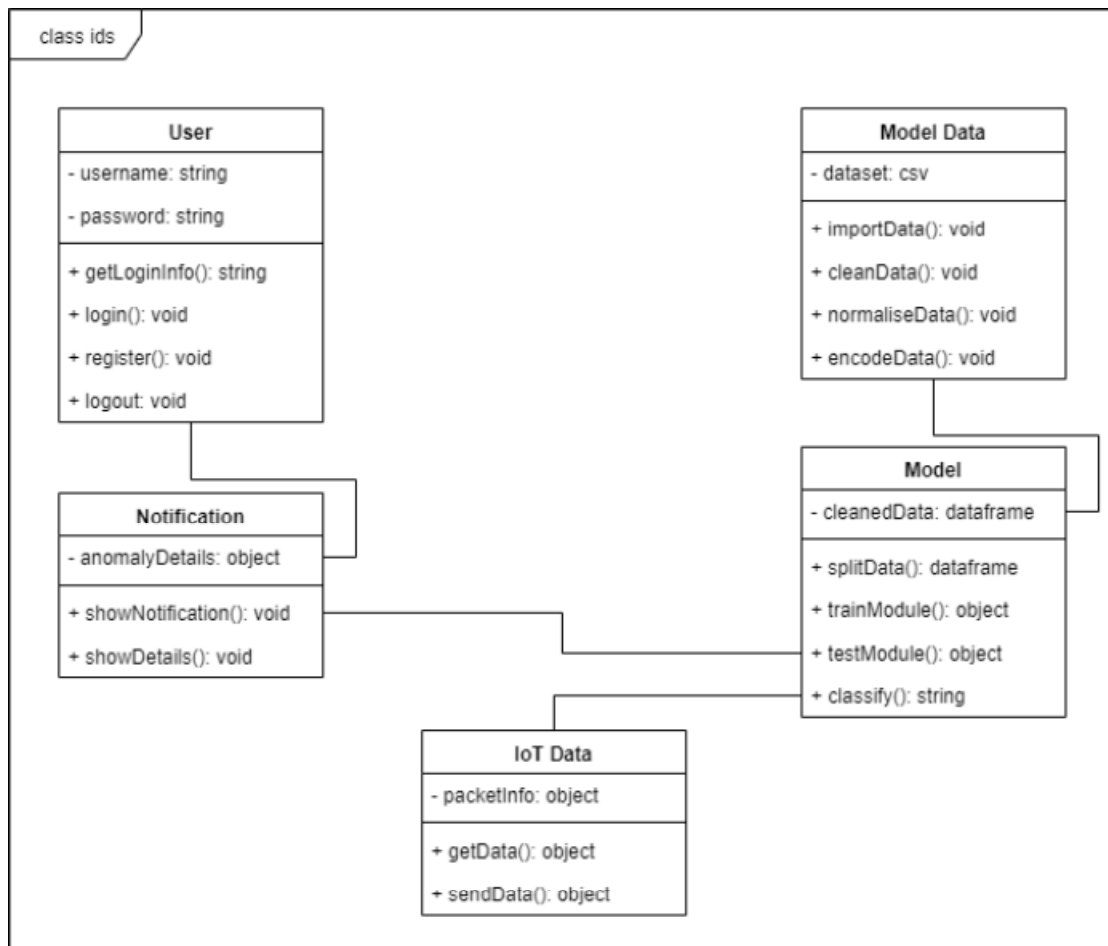
Fig 3.4 Class Diagram

3.6.4 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. The following diagram shows the sequence of steps that follows one after the other in the system. It shows when the connection is established and when the model starts giving the intrusion alerts.
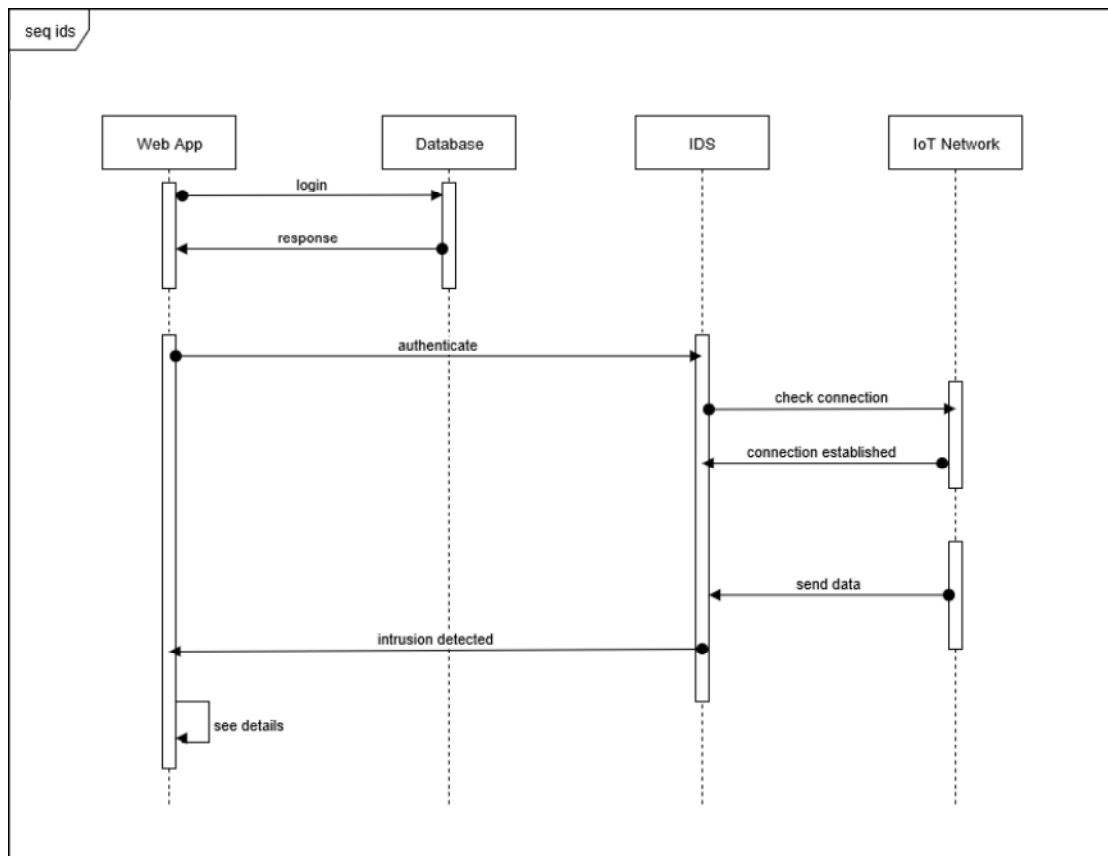
Fig 3.5 Sequence Diagram

3.6.5 Package Diagram

A package diagram in the Unified Modeling Language depicts the dependencies between the packages that make up a model. The diagram explains that the backend is dependent on the IoT network for data. The ML model is dependent on the backend. The GUI is dependent on the ML model for the attack classification to create the notification.
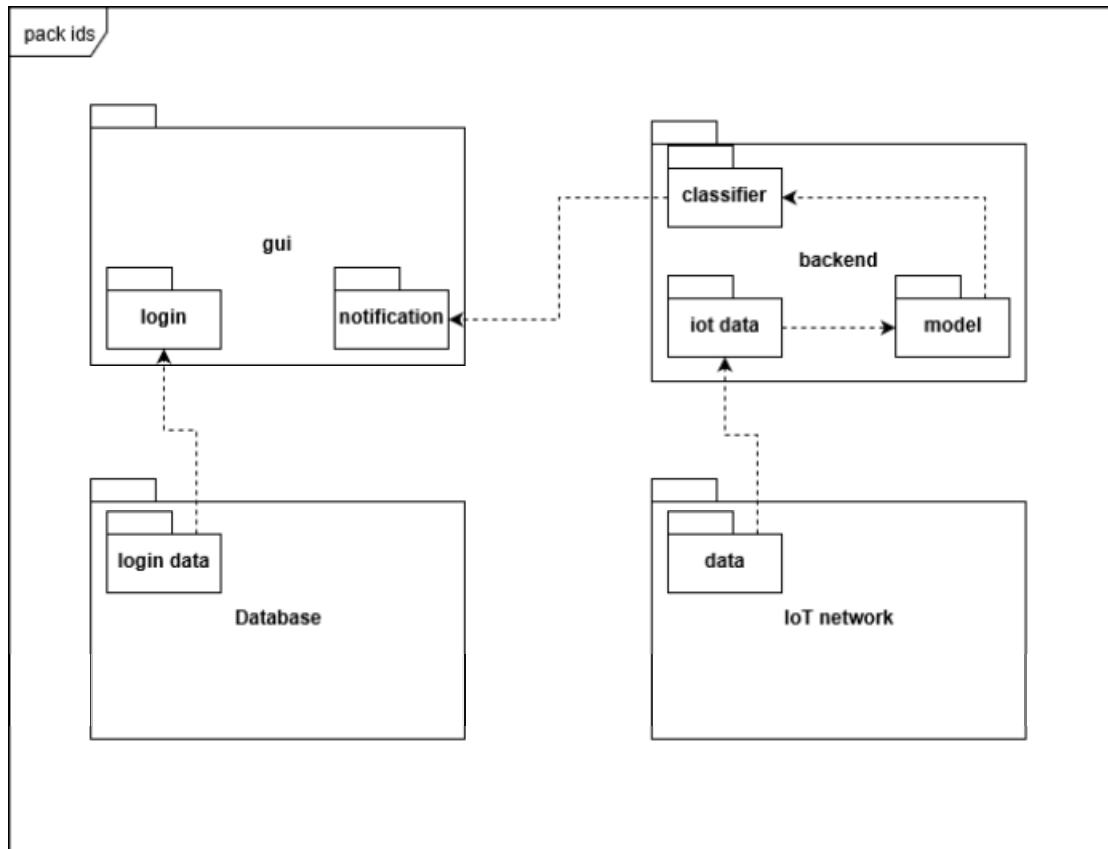
Fig 3.6 Package diagram

3.6.6 State Machine Diagram

UML state machine,[1] also known as UML statechart, is a significantly enhanced realization of the mathematical concept of a finite automaton in computer science applications as expressed in the Unified Modeling Language (UML) notation.

Anomaly based intrusion detection system
in IoT networks using Random Forest
Group No 27
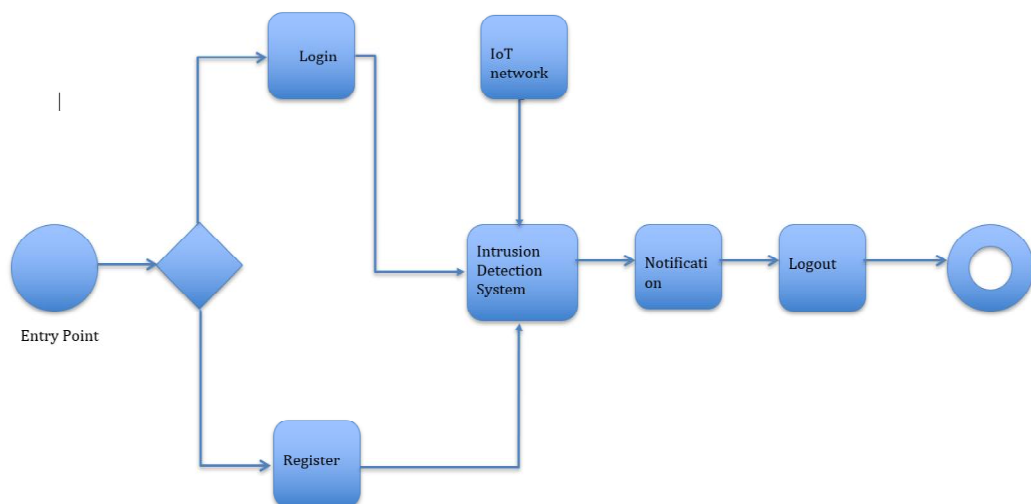
Fig 3.7 State Machine Diagram

3.6.7 Deployment Diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes.
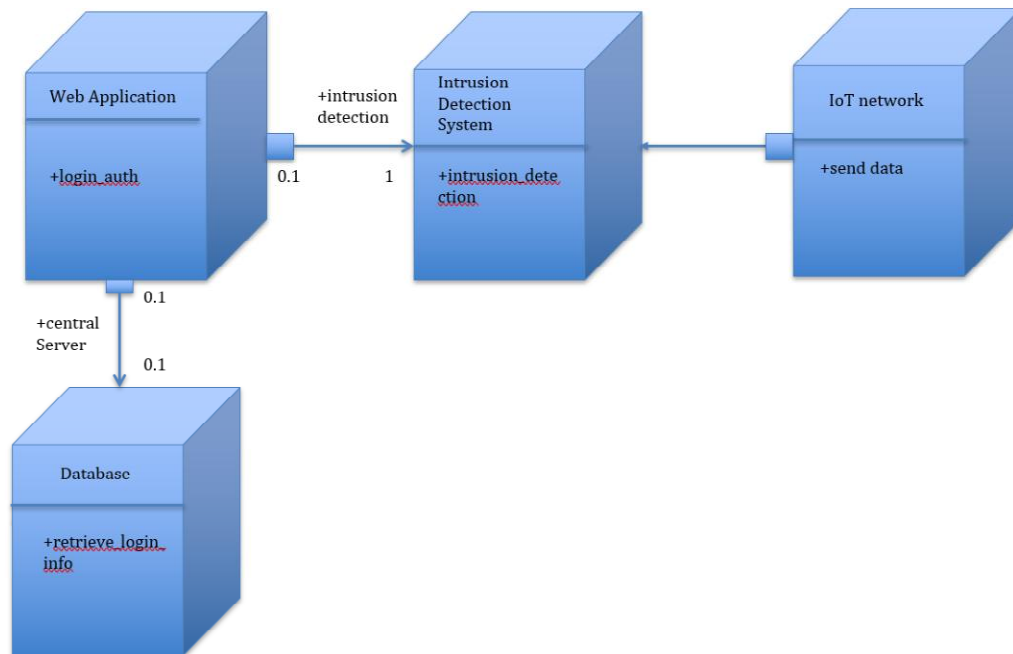
Fig 3.8 Deployment Diagram

3.7 IoT Node Connections Diagram
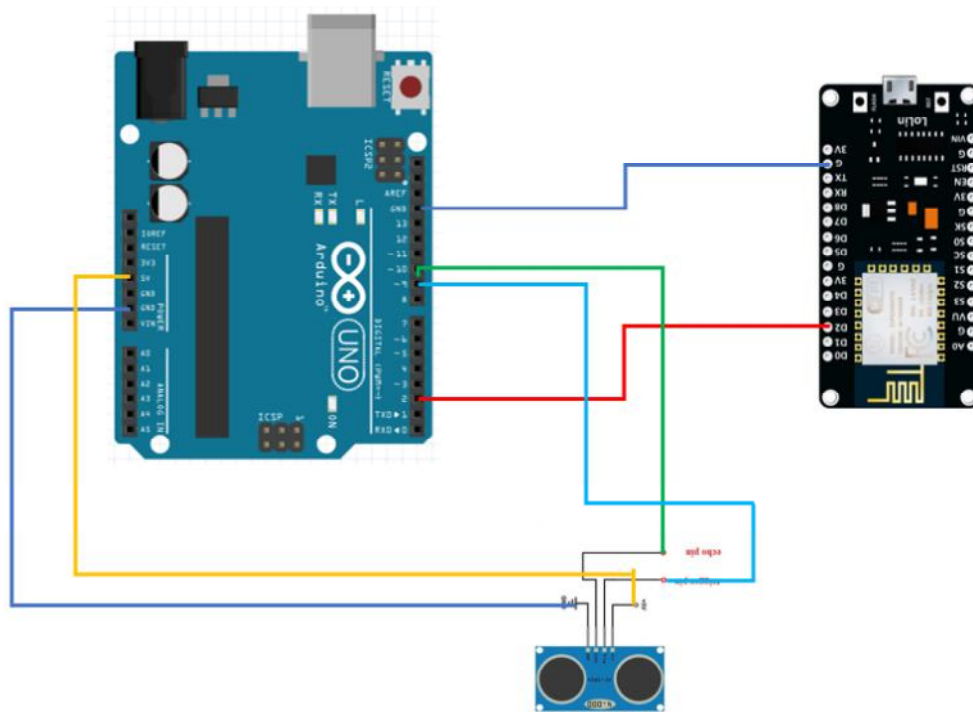
Fig 3.9 IoT Node Connections Diagram

Chapter 4
Implementation and Coding

4.1 Introduction

This chapter covers the role of various subsystems/modules/classes along with implementation details listing of the code for the major functionalities. The system consists of an IoT node (Arduino UNO, Node MCU, Ultrasonic sensor connected together), which sends distance data to the backend on the Apache server. This data is displayed on the GUI of the React.js front end. The network traffic is monitored by the TShark script and dumped in the file.csv. The Django Python backend reads this data in real time, feeds it to the ML model based on Random Forest and classifies it either as normal or attack. This data is sent to the front end and displayed in the form of a notification.

4.2 Module-wise Details

4.2.1 IoT Node

The IoT node consists of 3 devices – Arduino UNO, Node MCU, Ultrasonic sensor.

4.2.1.1 Ultrasonic sensor – Its end goal is to measure distance. It emits ultrasonic waves and receives them.

4.2.1.2 Arduino UNO – This controls the ultrasonic sensor, measures the time between sending and receiving the waves which is used to calculate the distance from the object using the formula-

$$distance= duration*0.034/2$$

4.2.1.3 Node MCU – The distance data is sent from Arduino to Node MCU using a serial communication. This is then sent to the php script on the Apache server by connecting to the wifi network.

4.2.2 TShark Script

TShark is a command line utility, a network packet analyzer that is used to capture packets from a live network and dump the traffic data into file.csv which is later read by the backend Python script. It captures the fields – frame number, frame time, frame length, source mac address, destination mac address, source IP address, destination IP address, IP protocol, IP length, TCP length, TCP source port, TCP destination port, packet info.

4.2.3 Apache Backend

The apache backend has 1 php script – server.php and 1 text file – datastorage.txt.

4.2.3.1 server.php – This php script is used to receive distance data from the IoT node and save it in the datastorage.txt.

4.2.3.2 datastorage.txt – This text file stores the 10 most recent distance values sent from server.php. The older values are deleted by server.php. This data is used by the front end to display graphics in Dashboard.js.

4.2.4 ML model

4.2.4.1 Dataset – The dataset needed to create the ML model was made using traffic data gathered from the TShark script. Multiple attacks (wrong setup, DDOS, Data Type Probing, Scan Attack, MITM) were performed while monitoring the traffic. The records were labeled according to known parameters used to identify particular type of attack.

4.2.4.2 Model Creation – The generated dataset was filtered for erroneous data in dataFilteration.py, resampling was done to deal with the issue of lesser attack data than normal data in resampling.py, then it was divided into training, testing datasets and a Random Forest classifier was trained based on training data in model.py. This was tested using the test data and later optimized. This model has been used in consumers.py in the Django backend for real time attack classification.

This can be better understood from the following sequence diagram.
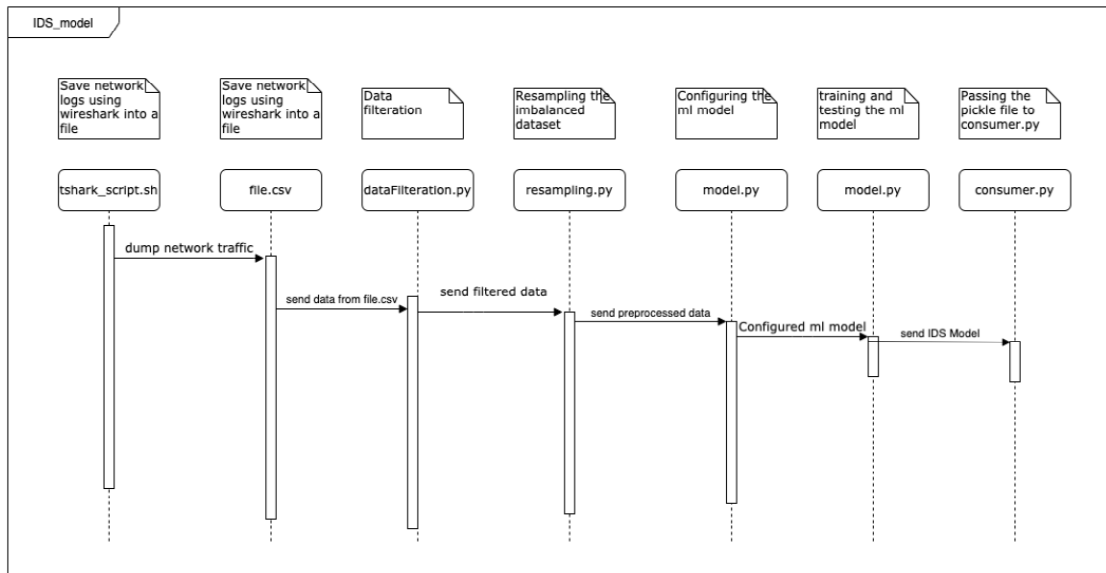
Fig 4.1 ML model module

4.2.5 Notifications

The notification module is responsible for displaying the notification to the user whenever an attack has been detected in the IoT network. A web socket (attackNotif ws) is created between the React.js front end and Django backend to send notification data from BE to FE whenever an attack is detected. The ws starts the real time read loop in consumers.py which takes the data from file.csv, processes it and constantly keeps feeding to the ML model. The model makes classification (normal or attack) in real time and sends a notification if an attack is detected. This notification is displayed in the front end. This can be better understood from the following sequence diagram.
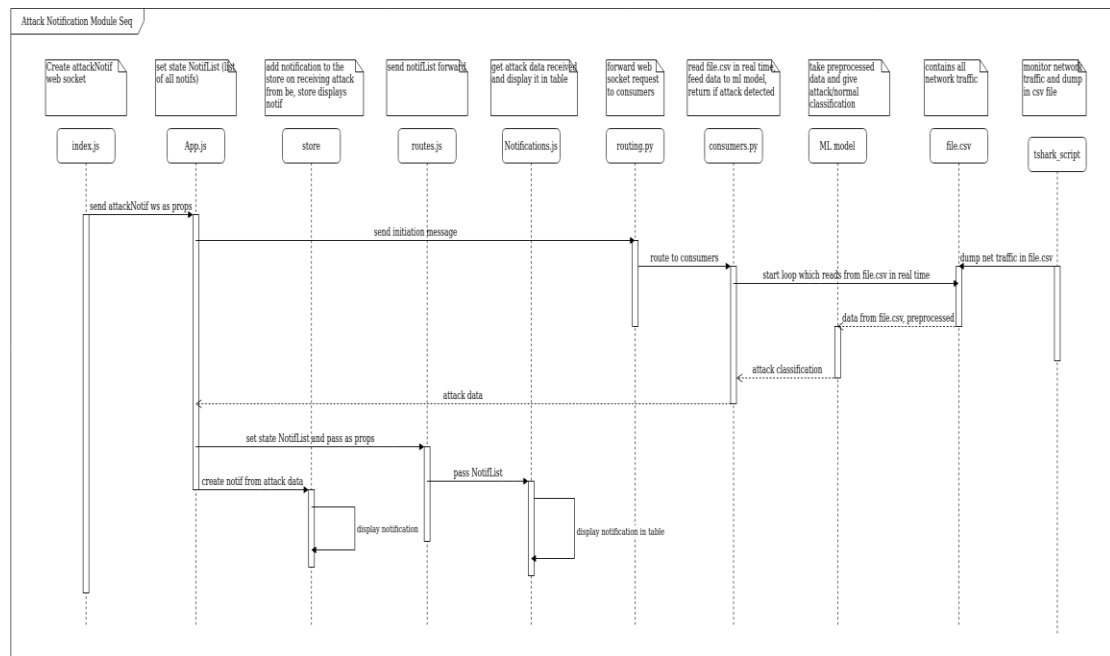
Fig 4.2 Notifications Module

4.2.6 Network Logs

The network logs module is used to display the traffic in the IoT network in real time. Similar to the notifications module, the web socket is created which initializes the loop in consumers.py that fetches traffic data from file.csv in real time. This data is sent to the FE where it is displayed. It updates in real time so the user can monitor IoT network traffic. This can be better understood from the following sequence diagram.
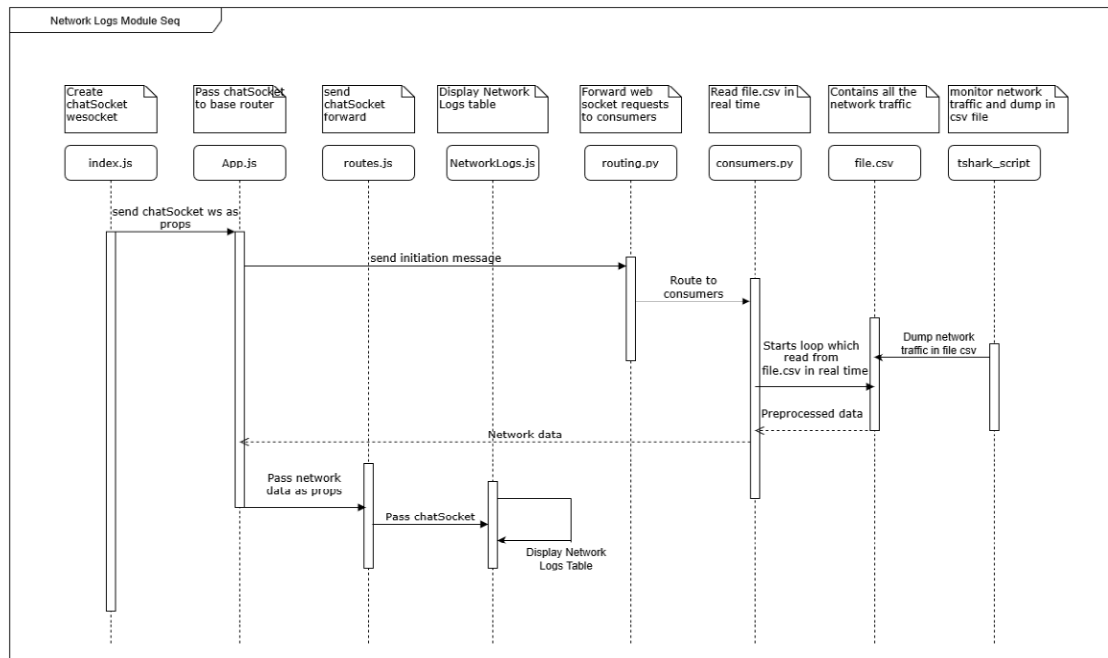
Fig 4.3 Network Logs Module

4.2.7 Dashboard

The dashboard module displays the distance data received from the IoT node with the help of graphics, thereby simulating the applications of the IoT network. A web socket is created between the FE and the php script websockets.php. An initiation message is sent which triggers the loop in the php script which reads the data from datastorage.txt and constantly sends this data to Dashboard.js where the node activity is shown. This can be better understood from the following sequence diagram.
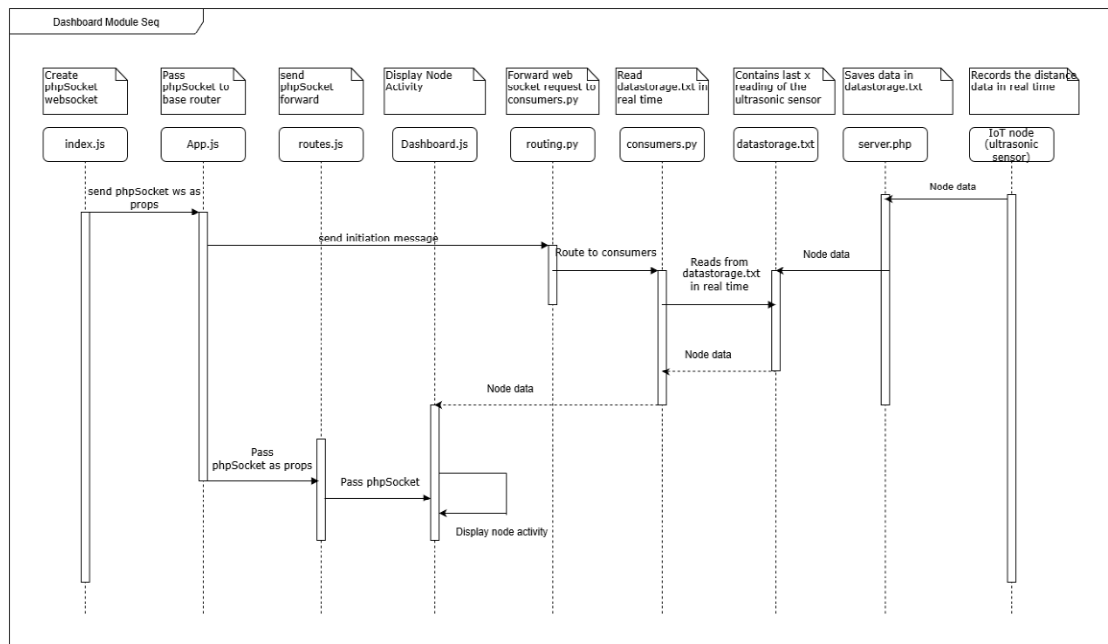
Fig 4.4 Dashboard Module

4.2.8 Visualisations

Visualisations modules showcases the data exploration, preprocessing and comparisons in results from the dataset and ML model. It has the visuals – ML model comparison, types of attack, types of protocol, correlation matrix, confusion matrix, effects of oversampling. The results were obtained in the python files dataFilteration.py, resampling.py, modelCompare.py, dataVisual.py and displayed in the visualisations module. This can be better understood from the following sequence diagram.
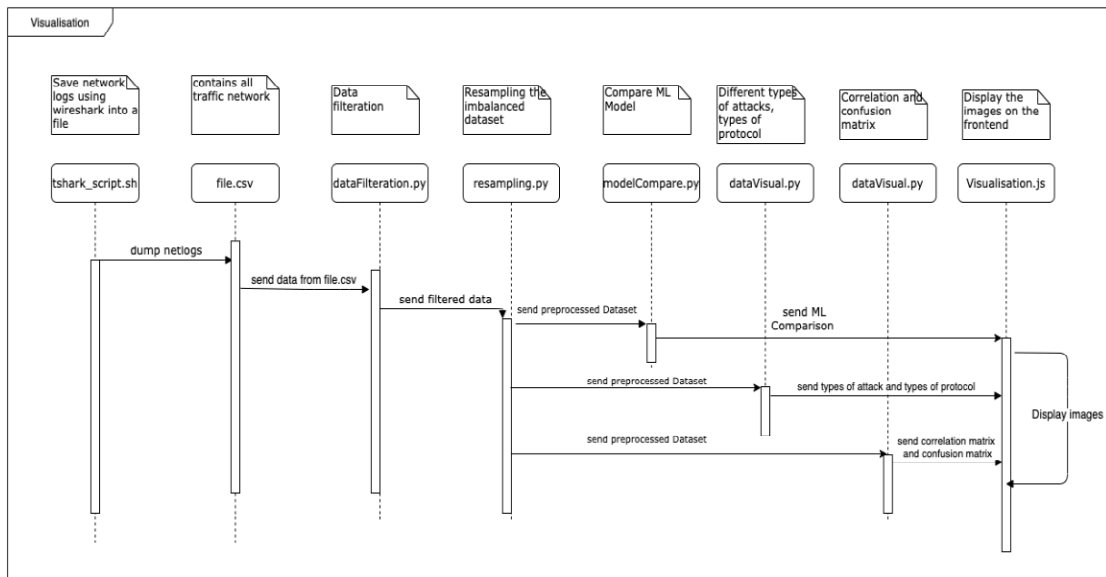
Fig 4.5 Visualisations Module

4.3 Code Listing

The code for this project is available on https://github.com/s3r-be. Following are important snippets of code from the modules of the project.

4.3.1 IoT Node

4.3.1.1 Arduino UNO

```
// send the ultrasonic wave
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microsecond
s
duration = pulseIn(echoPin, HIGH);

// Calculating the distance
distance= duration*0.034/2;
ArduinoUno.print(distance);
ArduinoUno.println("\n");
Serial.println(distance);
delay(100);
```

4.3.1.2 Node MCU

```
// connecting to wifi
WiFi.begin(ssid,password);
Serial.println();
Serial.print("Connecting");
while( WiFi.status() != WL_CONNECTED ){
    delay(500);
    Serial.print(".");
}
```

```
// send distance data to server.php
if (WiFi.status()==WL_CONNECTED)
{
    HTTPClient http;
    String url="http://" + ip + "/ids/server.php?data="+d;
    http.begin(url);
    http.addHeader("Content-Type","text/plain");
    int httpCode=http.GET();
    String payload=http.getString();
    Serial.println("While sending I received this from server : "+paylo
ad);
    if (payload=="SUCCESS. Data written in file."){
      sres=1;
    } else {
      sres=0;
      Serial.println("wrong or no payload!");
    }
    http.end();
    net=1;
}
```

4.3.2 TShark script

```bash
#!/bin/bash

echo s3rbeproj | sudo -S `#to get sudo permissions`\
tshark `#run tshark`\
-T fields -e frame.number -e frame.time -e frame.len -e eth.src -
e eth.dst `#give fields to capture`\
-e ip.src -e ip.dst -e ip.proto -e ip.len -e tcp.len -e tcp.srcport -
e tcp.dstport -e _ws.col.Info \
-E header=y -E separator="$" -E quote=n -
E occurrence=f `#set headers, separator as comma, double quotations, fi
rst occurrence`\
> file.csv `#dump into file.csv`\
-i wlp3s0 `#interface wifi`
```

4.3.3 Apache backend – server.php

```php
// receive data passed to the script
$val = $_GET["data"];
$fileContent=$val."\n";
$filename = "datastorage.txt";
// append data to datastorage.txt
$fileStatus=file_put_contents($filename,$fileContent,FILE_APPEND);
if($fileStatus != false)
{
        echo  "SUCCESS. Data written in file.";
        $file = file($filename);
        // remove first value (to control file size)
        unset($file[0]);
        file_put_contents($filename, $file);
}
```

4.3.4 ML Model

```python
for i in range(0,c):
    if(l[i].startswith("GET / HTTP/1.1 ")):
        num.append("-99")
    elif (l[i].startswith("GET")):   ###### wrong setup and data type pr
obing
        a = l[i].split("=")
        b = (a[1].split(" "))
        num.append(b[0])
    elif(l[i].startswith("Echo")):          ######### ddos
        num.append("-1")
    elif (l[i].startswith("Who")):            ############# scan
        num.append("-1")

    elif "duplicate " in l[i]:                ############# mitm
        num.append("-1")
    else:
        num.append("-99")
    i+=1
```

4.3.5 Notification

```python
# previous values to check if attack logging is being repeated
        prev_mac_source = ''
        prev_mac_dest = ''
        prev_prediction = 0
```

```python
        # receive message sent from front end
        text_data_json = json.loads(text_data)
        message = text_data_json['message']

        # Returns the Path your .py file is in
        workpath = os.path.dirname(os.path.abspath(__file__))
        logfile = open(os.path.join(workpath, "file.csv"), "r")

        # check if file.csv is populated, number of lines should be gre
ater than 1
        while(len([line for line in logfile]) <= 1):
            continue

        loglines = self.follow(logfile)  # create object of the generat
or
        # for loop runs the generator code in every iteration
        # once it reaches yield, it returns the line

        for line in loglines:
            returnLog = line  # storing in temp var to return as log
            line = line.split('$')
            # line contains data separated with $
            # the number of columns is 13
            # however some dataframes will show len < 13 because the da
ta is written incompletely by tshark
            # every few seconds 1 dataframe will be dropped

            if len(line) == 13:

                line[1] = line[1].split(' ')[3].replace(
                    ':', '').replace('.', '')  # pre time
                line[3] = int(line[3].replace(':', ''), 16)  # pre eth
src
                line[4] = int(line[4].replace(':', ''), 16)  # pre eth
dst

                try:
                    line[5] = float(line[5].replace('.', ''))  # pre ip
 src
                except Exception as ex:
                    line[5] = 0
                try:
                    line[6] = float(line[6].replace('.', ''))  # pre ip
 dst
                except Exception as ex:
                    line[6] = 0
                try:
```

```python
                line[7] = int(line[7])  # pre protocol
            except Exception as ex:
                line[7] = -1
            try:
                line[8] = int(line[8])  # pre ip length
            except Exception as ex:
                line[8] = 0
            try:
                line[9] = int(line[9])  # tcp length
            except Exception as ex:
                line[9] = 0
            try:
                line[10] = int(line[10])  # tcp source port
            except Exception as ex:
                line[10] = 0
            try:
                line[11] = int(line[11])  # tcp destination port
            except Exception as ex:
                line[11] = 0
            value = -99
            if(line[-1].startswith("GET / HTTP/1.1 ")):
                value = -99
            elif (line[-1].startswith("GET")):  # wrong setup and data type probing
                a = line[-1].split("=")
                try:  # if = hasn't been read, index 1 doesn't exist
                    b = (a[1].split(" "))
                    try:
                        # check if float data is sent, if string it
 is data type probing
                        value = float(b[0])
                    except Exception as ex:
                        value = -3
                except Exception as ex:
                    value = -99
            elif(line[-1].startswith("Echo")):  # ddos
                value = -2
            elif (line[-1].startswith("Who")):  # scan
                value = -4

            elif "duplicate " in line[-1]:  # mitm
                value = -5
            else:
                value = -99
            line[-1] = value
```

```python
                ip_df = pd.DataFrame([line[1:]])
                # get classification - attack or normal
                prediction = rf_model.predict(ip_df)[0]
                if prediction != 0:
                    # print('attack: ' + str(prediction))
                    # still a string to this point
                    returnLog = returnLog.split('$')

                    # check if attack is being repeated for same mac so
urce and dest
                    if not (prev_mac_source == returnLog[3] and prev_ma
c_dest == returnLog[4] and prev_prediction == prediction):

                        self.send(text_data=json.dumps({
                            'attack.type': attack_type[prediction],
                            'frame.number': returnLog[0],
                            'frame.time': returnLog[1],
                            'frame.len': returnLog[2],
                            'eth.src': returnLog[3],
                            'eth.dst': returnLog[4],
                            'ip.src': returnLog[5],
                            'ip.dst': returnLog[6],
                            'ip.proto': returnLog[7],
                            'ip.len': returnLog[8],
                            'tcp.len': returnLog[9],
                            'tcp.srcport': returnLog[10],
                            'tcp.dstport': returnLog[11],
                            '_ws.col.Info': returnLog[12]
                        }))
                        # set prev values -
 to prevent overloading of front end
                        prev_mac_source = returnLog[3]
                        prev_mac_dest = returnLog[4]
                        prev_prediction = prediction
```

4.3.6 Network Logs

```python
# receive request and start a loop to send network data back
    def receive(self, text_data):
        text_data_json = json.loads(text_data)
        message = text_data_json['message']

        # Returns the Path your .py file is in
        workpath = os.path.dirname(os.path.abspath(__file__))
        logfile = open(os.path.join(workpath, "file.csv"), "r")
```

```python
    # check if file.csv is populated, number of lines should be greater than 1
    while(len([line for line in logfile]) <= 1):
        continue

    loglines = self.follow(logfile)  # create object of the generator

    # for loop runs the generator code in every iteration
    # once it reaches yield, it returns the line

    for line in loglines:
        line = line.split('$')
        # line contains data separated with $
        # the number of columns is 13
        # however some dataframes will show len < 13 because the data is written incompletely by tshark
        # every few seconds 1 dataframe will be dropped

        if len(line) == 13:
            self.send(text_data=json.dumps({
                'frame.number': line[0],
                'frame.time': line[1],
                'frame.len': line[2],
                'eth.src': line[3],
                'eth.dst': line[4],
                'ip.src': line[5],
                'ip.dst': line[6],
                'ip.proto': line[7],
                'ip.len': line[8],
                'tcp.len': line[9],
                'tcp.srcport': line[10],
                'tcp.dstport': line[11],
                '_ws.col.Info': line[12]
            }))
```

4.3.7 Dashboard

```javascript
    d = d.split('\n', 10);
    // get first number from datastorage.txt
    d = Number(d[0]);
    // append to array
    movingAvg.push(d)
    if (movingAvg.length >= smoothingWindow) {
        // removes first value if length reaches smoothing window (eg 10)
```

```
        movingAvg.shift();
    }
    // get sum
    var tempAvg = 0;
    movingAvg.forEach(element => {
        tempAvg += element;
    });
    // get average
    tempAvg /= movingAvg.length;
    // condition for tempAvg going beyond threshold
    if (tempAvg > 1000) {
        tempAvg = tempAvg % 1000;
    }
    this.setState({
        pos: tempAvg
    });
```

4.3.8 Visualisations

```
<Button.Group>
    <Button size="mini" color='black' onClick={this.ModelComparison}>
Model Comparison</Button>
    <Button size="mini" color='black' onClick={this.ConfusionMatrix}>
Confusion Matrix</Button>
    <Button size="mini" color='black' onClick={this.Correction}>Corre
lation Matrix</Button>
    <Button size="mini" color='black' onClick={this.WithOverProtocol}
>Protocol(Oversampled)</Button>
    <Button size="mini" color='black' onClick={this.WithoutOverProtoc
ol}>Protocol</Button>
    <Button size="mini" color='black' onClick={this.OAttackDis}>Attac
ks(Oversampled)</Button>
    <Button size="mini" color='black' onClick={this.AttackDis}>Attack
s</Button>
    <Button size="mini" onClick={this.Disable}> Disable</Button>
</Button.Group>
```
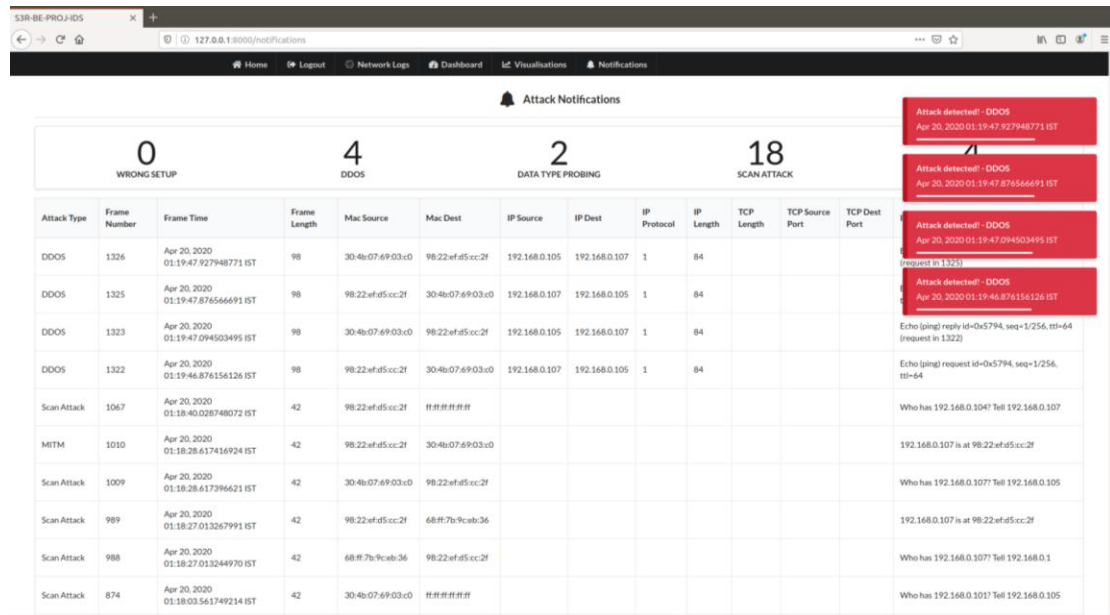
4.4 Screen shots

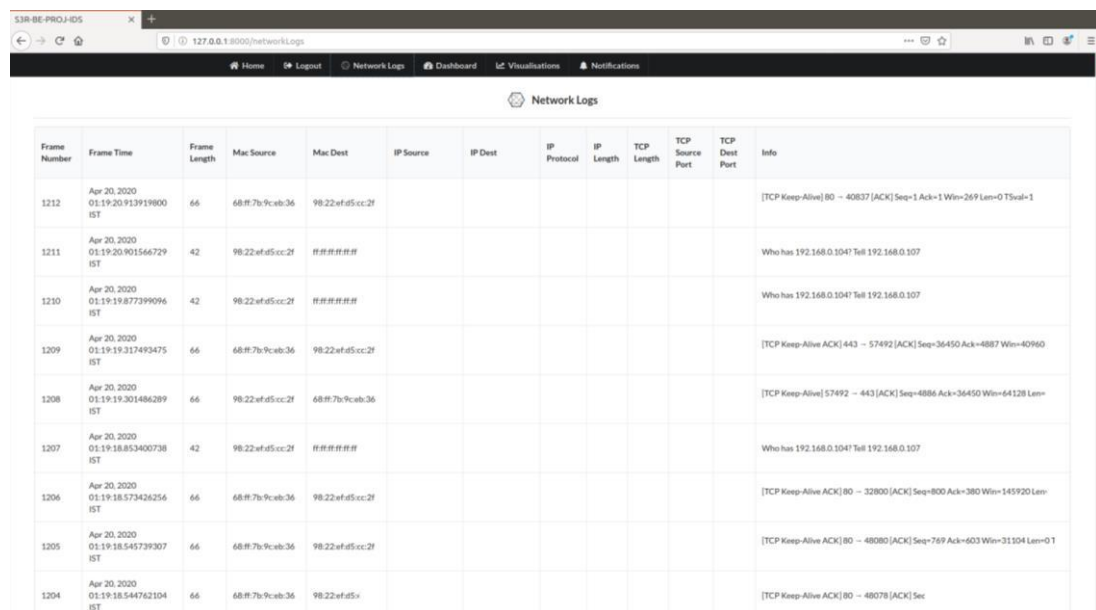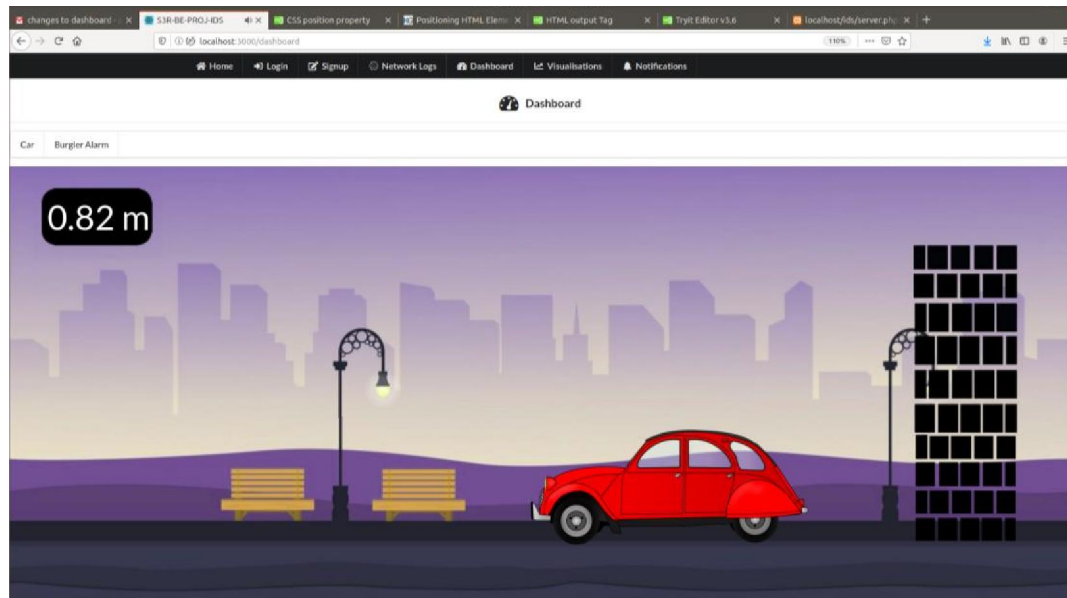**Fig 4.6 Notifications screen shot**

Fig 4.7 Network Logs screen shot
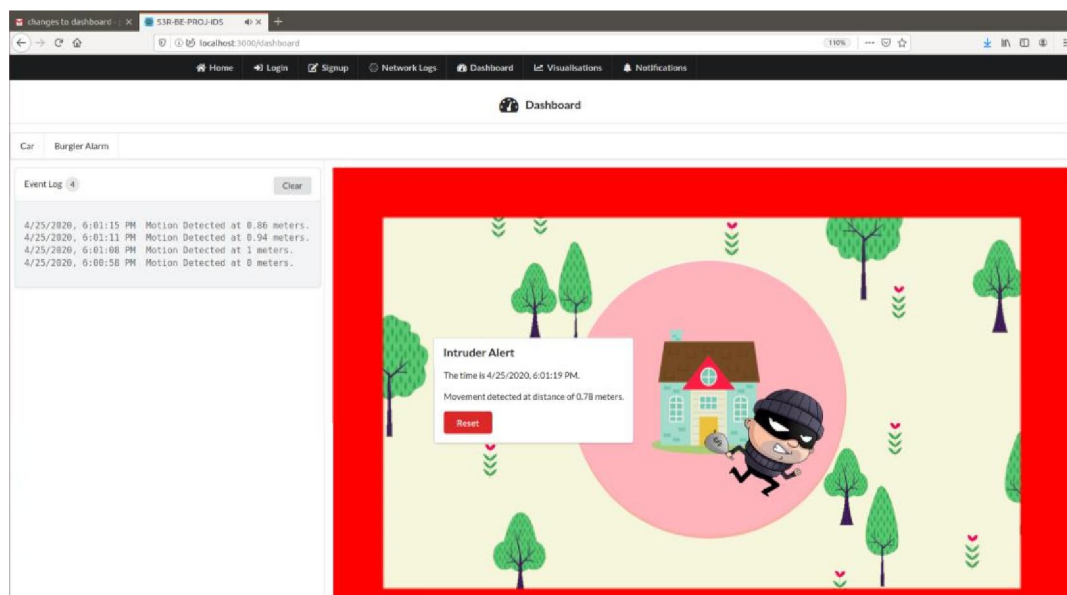
Fig 4.8 Dashboard car screen shot

Fig 4.9 Dashboard burglar screen shot
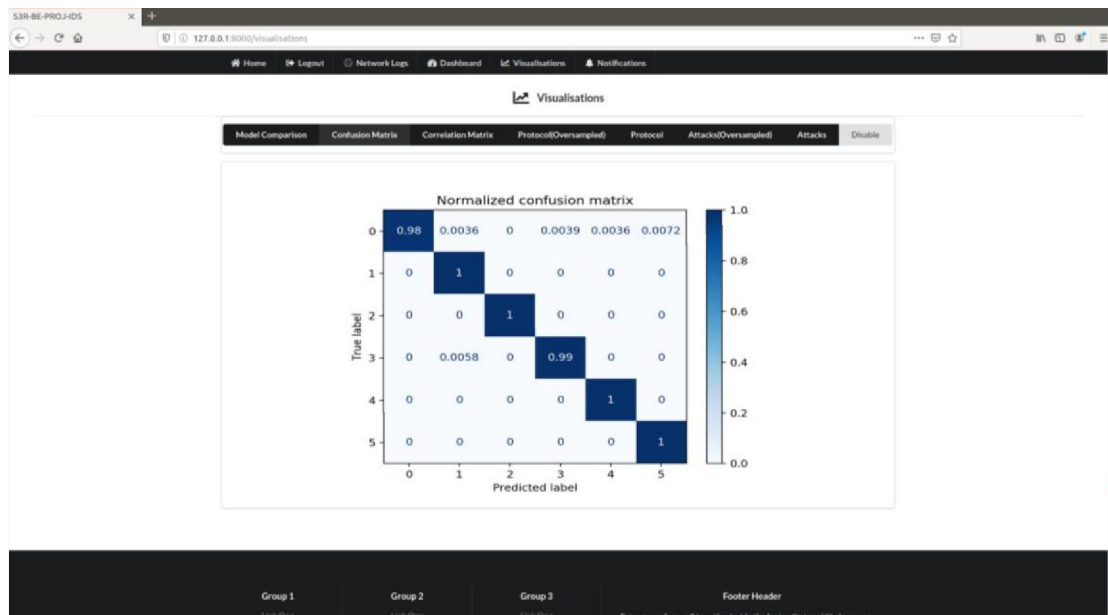
Fig 4.10 Visualisations screen shot

4.5 IoT Node Picture
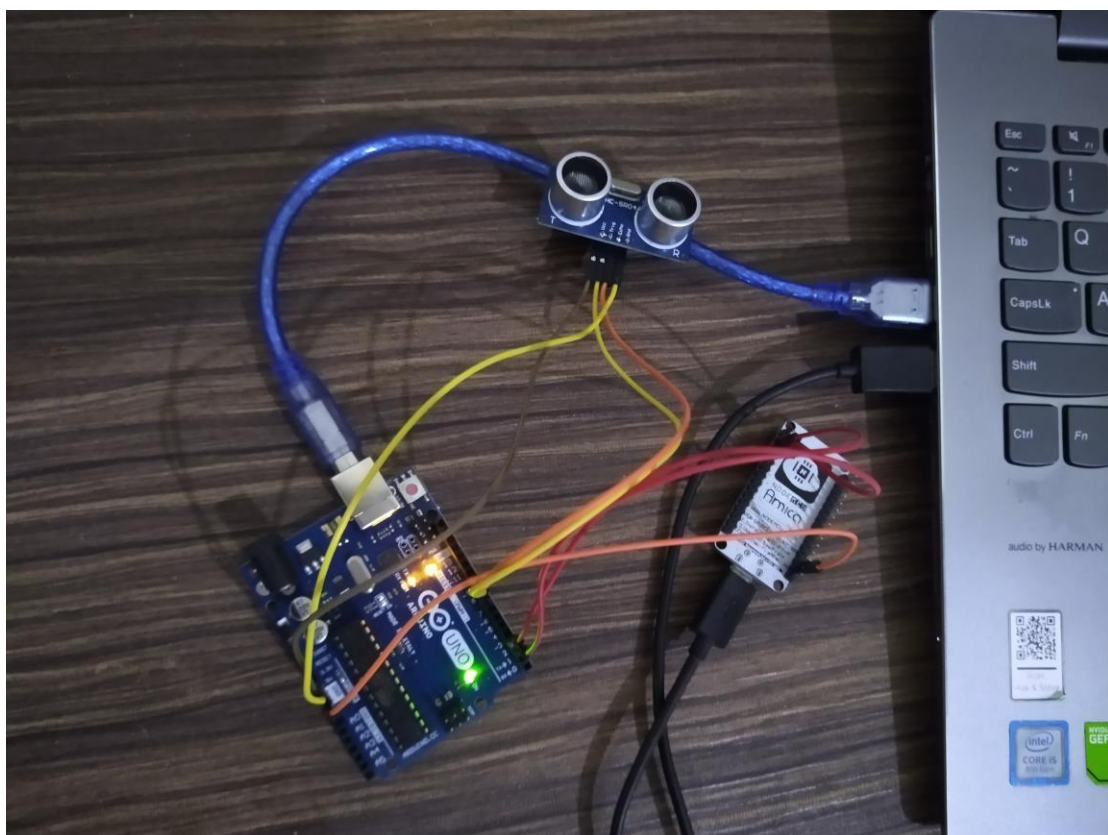
Fig 4.11 IoT Node Picture

Chapter 5
Testing

5.1 Introduction

This chapter covers the testing approach used and the test cases

5.2 Unit Testing

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

Table 5.1: Unit Testing 1

Title	Login and Authentication check
Test Item	Username, Password, Data packets
Input Specification	Test Item containing alphanumeric characters
Description	Specified details of Login/Register are entered and stored in database
Expected Results	Entry for Test Item is created in Database.
Actual Output	As expected

Table 5.2: Unit Testing 2

Title	Network Connection Check
Test Item	IOT devices, wifi module

Input Specification	Test Item containing alphanumeric characters
Description	Wifi (802.11) protocol is used for connection, communication and sending the sensed data to base station
Expected Results	Data sensed is sent to the server.
Actual Output	As expected

Table 5.3: Unit Testing 3

Title	Intrusion Detection System processing check
Test Item	Sensed based data of specified sensor
Input Specification	Test item containing various ongoing intrusions on IOT network
Description	Intrusion detection data is used to detect malicious attacks on the IOT network and ignore normal behavior of the system
Expected Results	Category of the attack is determined
Actual Output	As expected

Table 5.4: Unit Testing 4

Title	Notification Check
Test Item	Notification page of web application(GUI)
Input Specification	Test Item containing processed data
Description	Alert only for abnormal behavior of the system
Expected Results	Alert to the user on the notification page if malicious attacks are detected

Actual Output	As expected

5.3 Integration Testing

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Table 5.5: Integration Testing 1

Title	Integration of the IoT node with the system
Description	Data sent from the IoT node is captured in the react FE for display and Django BE for attack classification
Test Steps	Run the IoT node and the web app in the IOT network
Expected Results	IoT Data and Network data displayed on the front end
Actual Output	As expected

Table 5.6: Integration Testing 2

Title	Integration of all modules in web app
Description	Modules Notifications, Dashboard, Network Logs and Visualisations are added in the web app
Test Steps	Run the application program after integration
Expected Results	All modules work correctly i.e. Notifications show when attack is detected, dashboard shows node data, network logs are displayed and visualizations show graphs. The web app should not crash.

Actual Output	As expected

5.4 Acceptance Testing

Acceptance testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

Table 5.7: Acceptance Testing 1

Title	Sensed Availability data
Description	Ultrasonic sensor is used to sense data and is processed to gather sensed availability data in the database
Expected Output	Sensed Availability data is sent to the base station using the wifi protocol
Actual Output	As expected

Table 5.8: Acceptance Testing 2

Title	Accurate Intrusion Detection
Description	The attack should be correctly detected by the web app whenever it happens in the network
Expected Output	Correct detection of attack in real time functioning of the system is achieved
Actual Output	As expected

Chapter 6
Results and Discussions

6.1 Table of the results/findings or graphs
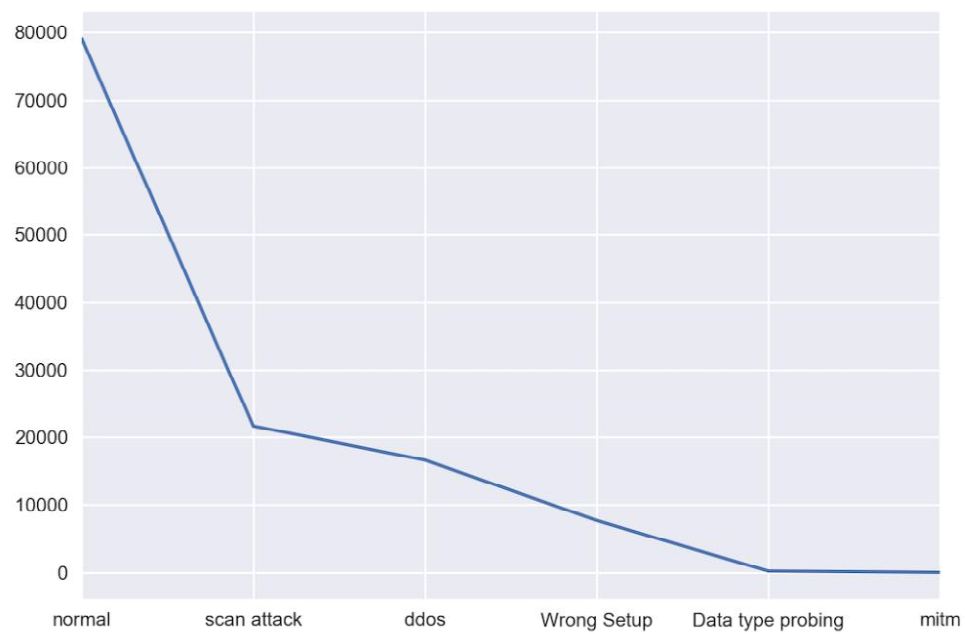
Fig 6.1 Attack distribution

Fig 6.2 Attack distribution post oversampling

Fig 6.3 Comparison of ML models

6.2 Discussions

The dataset created had a lower number of records labeled as attack as compared to the records labeled as normal as can be seen in fig 1. This issue was resolved by the oversampling done, the results of which can be seen in the fig 2. The Random forest ML model has bee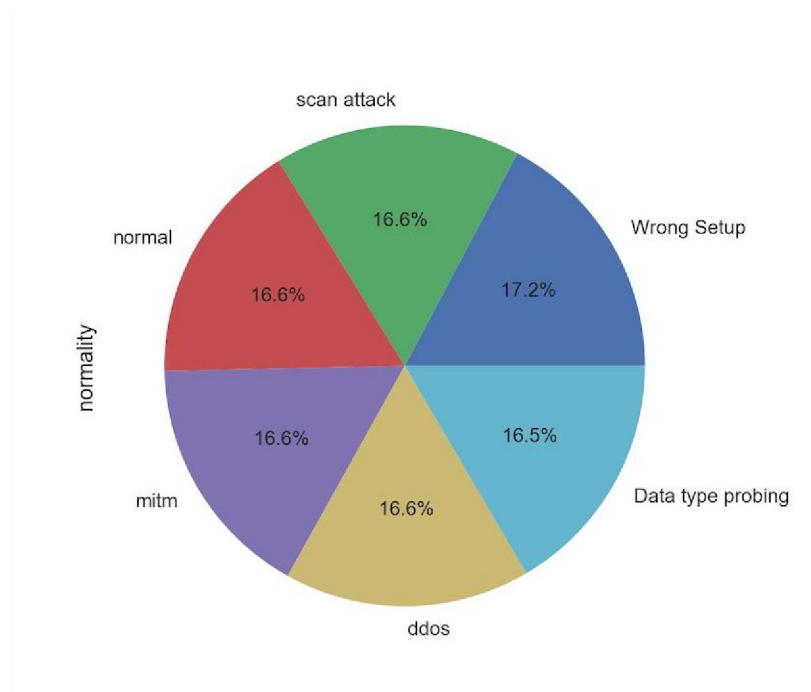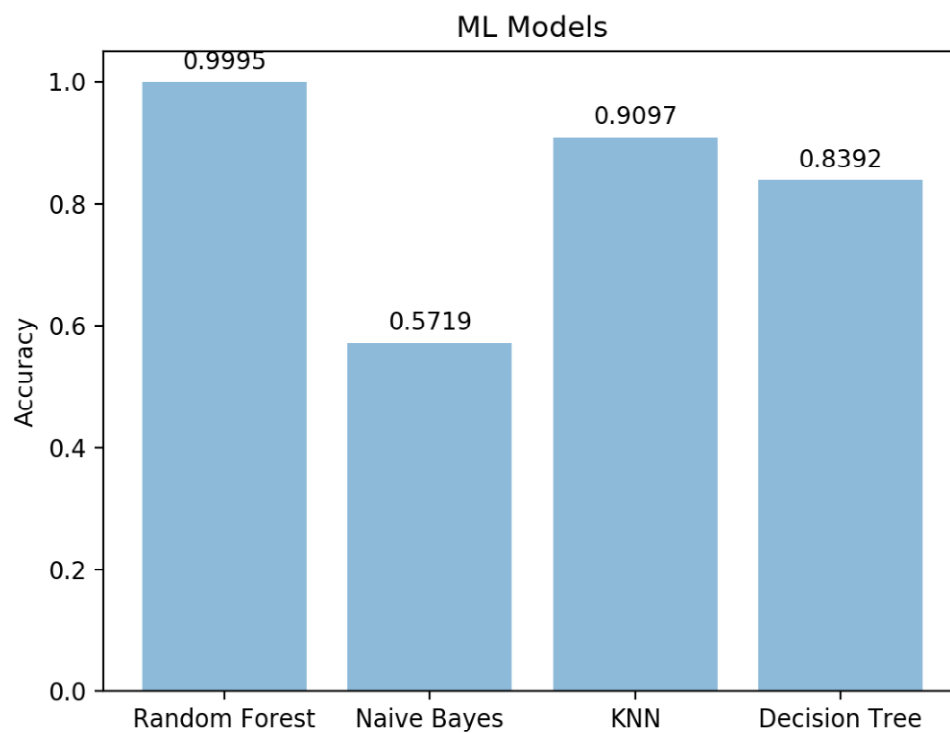n proven to be more accurate than other models in comparison – Naïve Bayes, KNN and Decision Tree with an accuracy of 99.95% as compared to the 57.19%, 90.97%, 83.92% respectively as can be seen in fig 3. Although the accuracy seems very high, it has in fact been overfitted on the dataset. This means that even if it shows very high accuracy on training set, it performs poorly on new data. This has given rise to a higher rate of false alarms in the real time working of the IDS. Although this can be an issue, the advantage of this is that even if it has false alarms, it does catch all of the intrusions that occur, thereby giving high true positive rates.

Chapter 7
Conclusion

Intrusion detection systems act as a second line of defense after the firewall and are beneficial for the security of IoT networks. This system has been developed so that can be used in real time IoT networks to improve their security. Using this system, the user can fetch their IoT network data, monitor network traffic, get notified when an intrusion is detected in the network. This has been achieved using the ML model Random Forest. The reduced latency of the system which makes it real time is achieved by web sockets. Since this system is generalized, it can be used for different types of IoT networks to detect intrusions in their network. The model shows a high level of accuracy 99.95%, it correctly classifies most of the attacks in real time, however it suffers from a high rate of false alarms, which is common with these types of models. More work can be done in the future to reduce the false alarm rates and make the model more robust.

References

Journal Article/ IEEE Paper

[1] Abebe A. Diro., Naveen C., "Distributed attack detection scheme using deep learning approach for Internet of Things", Future Generation Computer Systems, 82, 2017, 761-768.

[2] Ashima C., Brian L., Sheila F., Paul J., "Host based Intrusion Detection System with Combined CNN/RNN Model", IWAISe 2018- ECML PKDD Conference, 2018.

[3] Ebelechukwu N., Andre C., Gedare B., "Anomaly-based Intrusion Detection of IoT Device Sensor Data using Provenance Graphs", 1st International Workshop on Security and Privacy for the Internet-of-Things (IoTSec), 2018.

[4] Shadi A., Monther A., Muneer B. Y., "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model", Journal of Computational Science, 25, 2018, 152-160.

[5] TagyAldeen M., Takanobu O., Takayuki I., "Towards Machine Learning Based IoT Intrusion Detection Service", Recent Trends and Future technology in Applied Intellignece, 2018, 580-585.

Type of manuscript

Survey on anomaly based intrusion detection systems in IoT networks

[#1]Rhishabh Hattarki, [#2]Shruti Houji, [#3]Sanika Patil, [#4]Sahil Dixit, [#5]Manisha Dhage

[1]rhish9h@gmail.com,
[2]shrutihouji98@gmail.com,
[3]patilsanika02@gmail.com,
[4]sahildxgeneration10@gmail.com,
[5]mrdhage.scoe@sinhgad.edu

[#12345]Computer Department, Sinhgad College of Engineering,
Pune, Maharashtra, 411041, India

ABSTRACT

ARTICLE INFO

IoT devices are becoming popular day-by-day. Several vulnerabilities in IoT present the need for IoT security. The number of attacks on these devices keep increasing and most of them are slight variations of the previously known attacks, which can bypass the conventional firewall systems.

The existing systems are not suitable for IOT devices as IOT devices have low computational power. Those that use signature-based intrusion detection work only on known patterns and attacks, hence they cannot recognize newer attacks with unknown pattern. Also, many systems use cloud computing, which has a downfall that it needs access to internet at all times and are most often paid.
The following survey discusses a few of the anomaly based intrusion detection systems built to tackle this situation. It shows the different methods used along with their drawbacks or future scope. Also, an anomaly-based detection system has been proposed. It comes into effect when detecting newer attacks, that are not filtered by the firewall. It is capable of handling newer/unknown attacks, which signature based cannot deal with. It would be set up on a local higher powered device rather than on cloud.

Keywords— Intrusion detection, Internet of Things, Network security, Machine learning

I. INTRODUCTION

The advancement in computer technologies such as mobile and pervasive computing, internet applications and services has led to the proliferation of IoT (Internet of Things) devices. [1] IoT consist of devices such as smart devices, vehicles, home appliances, industrial smart devices that contain electronics, software, sensors, actuators, and connectivity which allows these things to connect, interact and exchange data.

With myriads of devices generating, processing and exchanging a vast amount of data that consists of safety, security and privacy critical information, it is bound to attract the attention of cyber criminals. The security issues are often known to the device manufacturers. However, a large chunk of the development time and effort is dedicated into getting the product to the market prior to the deadline. Hence, the security in IoT devices is either neglected or left out. As a result IoT devices become an easier target for hackers. [2]. Using the vulnerabilities in IoT design, software and hardware, the adversaries can plant malicious code, install a virus, etc. and infiltrate the IoT network.

When an IoT network is compromised, the intrusion detection system works as a second line of defence after the firewall. If an attacker breaks into a system or system server by forwarding malicious packets to the user system, which is used to steal, modify or corrupt any private important information, it is said to be an Intrusion. The intrusion detection system monitors the network traffic for suspicious activity using either misuse or anomaly detection systems. In the misuse detection system, the suspicious activity is compared to an existing set of behavior patterns whereas in anomaly detection system, intrusions are detected by checking for irregularities from the normal network traffic. [3]

An anomaly, also referred to an outlier, is data that deviates from the normal system behavior, which is learned by the detection system by the provided set of accepted network behavior data. Anomalies indicate that the system has a fault or that a malicious event has compromised the system. [4] Although it is prone to high false alarm rates, where a particular activity is considered as a threat even when it isn't, Anomaly based detection system can be preferred over signature based methods as it allows the detection of unseen attacks, whose signatures do not exist with the system.

II. IOT SECURITY REQUIREMENTS

1. Data privacy, confidentiality and integrity:

Data in IoT networks travel through multiple nodes/devices before reaching the target destination. Hence, a good encryption mechanism is required so that the data is not seen by the anyone other than the rightful owner, that is it remains confidential. A compromised node can violate the privacy of the data stored on the devices in the network. The vulnerable devices can have their data tampered with in a malicious way by the attacker thereby affecting data integrity.

2. Authentication, authorization and accounting:

The authentication is needed between 2 parties communicating with one another to secure communication in IoT. The devices must prove that they're the designated party and not an unwarranted adversary. The main goal of authorization mechanisms is to confirm that the access to systems or information is provided to the licensed ones. Implementing authentication and authorization ensures trust and security while communicating. The accounting for resource usage, along with auditing and reporting gives a reliable mechanism for securing network management.

3. Availability of services:

One of the most commonly implemented and tested attack, denial of service (DOS) is an attack where the attacker tries to deny service to the user by flooding the network with unnecessary traffic. This results in unavailability of services and eventually reduces quality of service (QOS).

4. Energy efficiency:

IoT devices are typically cursed with lack of resources like power and storage. They are lightweight and are susceptible to attacks that can leverage this vulnerability by flooding the devices' resources eventually leaving them unfit to use. Hence, security measures need to be energy efficient that can run on these low powered, low storage devices.

5. Single points of failure:

The explosive increase of IoT networks worldwide pose the issue of a huge number of single points of failure. This problem ruins the experience of IoT users that would have to deal with frequent device failures in their networks. This gives rise to the necessity of having implementations and measures that will provide robust networks. [5]

III. RELATED WORK

Abebe A. D. and Naveen C. have proposed a distributed rather than the centralized deep learning based IoT/Fog network attack detection system. [6] The distribution reduces the performance overhead from the individual IoT nodes which can be very helpful. The results of the distributed network show it to be better than the centralized network. However, the performance comparison of deep model vs shallow model is insufficient as only one shallow model is compared and no ensemble models are tested.

Ashima C. et. al. propose a CNN-GRU language model for the recently released ADFA-LD intrusion detection data set. [7] It uses the newer dataset; and Gated Recurrent Units rather than the normal LSTM networks to obtain a set of comparable results with reduced training times. However, it is an implementation of neural networks and it cannot match the performance of ensemble methods (stated in their conclusion).

Ebelechukwu N. et al. propose an anomaly detection algorithm for detecting anomalous instances of sensor based events in an IoT device using provenance graphs. [4] Provenance provides a comprehensive history of activity performed on a system's data. In this approach, an observed provenance graph is compared to an application's known provenance graph in order to detect the anomalies. This method has insufficient experimental data to prove it is better than other methods.

The hybrid efficient model proposed by Shadi A. et. al. [3], an ensemble consisting of J48, Meta Pagging, Random Forest, REPTree, AdaBoostM1, Decision Stump and Naïve Bayes is an anomaly based intrusion detection system. It is trained and tested on the NSL-KDD dataset and outperforms other simpler single machine learning models like the J48, SVM and Naïve Bayes in terms of accuracy in detecting the attacks.

Tagy Aldeen Mohamed ,Takanobu Otsuka and Takayuki Ito [8] proposed IDS based on machine learning techniques to be implemented into IOT platforms as a service. Random Forest is used as a classifier to detect intrusions. Also neural network classifier is used to detect the categorization of the detected intrusions. The experimental results showed the proposed model can efficiently use Cloud computing service to process data from different resources of IOT platforms.

GARUDA introduces a novel distance measure that can be used to perform feature clustering and feature representation for efficient intrusion detection. [9] GARUDA has better detection rates for U2R and R2L attack types, however it is more focused on feature reduction and dissimilarity measure used in the classifiers. Moreover, it did not improve classification accuracies on SVM.

Bayu A. T. and Kyung H. R. have presented an effective

anomaly detection by incorporating PSO-based feature selection and random forest model. [10] It uses a random forest model with particle swarm optimization-based feature selection. It focuses on feature selection and compares performance with only 2 other models.

Anomalous Payload-Based Network Intrusion Detection by Ke Wang and Salvatore J. Stolfo [11] have presented a detector called PAYL which establishes a base profile of the system using a byte frequency distribution, then detecting the anomaly by the standard deviation from the base using the Mahalanobis distance. The model improves on the false alarm rates and it is an incremental, unsupervised model. However, the dataset used, 1999 DARPA IDS dataset is fairly old and there is scope for more testing in live environments.

Belal S. K. et. al. [12] have implemented a fog computing based IDS for IoT using a Multilayer Perceptron (MLP) model. The fog layer sits between the cloud and the IoT network and improves on latency, mobility, user experience and geographical distribution. They have used the newer datasets ADFA-LD and ADFA-WD and raspberry pi as the fog node. This lightweight system addresses some of the issues from cloud based systems, although, they still have room for improvement in terms of accuracy and efficiency.

Aymen Yahyaoui et al. [13] have come up with a heirarchical solution for IoT Intrusion detection. They've used support vector machine for WSN intrusion detection and deep learning for detecting IP intrusions at the gateway. Historical data is used to determine the probability of getting an attack and SVM runs on the nodes with high probability of attack. This saves on energy which is critical in IoT networks. This study only covers few types of attacks and can be extended to detect more types of IoT attacks.

Veeramreddy J. and Koneti M. P. [14] proposed an Anomaly-Based IDS to develop generic meta-heuristic scale for both known and unknown attacks with a high detection rate and low false alarm rate by adopting efficient feature optimization techniques using the popular NSL-KDD dataset. Though existing intrusion detection techniques address the latest types of attacks like DoS, Probe, U2R, and R2L, reducing false alarm rate is a challenging issue with high classification accuracy.

Sébastien J.J. G., Alvin K.H. L., Craig O. F., et al. [15] have presented an anomaly based intrusion detection system using time based histogram comparison for the MIL-STD-1553 protocol (commonly used in military aircrafts). Baseline histograms are created for the time based features used in the considered protocol. They are compared with either the real time data or stored data histograms to determine the activity as anomalous or normal. It's a more basic approach compared to the modern techniques like machine learning, however it shows promising results for their requirements. There is room for improvement in terms of including multiple aircraft modes, using parallel processing for improved speed, adding additional features.

S. Venkatraman & B. Surendiran [16] have proposed a hybrid IDS for home IoT networks which includes the combination of signature based as well as anomaly based systems. The signature based IDS uses a crowd sourced framework for the pattern set. This hybrid proves to be more accurate than the systems running individually. However, this can be tested more extensively and compared with more similar IDSs, more number of IoT devices can be used.

Luis M.T, Eduardo M., Mikel I. et al. [17] proposed an anomaly-based IDS for IEEE 802.11 networks introducing a wireless IDS called S2WIDS. It is an anomaly based system that implements a multidisciplinary approach to detect the most common attacks in wireless environments and it may be able to fight some of the new threats that might arise in the future. However, it does not provide additional information about the AP (unauthorized wireless access point) to improve the corresponding part of the detection engine.

Ayyaz-ul-Haq Q. (B), Hadi L., Jawad A. et al. [18] have developed a heuristic intrusion detection system for IoT using a random neural network (RNN). The dataset used is NSL-KDD and the algorithm used is gradient descent. The dataset was reduced using feature selection and then tested using different learning rates. Highest accuracy was achieved at lower learning rates. The model faired well when it was compared to different models. This system can use more testing and implementation in an actual IoT network.

IV. PROPOSED METHODOLOGY

The system has been inspired by the already existing intrusion detection systems which attempt to protect the home-network against attacks and intrusions. We are taking these pre-existing models and combining them to get additional advantages and reduce the downsides as much as possible. Till date, there have been very few attempts at making an IDS for IOT devices. Most of the IDS present in the market cater to non-IDS networks. We are taking the principles of these IDS systems and modifying them as per requirement of an IOT network.
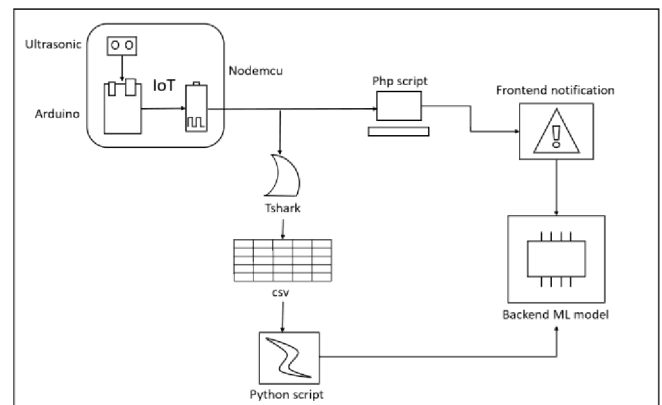

Fig 1. System Overview Diagram

The proposed system is a web app that constantly runs in the background. The IDS as shown in the Fig 1. Sits between the firewall and the IoT network.

A. Modules

1) User authentication

The intrusion detection system will get access only to the authorized personnel. Hence, the web app will have a user authentication system (login system).

2) Data connection establishment

Once the system has been started, it needs to check whether all the devices have been connected properly. If not, and error message will be sent to the front end.

3) Intrusion detection analysis

The machine learning model random forest will be used for the intrusion detection. The network data that is being monitored at run time will be preprocessed and sent to the model. This data will be analyzed here and classified as either normal or anomalous.

4) Notification

If an intrusion is detected, a notification will be sent to the user. The user will then be able to see more details if required.

B. Mathematical model

Let S be the solution set for the given problem statement. S={Input,Function,Output, Terminate,Success,Failure}. Where,Input =Input to the System. Function =Functions of the system. Output =Output of the system. Terminate= Terminating Condition of the System. Success =Success cases for the System. Failure =Failure cases for the system.

1. Input ={UserName,Password,Data Packets}
   a. UserName :-use rid
   b. Password : user passsword
   c. Data Packets:- A data packet is a unit of data made into a single package that travels along a given network path.

2. Function={Login auth,Network connection,train test,intrusion detection,Notification}
   a. Login auth: This function will take Username and Password as the input and gives the authorization rights to the user accordingly.
   IF Y=F(X) is the login auth function then
   X:- Username and Password taken as an input.
   Y:- Authorization rights of a user.
   b. Network connection =This function will take port number, ip address as an input to give the status of the connection network.
IF Y=F(X) is the Network Connection Function, then
   X:- Inputs to setup a network, connecting to a device ,routing the incoming data and device configuration.
   Y:- Status of the network connection.
   c. train test: This function will take features as an input to give the accuracy of the model and further it will be turned to improve the accuracy of the model. Random Forest has been used as a Machine learning algorithm for building the model.

IF Y=F(X) is the function to test and train the model then,
X:- Features of the DS2OS dataset as an input.
Y:- Accuracy of the model
   d. intrusion detection : This function will take features like sourceID, sourceAddress, sourceType, sourceLocation, destinationServiceAddress, destinationServiceType, destinationLocation, accessedNodeAddress, accessedNodeType, operation, value, timestamp as an input and notifies the user the system for the intrusion.

C. Algorithm

Random Forest Classifier has been used for the building and implementing the model.
Basically, a random forest is an average of tree estimators. As with non-parametric regression, simple and interpretable classifiers can be derived by partitioning the range of X.

Let n = A1, . . . , AN be a partition of X. Let Aj be the partition element that contains x. Then h(x) = 1 if XiAj Yi XiAj (1 Yi) and h(x) = 0 otherwise.

We conclude that the corresponding classification risk satisfies R(h) R(h) = O(n1/(d+2)).
Trees are useful for their simplicity and interpretability. But the prediction error can be reduced by combining many trees.

These are bagged trees except that we also choose random subsets of features for each tree. The estimator can be written as

$$m(x) = \frac{1}{M}\sum_j m_j(x)$$

where mj is a tree estimator based on a subsample (or bootstrap) of size a using p randomly selected features. The trees are usually required to have some number k of observations in the leaves. There are three tuning parameters: a, p and k. You could also think of M as a tuning parameter but generally we can think of M as tending to . For each tree, we can estimate the prediction error on the un-used data. (The tree is built on a subsample.) Averaging these prediction errors gives an estimate called the out-of-bag error estimate. IF Y=F(X) is the function then,

X: sourceID, sourceAddress, sourceType, sourceLocation, destinationServiceAddress, destinationServiceType, destinationLocation, accessedNodeAddress, accessedNodeType, operation, value, timestamp as an input
   Y: Notifies the system.
   e. Notification: This function takes the input which is provided by the function (intrusion detection) and notifies the user.
   IF Y=F(X) is the function then,
   X: Output of the function(intrusion detection). Y: Intrusion message to the user.

3. Output = {display intrusionmsg }
   a. display intrusionmsg : display error message if any intrusion occurs.

4. Intermediate Results

a. Successful working of module.
b. Successful Working of Network.
c. Successful User authentication.

5. Terminate= {Invalid details, Network failure, Timeout
   a. Invalid User Authentication.
   b. Network failure
   c. timeout

6. Success
   a. Successful user login.
   b. Successful connection establishment of nodes and ids.
   c. Successful detection of intrusion.
   d. Displaying the results.
   e. Appropriate error messages in case of invalid input.

7. Failure
   a. Web app Failure.
   b. Hardware faults.
   c. Network establishment failure.
   d. Not displaying required results.

V. CONCLUSION

Intrusion detection systems act as a second line of defense after the firewall and are beneficial for the security of IoT networks. The paper provides a survey of few of the different types of intrusion detection systems, mostly anomaly based systems. A system has been proposed using the pointers from the survey to detect intrusions in IoT networks. This can be used in a real time IoT networks to improve its security.

ACKNOWLEDGEMENT

REFRENCES

[1] Jesus P., Salim H., "IoT Security Framework for Smart Cyber Infrastructures", IEEE 1st International Workshops on Foundations and Applications of Self* Systems, 2016, pp. 242-247

[2] Jacob W., Khoa H., Orlando A., Ahmad-Reza S. et. al., "Security analysis on consumer and industrial IoT devices", 21st Asia and South Pacific Design Automation Conference (ASP-DAC), 2016

[3] Shadi A., Monther A., Muneer B. Y., "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model", Journal of Computational Science, 25, 2018, 152-160.

[4] Ebelechukwu N., Andre C., Gedare B., "Anomaly-based Intrusion Detection of IoT Device Sensor Data using Provenance Graphs", 1st International Workshop on Security and Privacy for the Internet-of-Things (IoTSec), 2018.

[5] Minhaj A. K., Khaled S., "IoT security: Review, blockchain solutions, and open challenges", Future Generation Computer Systems, 82, 2018, 395-411

[6] Abebe A. Diro., Naveen C., "Distributed attack detection scheme using deep learning approach for Internet of Things", Future Generation Computer Systems, 82, 2017, 761-768.

[7] Ashima C., Brian L., Sheila F., Paul J., "Host based Intrusion Detection System with Combined CNN/RNN Model", IWAISe 2018- ECML PKDD Conference, 2018.

[8] TagyAldeen M., Takanobu O., Takayuki I., "Towards Machine Learning Based IoT Intrusion Detection Service", Recent Trends and Future technology in Applied Intellignece, 2018, 580-585.

[9] Shadi A. A., Radhakrishna V., "GARUDA: Gaussian dissimilarity measure for feature representation and anomaly detection in Internet of things", The Journal of Supercomputing, 2018, pp 1-38

[10] Bayu A. T., Kyung-Hyune R., "An Integration of PSO-based Feature Selection and Random Forest for Anomaly Detection in IoT Network", MATEC Web of Conferences, vol 159, 2018

[11] Ke W., Salvatore J. S., "Anomalous Payload-Based Network Intrusion Detection", RAID 2004, LNCS 3224, pp. 203–222, 2004

[12] Belal S. K., Ainuddin W. B. A. W., Mohd Y. I. B. I., et. al., "A Lightweight Perceptron-Based Intrusion Detection System for Fog Computing", Appl. Sci. 2019, 9, 178; doi:10.3390/app9010178

[13] Aymen Y., Takoua A., Rabah A., "Hierarchical anomaly based intrusion detection and localization in IoT", 2019, 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp 108-113

[14] Veeramreddy J., Koneti M. P., "Anomaly-Based Intrusion Detection System", IntechOpen, 2019.

[15] Sébastien J.J. G., Alvin K.H. L., Craig O. F., et. al., "MAIDENS: MIL-STD-1553 Anomaly-Based Intrusion Detection System Using Time-Based Histogram Comparison", IEEE Transactions on Aerospace and Electronic Systems, 2019

[16] S. Venkatraman & B. Surendiran, "Adaptive hybrid intrusion detection system for crowd sourced multimedia internet of things systems", Multimed Tools Appl, 2019

[17] Luis M.T, Eduardo M., Mikel I. et. al., "An anomaly-based intrusion detection system for IEEE 802.11 networks", 2010 IFIP Wireless Days, 2010

[18] Ayyaz-ul-Haq Q. (B), Hadi L., Jawad A. et. al., "A Heuristic Intrusion Detection System for Internet-of-Things (IoT)", CompCom 2019, AISC 997, 2019, pp. 86–98