A Model to Detect MiTM Attack in IoT Networks:

A Machine Learning Approach

A Thesis Prepared By:

Abel Ashenafi Tadesse

To

The Faculty of Informatics

Of

St. Mary's University

In Partial Fulfillment of the Requirements for the Degree of Master of Science

In

Computer Science

Advisor: Dr. Tibebe Beshah (PHD)

January 15, 2022

ACCEPTANCE

A Model to Detect MiTM Attack in IoT Networks:

A Machine Learning Approach

By:

Abel Ashenafi Tadesse

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

_____

Internal Examiner
Dr. Alembante Mulu

_____

External Examiner
Dr. Sileshi Demesie

_____

Dean, Faculty of Informatics
Dr. Alembante Mulu

February 7, 2022

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Abel Ashenafi Tadesse

_____

Signature

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as advisor.

Dr. Tibebe Beshah (PHD)

_____

Signature

Addis Ababa

Ethiopia

February 7, 2022

Dedication

I would like to dedicate this thesis work to my Father who has taught me everything I know about Computers and Electronics. Not only to who I am now, but to everything I'll ever become Dad, I owe it to you.

"Pops this is to you"

Acknowledgement

First and foremost, I would like to thank almighty God and the Virgin Mary for giving me the strength to finish this intense thesis paper. I would also like to thank my beloved mother Mrs. Tadelech Tilahun for helping me all the way ever since childhood; Mommy, THANK YOU.

Next, I would like to express my sincere and HEART-Felt gratitude for my Advisor Dr. Tibebe Beshah. Really Dr. Tibebe, I was so lucky to have you as my advisor. You have taught, helped, and guided me in doing an M.Sc research. A knowledge I'll always use if I ever pursue another degree of any kind, God willing.

Lastly I would like to thank my friend Anania Mesfin who was there for me when I literally gave up on this study for various reasons. Next to him, I would also like to thank my colleagues Muhamed Hassen and Kirubel Wondimagegnehu for almost being a part of this study.

THANK YOU, ALL OF YOU

Abstract

The Man-in-The-Middle attack is a kind of cyberattack where a perpetrator intercepts an ongoing communication between two parties and use this communications breach to either eavesdrop on the communicated message or alter the message prior to reaching the intended legitimate receiver. In any IoT network, the basic purpose of any smart device in the network is taking part in collecting large amount of data from various sensors located in geographical dispersed locations and relay this information to a Master-Device in the IoT network. Once these collected sensors' data reach the Master device, it relays the sensors' data to a central database or server via gateways wirelessly. IoT devices are usually designed to be deployed in a mass scale and are also designed to operate in remote and hard-to-reach areas. IoT nodes are usually battery powered or scavenge power from their surroundings. Hence, IoT device manufacturers give little emphasis to security. In fact, IoT device manufacturers' main goal is designing nodes that get the job done whilst consuming as little power as possible for as long as possible. Despite their wide spread use and ubiquity, IoT networks are highly vulnerable to cyber-attacks like MitM attacks, and identification of these malicious behaviors is mandatory as tampering IoT data in a malicious manner by adversaries could lead to real-time, real-life catastrophes.

The main objective of this study is building a machine learning model that detects modified sensors' records that originated from IoT networks infected with ARP cache poisoning based on the IoT network's data patterns. Therefore, to build the model, both Normal and Attack data needed to be generated from an environment that mimics an IoT Network.

Hence, for this study, an IoT testbed was built using the NodeMCU ESP32 IoT Module which acts as the master device in the IoT network, a DHT22 Temperature & Humidity Sensor, an MQ2 Gas Sensor, a SW-420 Vibration sensor, and a wireless router. An Adversarial system was also built using a DELL® Core-i3 laptop which runs on Kali Linux with a processor speed of 2.1GHZ and a total installed RAM of 4GB.In this Testbed, data captured form the three sensors are Temperature, Humidity, Smoke in Parts-Per-Million and the level of vibration which are transmitted to a cloud named ThingSpeak server via a wireless router.

In the normal phase, sensors' values are extracted by the NodeMCU device and then transmitted to the ThingSpeak cloud. This data is then labeled as 'Normal' data. The attack phase is performed by the adversarial system which intercepts data coming from the NodeMCU device, modifies it and sends these modified Sensors' readings to the ThingSpeak cloud. This data is labeled as 'Attacked' data. Machine learning classifiers such as SVM, Naïve Bayes, Decision Trees, KNN and Adaboost are built to differentiate the sensors' data as 'Normal' or 'Attacked' data using the Weka Explorer software based on the IoT Network's Sensors' records. From the five candidate algorithms, Decision Trees had the highest accuracy of 95.125 %.

Keywords: IoT Networks, IoT Vulnerability, IoT Network Attacks, Address Resolution Protocol Poisoning, Machine Learning

Table of Contents

Chapter One 1
Introduction 1
1.1 The Internet of Things (IoTs) 1
1.2 Machine Learning 2
1.3 Man-in-the-Middle attack 2
1. 4 Statement of the Problem 3
1.5 Objectives of the Study 4
1.5.1 General objective 4
1.5.2 Specific objectives 4
1.6 Scope and limitations of the study 4
1.7 Significance of the Research 5
1.8 Organization of the Research 5
Chapter Two 6
Literature Review 6
2.1 Internet of Things (IoT) 6
2.1.1 Devices and Inner workings of IoT Networks 7
2.2 Machine Learning 10
2.2.1 Supervised Machine Learning 10
2.2.2 Unsupervised machine learning 11
2.2.3 Other types of Machine Learning 11
2.3 Attacks in IoT Networks 17
2.4 Man in the Middle attacks (MitM) 18
2.4.1 Address Resolution Protocol (ARP) 20
2.4.1.1 ARP Poisoning or ARP Spoofing 21
2.5 Related Works 22
2.6 Summary of Related Works: 23
Chapter Three 24
Research Methodology 24
3.1 General Approach 24
3.2 Specific Research Design 24
3.3 Experimental Setup of the IoT Testbed: 24
3.3.1 The IoT Testbed’s Hardware Components: 25

3.3.2 The IoT Test Bed's Software Components 26
3.4 Dataset Generation 28
3.4.1 Normal IoT Dataset Generation 28
3.4.1.1 Hardware connection between the NodeMCU and the Sensors 29
3.4.2 Attacked IoT Dataset Generation 31
3.5 Data Analysis Methods 39
3.6 Validity and Reliability 40
3.6.1 Research Validity 40
3.6.2 Research Reliability 40
3.7 Communication 40
3.8 Dataset Used in this Study 41
3.9 Data Preprocessing and SMOTEing 42
3.10 Detail about the Experimentation 43
3.11 Preprocessing the data: 44
3.12 Cross-Validation of the Dataset 47
3.13 Experiments: 47
3.13.1 Experiment using the SVM Algorithm 47
3.13.2 Experiment Using the Naïve Bayes Algorithm 48
3.13.3 Experiment using the Decision-Trees Algorithm 49
3.13.4 Experiment using the KNN Algorithm 50
3.13.5 Experiment using the Adaboost Algorithm 51
3.14 Comparison of Experiments 51
Chapter Four 52
Demonstration and Evaluation 52
4.1 Evaluation of Experiments 52
4.2 Demonstration 53
Chapter Five 54
Conclusion and Recommendation 54
5.1 Conclusion 54
5.2 Recommendation 55
References: 56
Appendix: 59

List of Figures:

Figure 1: Simple Microcontroller based embedded systems application 9
Figure 2: Example of an IoT System [8] 10
Figure 3: Linear regression between Weight and Height [13] 12
Figure 4: Graph of a Logistic Regression [13] 13
Figure 5: Decision trees [14] 14
Figure 6: SVM graph between a person's hair length and height [13] 14
Figure 7: Schematic representation of Ada Boost [15] 16
Figure 8: Example of MiTM attack using ARP Poisoning [19] 19
Figure 9: ARP Poisoning or ARP Spoofing [22] 21
Figure 10: Experimental setup of the IoT Test Bed 27
Figure 11**:** The IoT Testbed of This Study 29
Figure 12: Sensors' data in ThingSpeak Server 30
Figure 13: Attacked data generation using the Adversarial system [31] 31
Figure 14: ARP poisoning using Ettercap [31] 32
Figure 15: Sensors' data interception and modification using BurpSuite [31] 33
Figure 16: IP and MAC addresses of the default router and the Adversarial System 34
Figure 17: Host List in the IoT testbed 34
Figure 18: Adding Hosts to Target1 and Target2 35
Figure 19: ARP Poisoning Victims 35
Figure 20: Poisoned ARP cache 36
Figure 21: Sniffed Sensors' data 36
Figure 22: Setting Proxy Port and Destination Port for BurpSuite 37
Figure 23: Setting up the BurpSuite Proxy listener Address 37
Figure 24: Writing Kali Commands to Forward HTTP requests to BurpSuite 37
Figure 25: Forwarding BuprSuite Modified Sensors' readings to ThingSpeak Server 38
Figure 26: Altered and Modified Sensors' Data on ThingSpeak Server 38
Figure 27: Performance metrics for classifiers [31] 39
Figure 28: Sample DataSet 41
Figure 29: Labeled Dataset 42
Figure 30: Original Dataset Used in this Study with Attributes 43
Figure 31: Tuning Parameters to double size of Dataset 44
Figure 32: Normal and Attacked Sensors' records after being balanced 45
Figure 33: Dataset after being doubled and Balanced 46
Figure 34: Experiment using the SVM Algorithm 47
Figure 35: Experiment Using the Naïve Bayes Algorithm 48
Figure 36: Experiment using Decision-Trees 49
Figure 37: Experiment Using the KNN Algorithm 50
Figure 38: Experiment using the Adaboost Algorithm 51
Figure 39: Predicted Outcome using the Decision Tree's Model 53

List of Abbreviations:

ARP: Address Resolution Protocol

DK: Design Knowledge

DNS: Domain Name System

DDoS: Distributed Denial of Service

DoS: Denial of Service

DSR: Design Science Research

EEPROM: Electrically Erasable Programmable Read Only Memory

Hz: Hertz

IDE: Integrated Development Environment

IDS: Intrusion Detection System

IoT: Internet of Things

IP: Internet Protocol

LED: Light Emitting Diode

MAC: Media Access Control

MiTM: Man in the Middle Attack

ML: Machine Learning

MCU: Microcontroller

PDA: Personal Digital Assistants

PPM: Parts Per Million

RAM: Random Access Memory

ROM: Read Only Memory

SMOTE: Synthetic Minority Oversampling Technique

TCP: Transmission Control Protocol

UID: Unique Identifier

Chapter One

Introduction

1.1 The Internet of Things (IoTs)

The Internet of Things (IoT) is a network of smart devices that is composed of sensors, actuators, wireless gateways/ routers, and a programmable central microcontroller or other kind of processor which has the capability of being connected to the router / gateway wirelessly.

Embedded systems are a subset of IoTs in which a group of sensors and actuators are connected to a central electrically programmable and erasable microcontroller or other kind of processor which lies at the heart of the IoT network that is responsible for the communication, computational processing and analytical reasoning tasks needed in the IoT network.

Sensors in the IoT network are responsible for gathering changes in the physical environment like temperature, humidity, angular orientation of objects, pressure etc. and send these physical environmental changes to the microcontroller or processor in a way that is understandable by the processor in either a digital or analog format. Once the processor receives data from the sensors, it then performs necessary computations as per the received data and decodes the sensors' readings to control actuators embedded to it. These actuators might be as trivial as small LEDs (Light Emitting Diodes) or as complex as memory devices which usually communicate to the processor using embedded systems communication protocols or the actuators even might be other nodes in the IoT network. Both sensors and actuators are connected to the peripherals of the microcontroller or processing device.

In the case of IoTs, the microcontroller or processor found in the IoT network is capable of connecting to routers or other nodes in the network wirelessly besides performing computational processing and analytical reasoning. Processors used in both embedded systems applications and IoT networks are usually battery powered and the microcontrollers that make up the heart of the network are resource constrained in terms of Power, RAM, and ROM.

Nevertheless, ever since their advent, IoTs became ubiquitous as engineers and tech-enthusiasts found ways to deploy them in various aspects of our lives. IoTs are found everywhere starting from small wearable devices like PDAs (personal digital assistants), Smart homes, to remote medical monitoring devices. Recently, IoTs are even being deployed in power grids and other mission critical infrastructures.

1.2 Machine Learning

Machine learning is a branch of Artificial Intelligence and computer science that focuses on analyzing data and recognizing patterns in the data using statistics, mathematical computations, and computer algorithms to mimic the way we humans learn and gradually improve accuracy [1].

Machine learning equips computers to learn and improve automatically from experience without being explicitly programmed. In machine learning, the process of learning is entirely dependent on the amount and clarity of data presented to the algorithm. The learning phase begins with observations on the data, performing analysis on the data, discovering patterns in the data, and finally make predictions based on the knowledge obtained from the training dataset and examples provided to it. Intelligence requires knowledge, hence, it is generally endorsed to provide computers with as much knowledge as possible to make them intelligent.

Supervised machine learning is a subset of machine learning where an algorithm teaches a model from a dataset labeled as a training dataset and makes predictions based on that model. Hence, generally speaking, the presence of large amount of training dataset and clarity of the dataset is imperative for building a model that makes accurate predictions.

1.3 Man-in-the-Middle attack

Man-in-the-Middle attack (MitM) is a type of cyber-attack in which an adversarial system or attacker secretly intercepts or infiltrates the communication between two parties and either secretly eavesdrop on the communicated data or alter the data and send this forged data to the receiving party disguised as the legitimate sender.

MitM can be further classified as passive MitM attack and active MitM attacks. In the case of passive MitM attack, the adversary secretly intercepts the communication and simply eavesdrop on the communication without altering the data. However, in the case of active MitM attacks, the perpetrator initially intercepts the communication, alters the data and send the modified data to the receiving party as if the message was untampered with and came from the legitimate sender.

In a Man-in-The-Middle attack, traffic between two devices is passed through a rouge malicious device that is controlled by the attacker. Thus, the attacker can get access to the original traffic and is able to read the communicated data if the communication is unencrypted, he may even inject malware to the traffic [5].

1. 4 Statement of the Problem

According to [2], it is estimated that there will be 30.9 billion active IoT devices connected to the internet in real-time by the year 2025. However, little to no work is being done on addressing the security vulnerabilities found in IoT networks.

The majority of microcontrollers or processors that make up the heart of IoT networks are generally battery powered and are specifically designed to do a single or few repetitive tasks. Hence, most of these processors inherently are resources constrained (i.e. Power, RAM & ROM Constrained). As a result, in most IoT application designs, a lot of emphasis is given to minimizing UP-time and prolonging battery life.

For this reason, IoT chip manufacturers give more emphasis on producing smaller and faster chips that consume as low current as possible and prolonging battery life. However, the authors in [3], have stated that there are various attacks that target IoT devices and IoT networks, such as jamming, spoofing attacks, and exploiting vulnerabilities to gain unauthorized access to the network.

One attack that targets IoT networks is the MitM attack and mitigation strategies proposed for IT networks like encryption to combat MitM attacks in IT networks is generally not suitable for IoT processors since these devices are resource constrained and the fact that these processors usually run on batteries; Hence, even if encryption adds a layer of security in IoT networks, encryption also negatively affects battery life and incur additional overheads on the Microcontrollers of the IoT network. Therefore, simply implementing traditional mitigation strategies proposed for IT networks to combat MitM attacks wouldn't work flawlessly for IoT MCUs because of their resource limitations and few Machine Learning models are developed to identify attacks based on the IoT Network's data Patterns or the network's sensors' records [4].

Although the authors in [31] prepared an IoT testbed to generate MitM attacks on an IoT testbed and used this data to train different supervised ML algorithms in detecting the attacks based on the IoT network's data patterns, the authors were limited to comparing the machine learning algorithms and figuring out which ML algorithm is best in detecting the attacked network's sensors' records. Furthermore, the authors in [31] used only a single sensor and used data comprised of just 480 records to train, test, and compare their ML models.

To train machine learning models with a better training dataset and acquire highly accurate predictions, it is imperative to use a large training dataset with high quality in terms of clarity. As a general rule, the more the training data, the better the ML model will be. However, due to the unavailability of IoT sensors' training dataset regarding MitM attack on IoT networks, part of this study focuses on creating the training dataset itself.

Hence, to address the aforementioned problem, the following research questions are formulated:

- What is the best machine learning algorithm in detecting MitM attacks in IoT environments based on the IoT Network's data patterns?

- How to build a ML model that detects MitM attacks in IoT environments based on the IoT Network's data patterns?

1.5 Objectives of the Study

1.5.1 General objective

The General Objective of this study is to design a Machine Learning Model that detects MitM attacks in IoT networks based on the IoT network's data patterns.

1.5.2 Specific objectives

In order to achieve the general objective, the following steps will be taken:

The specific objectives of this study are:

- Conducting literature review related to this study

- Determining potential supervised machine learning algorithms to be used in this study

- Building an IoT testbed for generating 'Normal' and 'Attacked' Datasets

- Building the machine learning models

- Evaluating the models and making predictions using the best ML model

1.6 Scope and limitations of the study

The scope of this study is to identify the best supervised ML algorithm in detecting sensors' records coming from MiTM infected IoT networks using the training dataset generated by the IoT testbed. Once the best ML algorithm is identified, the model makes predictions on a set of sensors' records and attempt to predict whether a given IoT network's sensors' record is either Normal or Attacked.

The limitation of the study is the fact that, the IoT testbed used in this study is comprised of a single IoT module, 3 sensors, and 4 sensor attributes. Hence, limiting the complexity of data. The other limitation of this study is that, the ML models developed in this study could only be

applicable in detecting infected IoT network's sensors' records, only for sensors' records that have originated from the same environment the training data was collected from.

1.7 Significance of the Research

Having a ML Model that detects MitM attacks in IoT Networks without compromising the Power, RAM, and ROM constraints inherent to IoT's Microcontrollers and using just the IoT Network's data patterns will not only enhance IoT security but also validates the source of the communicated information which is dreadfully needed in IoT environments as, nowadays, IoT applications are ubiquitous and are found in every aspect of our lives; some are even being deployed in mission-critical governmental infrastructures.

Besides the aforementioned points, this research will also provide the following merits for further studies and other scholars interested in the area:

- Creating an IoT testbed using an ESP32 NodeMCU IoT module, a DHT22 temperature & humidity sensor, an MQ2 Gas Sensor, and a SW-420 vibration Sensor.

- This study also provides both 'Normal' and 'Attacked' datasets for scholars interested in doing research in this area.

1.8 Organization of the Research

This study is organized in five chapters. The first chapter is an introductory part, while the second chapter of this study assesses studies related to this one, the third chapter of this study discusses about the methodology, design and experimental setup used in this research, the fourth chapter addresses evaluations and demonstration on the ML models, and the last chapter, chapter five is about conclusion and recommendations for future work.

Chapter Two

Literature Review

2.1 Internet of Things (IoT)

The Internet of Things or IoT for short is a group of wirelessly connected electronic devices used for communicating with one another for the purpose of sharing, relaying, and analyzing sensors' or actuators' digital data to either log the sensors' and actuators' parameters to a central database or activate actuators that might be embedded to IoT networks based on environmental changes obtained from sensors embedded to the IoT networks.

Embedding sensors and actuators to intelligent Microcontrollers and other programmable processors dates a few decades back. However, apart from trivial endeavors like connecting vending machines to the internet, the progress was fairly slow as back then, the production cost of microcontrollers was extremely high, they were big in size and were only capable of executing a few instructions per second which all contributed to a sluggish progress [6]. Despite the slow progress of connecting electronic devices to the internet, the term "Internet of Things" or "IoT" until 1999, was not known, until it was finally coined by an English tech pioneer by the name Kevin Ashton [6]. Even if Kevin Ashton came up with the vision of connecting sensors, actuators, and processors to the internet and coining the term "IoT", the technology back then simply didn't allow his vision to become a reality for another decade [6].

The Internet of Things or IoTs is a description for the networks and interconnections amongst physical things which are technologies that have embedded sensors, actuators, and network devices that are used for establishing connections with other network devices and exchanging data amongst one another over the internet [7]. These devices or objects, or "things" range from minute household objects to extremely sophisticated and intricate mission-critical industrial devices [7].

According to [8], IoTs or the Internet of Things are a group of interrelated and networked devices or 'things' which might be animals, objects, machines or even people where any 'thing' or device in the network has its own unique identifier or id termed as UID or unique identifier which is an identifier used to traverse data on the network autonomously without the need for direct human to computer or human to human or interactions.

According to [9] and [10], IoTs or the Internet of things are a collection of smart and intelligent electronic devices with ON / OFF switches that use the internet and the internet protocol address or IP addresses to transfer and analyze digital data which usually regards to environmental parameters of the surrounding they're immersed in, in the absence of direct human intervention over the internet using the wireless communication medium.

2.1.1 Devices and Inner workings of IoT Networks

IoT networks are made up of vast number of objects or things of various types, shapes, and sizes ranging from small household intelligent-light-bulbs that turn themselves "ON" after recognizing the presence of humans or object in a given environment and turn themselves "OFF" in the absence of humans or objects autonomously, to smart microwaves, which automatically begin cooking food after calculating the time it takes for the owner to get back home using his / her current location obtained from GPS modules installed in the owner's smart car once the owner's car gives signals to the microware he / she is returning home from a long day of work. IoT networks could also be found in extremely complicated and intricate self-driving cars, which makes use of vast amount of sensors' data gathered and analyzed using their complex and intricate network of sensors and microcontrollers embedded on the car that detects objects and obstacles in their path and instructing the car to take necessary actions after synthesizing sensors' readings mounted on the autonomous car. The smart car could also make use of sensors' data obtained from sensors installed on the routes the car is travelling on.

To their very core, any IoT system is comprised of at least 4 set of components;

i. Sensors
ii. Microcontrollers or Processors
iii. Hubs or Gateways
iv. Actuators

Sensors: Sensors in the IoT network are used to sense environmental variables like temperature, humidity, and atmospheric pressure of the surrounding they're immersed in. These sensors then convert these physical environmental variables to either an Analog format in the form of a varying voltage or current or to a digital format in the form of 1s and 0s and relay this data or information to a Microcontroller or processor found in the IoT network.

Microcontrollers or Processors: The Processors or Microcontrollers are the most vital components that make up the hearts and brains of any IoT network or any embedded systems application as these are the devices responsible for the analytical reasoning, computational processing, and the communication tasks needed in the IoT network.

A Microcontroller in its simplest definition is a single-chip computer which has to the minimum; Input and Output ports or IO ports, RAM (Random Access Memory), ROM (Read Only Memory), and CPU (Central Processing Unit) all itched into a single silicon chip during manufacturing.

The word microcontroller which is also called embedded controller is basically derived from two words; Micro and Controller. The term micro is used to depict the size of the controller and the word controller is used to depict that the device is used to control things or objects embedded to its peripherals [11].

Unlike a Microcontroller, A Microprocessor is basically a CPU (Central Processing Unit) that's sole purpose is performing millions or even billions of computations on data fed to it in a very minute amount of time normally in micro-seconds and even nano-seconds recently. In other words, a Microprocessors depend on external devices like Analog to digital converters, digital to analog converters, clock circuits, RAM, ROM, and IO ports to be used in embedded systems or IoT applications.

The execution of tasks by a microcontroller solely depend upon the set of instructions or user-defined programs which reside in the microcontroller's EEPROM or flash memory. To accomplish this, the microcontroller fetches each and every one of the instructions stored in its EEPROM or Flash memory and loads each instruction to its RAM one by one, decodes these instructions and finally execute the required operations as per the instructions loaded on it.

Generally, Microcontrollers are designed to do a single or very few tasks repeatedly in a continuous and never ending cycle unless power is interrupted. However, microprocessors are generally designed to be used in a variety of applications like those found in personal computers.

The other factor to consider is that since the general production and shelve cost of microcontrollers are very cheap as compared to general purpose microprocessors, it is basically unfeasible to deploy microprocessors in IoT networks as IoT networks are made up of thousands or more nodes.

Traditionally microcontrollers used to be programmed using the assembly language commands that were basically unique to a specific device and each assembly language differed from device to device. Although assembly language is fast, it comes with several drawbacks. The Assembly language is made up of various microcontroller manufacturer specific mnemonics which not only forces a programmer to learn a new set of assembly language instructions for every device he / she attempts to program but also makes maintaining, debugging and learning programming microcontrollers using assembly language much more difficult than need be [11].

Recently, microcontrollers could be programmed with high-level languages such as C, C++, Java, and recently even using Python. The fact that high-level languages are much easy to learn and are easy to debug as compared to assembly language has made the development of large and complex firmware (Microcontroller's Program) relatively easy.

Basically, a microcontroller executes a user-defined program stored in its program memory coupled with data received from external sensors (inputs) that are connected to the input ports of the microcontroller and uses this data to control actuator devices (outputs) that are connected to the output ports of the microcontroller.

To clarify and illustrate this concept, Figure 1 shows a microcontroller application which is used to automatically control the temperature of a given room.

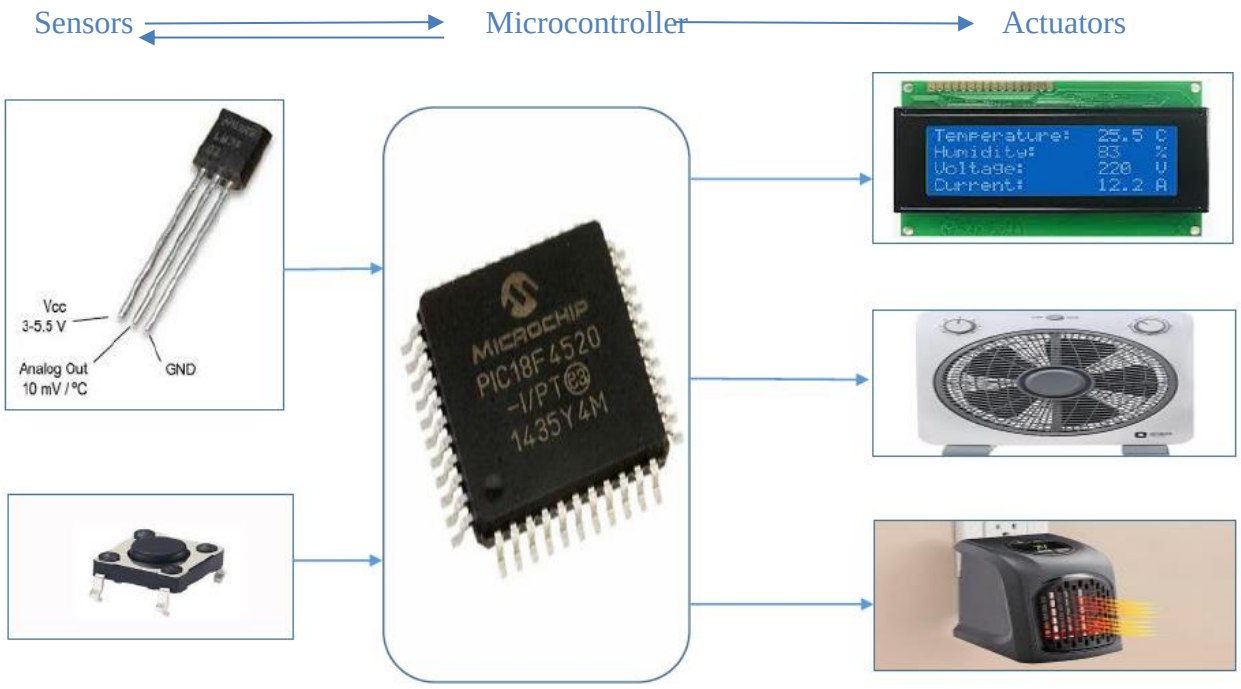


Figure 1: Simple Microcontroller based embedded systems application

Hubs or Gateways: IoT gateways or IoT hubs in the IoT network are used to relay information back and forth between IoT nodes and central servers that are physically apart from one another. These wireless communication devices are usually used to send sensed environmental data obtained using the IoT network's sensors' to the cloud, a central database, or even to other nodes found in the IoT network. IoT hubs and gateways could also be used to relay execution commands coming from a central server to actuate or activate actuators connected to the IoT nodes.

Actuators: Actuators are nothing but devices in the IoT network that actuate or get activated when certain conditions are met. An actuator in the IoT network could be as small and trivial as a small Light emitting diode (LED) or complicated as memory chips or other intricate embedded systems devices. However actuators in an IoT network, unlike traditional actuators found in embedded systems could also be remote databases located elsewhere in the IoT network that are accessed via the hubs or gateways wirelessly.

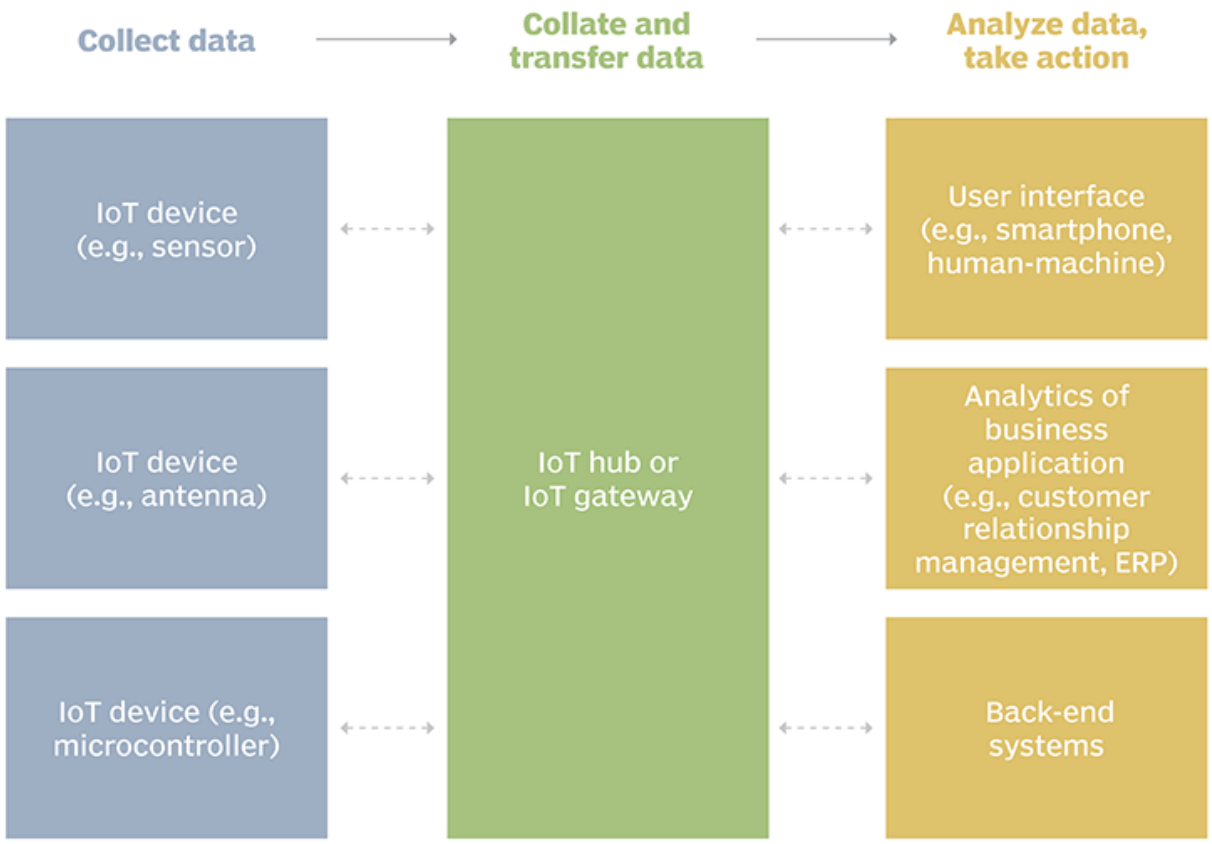


Figure 2: Example of an IoT System [8]

Figure 2 illustrates a typical IoT environment. However, microcontrollers used in IoT environments are usually capable of connecting to the internet or to a central server on their own without the need for additional communication devices. This is because during manufacturing, WiFi (Wireless Fidelity) capability is embedded on the Microcontroller's silicon chip along with other functionalities.

2.2 Machine Learning

Machine learning is a branch of computer science and statistics that studies about the science behind; how machines could be used to make better predictions and decisions based on a training data that is provided to it autonomously without being explicitly programmed. To the very basics, machine learning algorithms could be subdivided in to two categories namely; supervised algorithms and unsupervised algorithms.

2.2.1 Supervised Machine Learning

In supervised machine learning, a learning algorithm is given a labeled training dataset with special features or attributes (typically € R^n×d) and training labels (regularly € 0, ln×k within the case of multi course classification or € R^n in the case of regression) and inquired to learn a model

or function that will outline new items form the same feature space to a prediction within the label space. In the above explanation the letter n is used to denote amount of training examples and the letter d is used to denote the amount of training features used in training the supervised machine learning algorithm [12].

Likewise, k *is* used in the case for multi-class or multi-label problems to represent the number or distinct label values [12]. A learning algorithm will generally identify the model parameters that minimize some loss function between the training examples and training labels [12]. Common types of supervised learning tasks include classification tasks such as text categorization, speech detection, defect reorganization, image recognition, and regression that attempts to learn the mapping between the input data and real-valued number [12].

In the case of supervised machine learning algorithms, the final goal of the algorithm is predicting the outcome of a given set of data based on what it have learnt using a labeled training dataset. Supervised machine learning algorithms predict outcome values using a given set of predictors and finally come up with a supervised machine learning model that accurately predicts the outcome of a dependent variable using the variable's independent variables.

2.2.2 Unsupervised machine learning

Contrary to supervised machine learning techniques, unsupervised machine learning algorithms are NOT trained with examples and their correct answers; rather their purpose is to find patterns within a set of data where no one piece of the data is the answer [16]. In the case of unsupervised machine learning, the whole purpose of the algorithm is articulated towards identifying hidden patterns in the dataset [16]. Hence, unsupervised machine learning algorithms are usually used for grouping and clustering purposes [16].

2.2.3 Other types of Machine Learning

In addition to the aforementioned, supervised and unsupervised machine learning algorithms, other types of machine learning algorithms also exist. A semi-supervised machine learning algorithms work on data that only partially labeled and is often used in the important task of knowledge base construction [17]. A Reinforcement learning type of machine learning algorithm typically comes in handy in the study of robotics and addresses the training of models using trainings and punishments [17].

Some of the most commonly used supervised machine learning algorithms are; linear regression, logistic Regression, decision Trees, Support Vector Machine (SVM), Naïve Bayes, KNN, and AdBoost amongst which the last five algorithms are used in this thesis work.

Linear Regression: Linear regression is a most commonly used type of supervised machine learning algorithm where the linear regression algorithm's model is trained and designed to make predictions on a given dataset and show the link between variables [13]. In linear regression, the link or existing association between the labeled dataset's dependent and independent variables is

depicted by drawing a straight line. The regression line could be represented by a line of a linear equation such as: Z = D * M + E [13].

In the above linear equation:

Z: is used to represent the dependent Variable in the dataset

D: is used to represent the slope of the regression line

M: is used to represent the independent variable in the dataset

And E: is used to represent the Intercept

The coefficients D and E are stemmed based on attenuating the summation of the squared difference between the regression line and the data points in the graph [13]. Figure 3 below illustrates a linear regression with the equation Z = 0.2811 M + 13 between the body and height of a person [13]. So using this equation, we can find the weight of a person given his or her height.

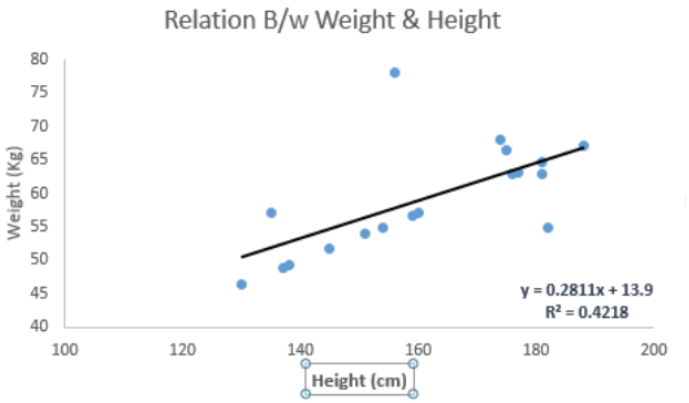


Figure 3: Linear regression between Weight and Height [13]

Linear regression can subdivided to two types of regressions namely; Multiple Linear regression and simple linear regression [13]. The basic difference between simple linear regression and multiple linear regression is the fact that in the case of simple linear regression the number of independent variable is a single variable, however, in the case of multiple linear regression, there exists two or more independent variables [13].

Logistic Regression: Logistic regression like linear regression is a classification algorithm not a regression algorithm as the name implies. Logistic regression is used to estimate discrete values (Binary Values like 0 or 1, Yes or NO, TRUE or False) based on a given set of independent variable(s) [13]. Unlike linear regression, logistic regression is a type of supervised machine learning algorithm that predicts the occurrence probability of an event using logit regression or a

logit function [13]. Since logistic regression predicts probability, its output values lie between 0 and 1.

The odds of probability is as follows:

Odds = P / (1 - P)

Ln (Odds) = ln (P / (1 - P))

Logit (P) = ln (P / (1 - P))

In the above equation:

P: Probability of event occurrence

1 – P: Probability of not event occurrence

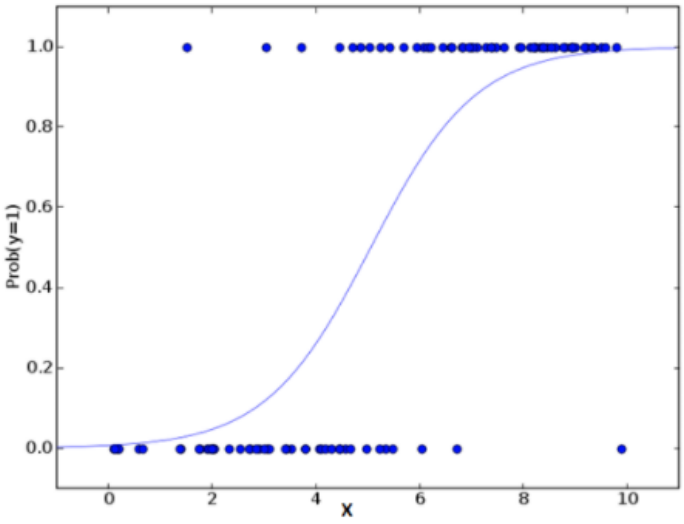


Figure 4: Graph of a Logistic Regression [13]

Decision Trees: When it comes to decision trees, the parameters that are learned re the questions about the data and are basically flowchart like structures that lets classifying input data points or predicting output values based on given inputs [14]. Decision trees are an effective type of classifier machine learning algorithms which are easy to interpret and visualize [14].

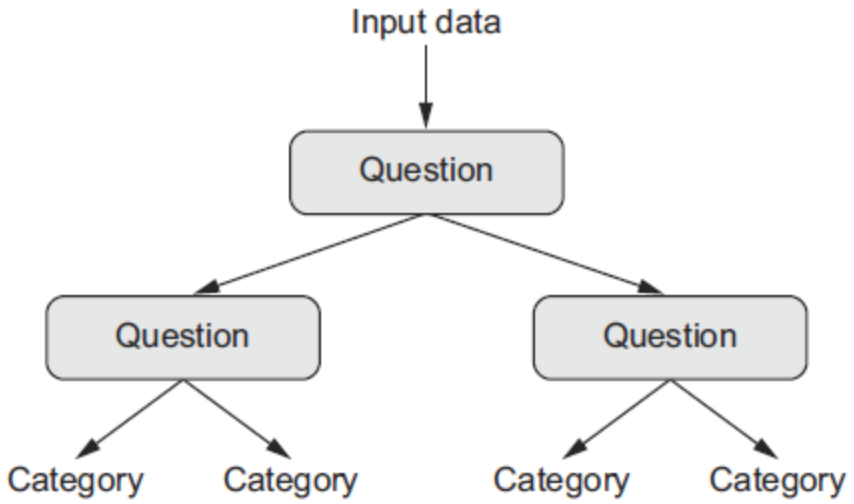


Figure 5: Decision trees [14]

Support Vector Machine (SVM): SVM is a type of supervised machine learning algorithm used for classifying a dataset's records after analyzing the dataset [13]. During the learning phase of SVM algorithm each data point or item is first plotted on a graph where the graph's dimension of space is based on the number of features [13].

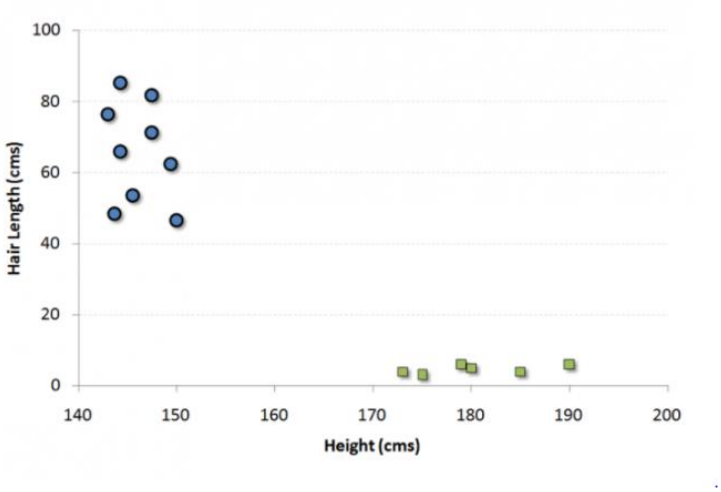
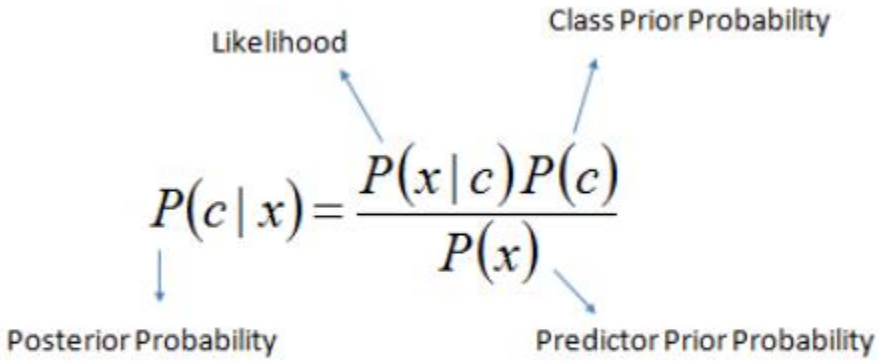


Figure 6: SVM graph between a person's hair length and height [13]

Naïve Bayes: Naïve Bayes like decision tress and SVM is a supervised machine learning algorithm that is used to make predictions on independent variable by utilizing Bayes' theorem and assuming the existence of a particular feature in a class unrelated to the existence of any other feature [13].

The Bayes theorem basically endows a means of calculating the posterior probability i.e. P (c|x) from P(c), P(x), and P(c|x) [13].



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

In the above equation:

- Probability P (c|x) is the posterior probability of (target) given predictor (attribute).
- Probability P (c) is the prior probability of class.
- Probability P (x|c) is the likelihood which is the probability of predictor in a given class.
- Probability P(x) is the prior probability of predictor.

AdaBoost: AdaBoost is short acronym for adaptive boosting. In Ada boost algorithm, a weight is applied to every example in the training data and this weight vector is called D and in the beginning, all these weight are equal [15]. During the training phase of adaboost algorithm, initially a weak classifier is made to train on the data, later the errors made by the weak classifer are calculated and the same weak classifier is then trained for a second time with the same training dataset [15]. However, on the second time of training, weights of the training dataset are adjusted for making the algorithm more accurate and to achieve this, during the second phase of training examples correctly classified initially are weighted less and those that were incorrectly classified are given more weights [15].

For the sake of getting a solo answer from all weak classifiers, adaptive boosting assigns α values to each weak classifier [15].

α values in the case of adaboost are based the error of each weak classifier in the algorithm [15].

The error is represented by the symbol Ɛ which is calculated as:-

$$\frac{\text{The Number of incorrectly classified examples}}{\text{Total Examples}}$$

And the value of weak classifiers represented by the symbol α is calculated as:-

$$(1 / 2) * \ln ((1 - Ɛ) / 2)$$

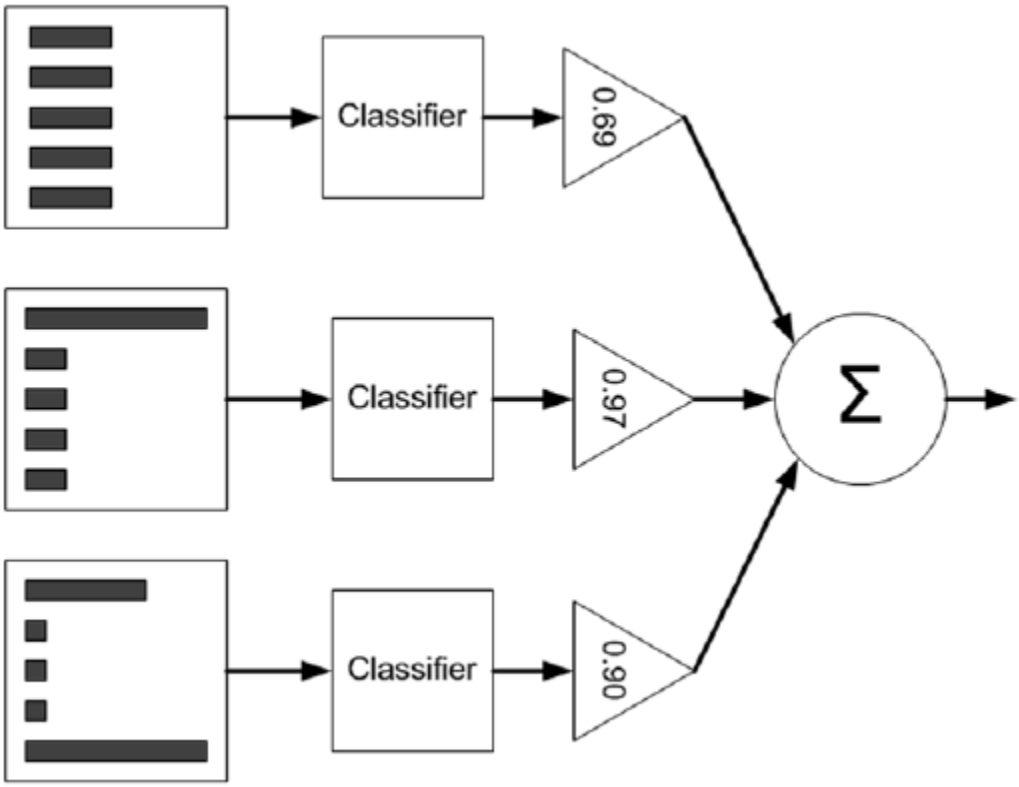


Figure 7: Schematic representation of Ada Boost [15]

KNN: KNN which is an abbreviation for, K Nearest Neighbor, is a type of machine learning algorithm that could be used as either a classifier algorithms or as an algorithm that could be used in solving regression problems [42].

2.3 Attacks in IoT Networks

IoTs are changing the quality of life in every way imaginable at an exponential rate. It is a good thing that these days, we could control the temperature of our home miles away and even turn ON our stoves before we reach home, all made possible because of IoTs. It is a matter of time but inevitably sooner or later, everything connected the internet is prone to cyber-attacks, and IoTs are no exception.

In an IoT network whenever a command or data is communicated between a user and an IoT device, the data or command passes though all four steps of the TCP/IP model, and here the communicated data or command is exposed to a plethora of cyberattacks.

According to [18], IoT Networks are exposed to at least 10 kinds of different cyberattacks:

i. Physical Attacks

Physical Attacks occurs when IoT devices are accessible by anyone who has access to the physical devices themselves. According to [18], most physical attacks infiltrate a given network via an inside perpetrator with a deliberate intent of injecting the malicious codes that usually reside in a flash memory onto the network.

ii. Encryption Attacks

Encryption attacks target IoT communications when the communicated data is usually unencrypted, hence opening the door for an adversary to capture and sniff the communicated message if the adversarial system wishes to do so [18]. In addition to that, once encryption keys are unlocked by an adversary, the adversary can take control of the system by installing his or her own algorithm [18].

iii. DoS (Denial of Service)

A Dos Attack is an attack that occurs when a service becomes unavailable due to various perpetrators sending continuous but futile requests to a server for the sole purpose of exhausting the system and finally forcing it to a halt.

iv. Firmware Hijacking

Firmware is an alias for Microcontroller's software. Firmware Hijacking occurs when an IoT device attempts to perform firmware updates. Another way that firmware hijacking occurs is when perpetrators use the firmware update feature to install malicious code for their own benefits.

v. Botnets

These are types of IoT attacks which occur when an attacker turns a group of networked IoT devices to bots as part of a botnet that is controlled by the attacker [18]. Botnets have the capability to use networked smart devices to transfer private and sensitive data to a location set by the perpetrator.

vi. Man-in-the-Middle Attacks

MiTM attacks occur when the legitimate communication between two legitimate communicating entities is secretly intercepted by an attacker or adversarial system. In a MiTM attack, the adversary tricks both parties that one is communicating with the other. Once the adversary intercepts the communication, he or she can manipulate and alter the communicated data and send it to the receiver as if it came from the legitimate sender.

vii. Ransomware

Ransom is a type of malware that locks down access to sensitive data and information that resides in the victim's system by encrypting them and rendering them futile to the user.

viii. Eavesdropping

In this type of attack, a perpetrator intercepts a weak communication network and continuously listens IN on the communication passively without altering the communicated data in any manner.

ix. Privilege Escalation

In this kind of attack, adversaries look for bugs and vulnerabilities in IoT devices and once these vulnerabilities are discovered, the adversary plants malware on an IoT device and steals confidential data.

x. Brute Force Password Attack

In a brute force password attack, adversaries attempt to gain access to a network by submitting many passwords with the hopes of finding the correct password eventually after many attempts. Sometime adversaries even use automated software that generate random guesses for them. If they succeed in gaining access to the network, they will plant a malware on the device and later use this malware to steal confidential data.

2.4 Man in the Middle attacks (MitM)

A man-in-the-middle attack is a common type of attack where perpetrators immerse themselves in the middle of a communication happening amongst other communicating parties and once the perpetrator is inside the communication channel, the perpetrator can eavesdrop on the

communicated message silently or even alter the message before it reaches the intended legitimate receiving party.

Nevertheless, MiTM attacks existed centuries prior to the appearance of IoTs and even computers in general. To show the very basics of how MiTM attack works, a typical MiTM example is a malicious secretary who opens sealed memoranda messages and either read the messages or even alter the contents of the memoranda messages prior to giving the memoranda to the intended receiver. In the era of modern computers, MiTM attack is executed using different techniques such as ARP cache poisoning, Session Hijacking, DNS cache poisoning etc. However, all of the techniques basically intend to intercept a communication occurring between two parties. Figure 8 below shows how to execute MiTM attack using ARP cache Poisoning.

Figure 8 shows how we can exploit the vulnerabilities of ARP protocol to execute MiTM attacks by poisoning ARP caches. The ARP cache table is designed to allow hosts in a network to map IP addresses other devices on the network to their corresponding MAC addresses. However, according to figure 8, if an attacker wishes to receive data intended to 'Host B', the attacker could forge an ARP reply message indicating that 'Host B's IP address maps to the attacker's MAC address and advertise this packet by sending it to the network's broadcast address. Hence, any other host connected to the switch attempting to send data to Host B, in reality will be sending the data to the attacker rather than the legitimate receiver, Host B which is pictorially illustrated in Figure 8. The attacker could be even more malicious and advertise itself as the Network's default gateway [19]. Hence, any outbound targeted at remote hosts such as HTTP using online commercial websites containing sensitive users' credentials would initially be routed to the attacker first [19]. Using this trick, if the adversary wishes, could now modify the communicated data and forward it to the correct gateway, hence creating the MiTM attack.

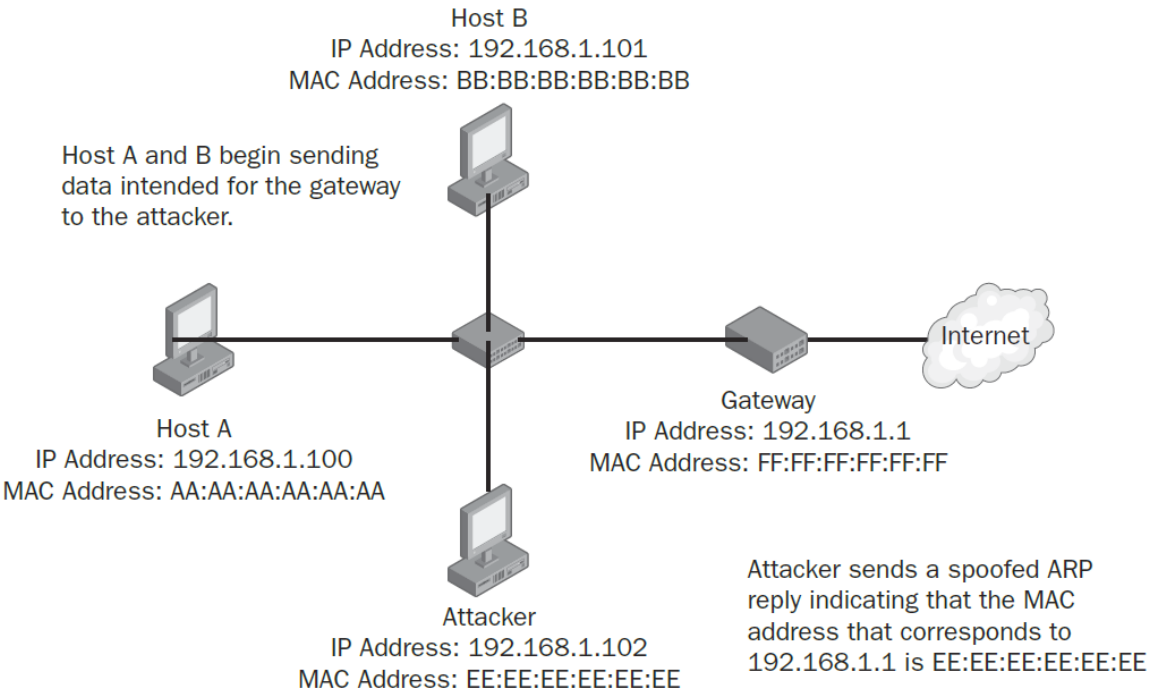


Figure 8: Example of MiTM attack using ARP Poisoning [19]

In a typical MiTM attack, the attacker must possess control of one of the many firewalls, routers, or gateways through which the targeted communication traverses [20].

An adversarial system could execute MitM attacks on an ongoing communication using several ways which might by poisoning the ARP caches or ARP tables of its targets, by spoofing the Domain Name system or by executing SSL Hijacking just to name a few [19].

In past times, MitM attacks basically targeted vulnerable network devices like personal laptops, however, recently due to the mass deployment and even more exploitable vulnerabilities found in IoT devices, a tremendous number of users are on the verge of being exploited [19].

2.4.1 Address Resolution Protocol (ARP)

For the sake of traversing data packets from one end of a network to the other, a great number of internet services rely on the Transmission Control Protocol or TCP which provides virtual circuits for communication by sitting on top of IP [21]. This provision of virtual circuits is achieved by initially tearing apart a communicated data stream to individual IP packets which will later be merged back to a single data stream at their destination [21]. The Transmission Control Protocol also ensures each packet reaches its intended destination using acknowledging signals [21].

Local Area Networks or LANs for short usually communicate with one another using Ethernet and by making use of each devices' unique MAC [Media Access Control] addresses and IP addresses that are either statically or dynamically assigned by the LAN which is all achieved because of ARP as it is the one responsible for keeping records that maps IP addresses with MAC addresses [21].

ARP translates IP addresses to MAC addresses, and vice versa [22]. Most of the time nodes on a network use ARP to connect to the World Wide Web via a gateway or router.

In the ARP protocol, each host in the given network keep a record of IPs corresponding to individual MAC addresses using it ARP table as this information would come in handy when it wishes to communicate with other host in a later time but in the case when a host couldn't find the corresponding MAC address for an IP address, it tries to establish communication with that device by broadcasting ARP request packets to all devices connected to the network [22].

Unfortunately the Address Resolution Protocol was not designed having cyber vulnerability exploitations in mind and due to this weakness, ARP does not verify that ARP replies came from legitimate owners of the IP address it was searching for in the network, not only that, this protocol also allows hosts to accept ARP reply messages even if a host in the network never sent out ARP broadcast messages which widely opens the door for ARP spoofing or ARP poisoning attacks [22].

2.4.1.1 ARP Poisoning or ARP Spoofing

ARP poisoning or ARP spoofing, is a type of MitM attack that allows perpetrators intercept an ongoing communication between two networked devices. The operation of ARP poisoning is listed as follows [22]:

1. The first thing the adversary performs prior to intercepting the communication is gaining access to the network and once the perpetrator is inside the network he / she then determines at least two target devices [22]. Let's assume the two target devices are a workstation computer and a router

2. Now since the attacker has infiltrated the network, he / she needs to send out forged ARP responses to the targeted devices using spoofing tools such as Ettercap [22].

3. The attacker forges fake ARP responses for tricking both workstation and router that the MAC address for both target devices' IP address is the attacker's own MAC address and deceits both devices to connect to the perpetrator itself, rather than to each other [22].

4. Once the workstation and router update their ARP tables with the forged ARP responses, the two devices assume they are communicating with each other but in reality, both devices are unknowingly communicating to the attacker's machine [22].

5. This deceptive act committed by the attacker's system allows the attacker's machine to lay amid all ongoing communications conducted between the workstation and router [22].

The whole aim of ARP poisoning is to allow an adversarial system's machine pose as both sides of an ongoing communication [22]. Figure 9 illustrates how an ARP poisoning or ARP spoofing takes place.

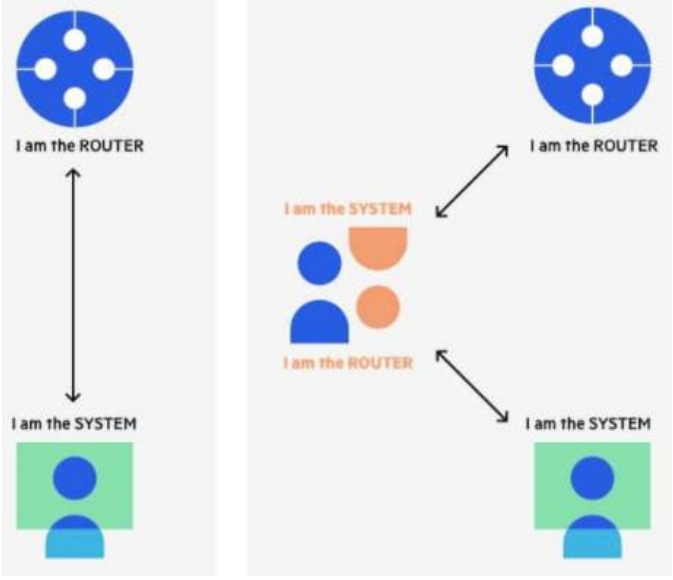


Figure 9: ARP Poisoning or ARP Spoofing [22]

Once the perpetrators succeed in executing an ARP poisoning attack, they can continue routing the communication as-is (they can read sensitive data), perform session hijacking (gaining access to accounts the user is currently logged into), and alter communication (inserting a malicious file or altered data) [22].

2.5 Related Works

As shown by Lawrence [23], IoT networks are highly vulnerable to software layer attacks which usually involve remote network attacks that exploit vulnerable communication protocols, weak cryptography implementations, malware, and social engineering attacks. The study conducted by N. Vlajic et al [24], showed experiments conducted on IoT based webcams regarding the executing DDoS attacks on the webcams and suggested means to prevent the DDoS attacks. In their study, the attacks were categorized based on the layer of origin of attack which were the physical, network and application layers.

The study conducted by Andrea et al [25] explored the security goals required for secure IoT systems, and classified security challenges and issued using new countermeasures for four classes of attacks; Physical, Network, Software, and Encryption attacks. Furthermore, future directions for IoT networks security were discussed in [25].

The study conducted by Liang Xiao et al [26], demonstrated using supervised ML, unsupervised ML, and reinforcement learning techniques to keep users' privacy in IoT networks by addressing attacks such as spoofing attacks, denial of service attacks, frequency jamming attacks and eavesdropping on IoT device communications. The study conducted in [26] focused on keeping the data privacy of users by the use of ML based access control, malware detection, secure offloading, and IoT authentication.

The study conducted by K.V.V.N.L Sai Kiran et al [31], demonstrated a comparison of supervised machine learning models in detecting MiTM attacks in IoT environments based on the IoT network's sensors' readings. Their comparison was based on the ML models' performance in detecting ARP poisoning infected IoT Network's sensors' records. For mimicking an IoT network, the authors in [31] prepared an IoT testbed using the ESP8266 IoT module and a single DHT11 Temperature & humidity sensor. For generating the MitM attack on their IoT test bed, the authors in [31] used a laptop running on Kali Linux.

The study conducted by Mohammad Noor Injadat et al [27], proposed an optimized ML-based framework that combined Bayesian Optimization Gaussian Process (BO-GP) and decision trees classification model to detect botnet attacks on IoT devices. The aim of the study conducted in [27] was emphasized on creating an efficient and dynamic framework for IoT devices to detect botnet attacks.

Using the ARP table tool, the defense tool against ARP attacks, is proposed in the study [28]. The study conducted in [29] described how to conduct MiTM attack in any network using Kali Linux and Ettercap. The study conducted by Chaabouni et al [30] proposed an architecture to

detect Network Intrusion Detection for IoT systems to detect known and unknown IoT attacks without being protocol-dependent and with less resource consumption, thus, with respect to IoT challenges.

Nevertheless, the authors in [31] conducted a study using an IoT network's sensors' records and an infected their IoT network using ARP poisoning and used both dataset as a training dataset to compare different machine learning models to identify the best ML model in detecting MitM attacks in the IoT network. The authors in [31] were limited to comparing different ML models and figuring out the best algorithm in detecting infected Sensors' readings. Furthermore, the other limitation of the study conducted in [31] is, whilst preparing the IoT testbed, the testbed in [31] was comprised of a single IoT module and a single sensor in addition, 480 records were used to train and test the ML models and comparison of the machine learning models was based on the 480 records the authors generated.

According to the literature, this research observed that there exists various research on building Intrusion Detection System (IDS) for IoT networks based on their network patterns. However, as per the researcher's knowledge, not many researches have been conducted in building IDS systems for IoT networks based on the IoT Network's Sensors' records.

2.6 Summary of Related Works:

The authors in [23], [24], and [25] have showed vulnerabilities in IoT networks and also showed how to exploit these vulnerabilities to generate various kinds of attacks on IoT networks. The authors in [26], [30], and [27] have showed how to build Machine learning models to mitigate attacks that attempt to exploit IoT network vulnerabilities based on the IoT networks' network patterns. The authors in [29] have showed how to perform ARP poisoning using the Ettercap tool in Kali Linux and the authors in [28] have proposed ways to defend against attacks that target the Address Resolution Protocol. According to the literature review, this study observed that few machine learning models have been developed to identify MiTM attack in IoT networks based on the IoT networks network patterns and as per the researcher's knowledge, no machine learning model have been developed to detect modified sensors' records originating from MiTM infected IoT networks based on the IoT network's sensors' records.

Hence, this study aims at developing a machine learning model that detects intercepted and modified sensors' data that originated from infected IoT networks based on the network's data patterns or sensors' records. The IoT testbed used in this study is comprised of an ESP32 based NodeMCU IoT module, a DHT22 Temperature & Humidity sensor, an MQ2 Gas sensor, and a SW-420 vibration sensor. In this research, for training and testing the candidate machine learning models, a total of 2,000 records have been used. Of the 2,000 records used in this research, 1,500 records were of type 'Normal' and the remaining 500 were of type 'Attacked' sensors' records.All sensors' records i.e. 'Normal' and 'Attacked' sensors' records are obtained from the IoT testbed developed for this study.

Chapter Three

Research Methodology

3.1 General Approach

The research methodology used in this study is the Design Science Research (DSR) methodology. According to [38] Design Science Research (DSR) is a research paradigm that aims at creating innovative artifacts that will solve human problems and enhancing human knowledge. According to the authors of [39], Design Science Research aims to enhance human knowledge with the advent of innovative artifacts and the creation of design knowledge (DK) via innovative solutions for real-world problems.

3.2 Specific Research Design

This research is conducted using the cyclic process of DSR suggested by Peffers et al [41]. According to [41], DSR process is a combination of five core processes; these five processes are problem identification & Motivation, Defining Objective of Solution, Design & Development, Demonstration & Evaluation, and Communication.

In this study, the DSR methodology is applied to design a model that detects MiTM attacks in IoT networks based on the IoT network's data patterns using machine learning models. To achieve this, an IoT test bed is built that mimics an actual IoT network. Dataset used to train the machine learning models is generated using this IoT testbed. A detailed definition of each process used to generate both type of datasets is briefly described on the next sections.

3.3 Experimental Setup of the IoT Testbed:

IoT testbeds are platforms which are used to mimic an actual IoT network for the purposes of testing vulnerabilities in IoT networks. IoT test beds are also used to propose and test solutions that aim to mitigate IoT network vulnerabilities.

The IoT test bed used in this study is comprised of both hardware and software components. The hardware components used in the IoT test bed used in this study include: the NodeMCU ESP32 IoT module, the DHT22 Temperature & Humidity Sensor, the MQ2 Gas Sensor, the SW-420 Vibration sensor, an ADSL wireless router, and an adversarial system. The software components of the IoT test bed include: Kali Linux, WireShark, Ettercap, BurpSuite, the ThingSpeak cloud, the Arduino IDE, and Weka Explorer.

A detailed description of the IoT Test Bed's hardware and software components is given on the next sections of this study.

3.3.1 The IoT Testbed's Hardware Components:

The NodeMCU ESP32 IoT Module*:* The ESP32 is an IoT module developed and released by Espressif Systems® in September 6, 2016. The ESP32 incorporates WiFi and Bluetooth capabilities and harbors a Dual-Core Tensilica Xtensa® LX6 Microprocessor that uses a clock frequency of 2.4 GHz (Giga Hertz) [32]. The ESP32 is a successor of Espressif Systems' infamous ESP8266 IoT module which unlike the ESP32 only provides WiFi connectivity. The ESP32 operates on 3.3 V DC, has a SRAM of 512 KB, and a flash memory of 16 MB all incorporated on a single chip [32]. Due to its cheap cost and minute size most IoT applications are built using the NodeMCU ESP32 IoT module.

The NodeMCU ESP32 is used in this study to extract data from sensors and send readings of the sensors to the ThingSpeak cloud via a wireless router.

The DHT22 Temperature & Humidity Sensor*:* The DHT22 is a cheap Capacitive-type Humidity & Temperature sensor developed by Aosong Electronics. The DHT22 sensor operates in DC voltage ranges between 3.3 V – 6V. The DHT22 sensor senses relative humidity and has a wide operating temperature range of -40 to 80 degree Celsius. The DHT22 sensor converts the temperature and humidity data to a digital format and sends this data to a master device using its data communication pin.

The DHT22 Temperature & Humidity sensor is used in this study to sense temperature & humidity and send these parameters to the ESP32 master device.

The MQ2 Gas Sensor: The MQ2 Gas sensor is a sensor with high sensitivity and fast response time which is used to sense the concentration of certain gasses in an environment. Gases the sensor is able to sense are propane, methane, LPG, smoke, hydrogen, carbon monoxide and alcohol. The MQ2 operates on voltage ranging from 3.3V - 5V DC.

The MQ2 is used in this study to sense the concentration of gases in the atmosphere and then send this parameter to the ESP32 master device.

The SW-420 Vibration Sensor: The SW-420 vibration sensor is a small vibration sensor released in September 2018 that is used for detecting vibrations in its environment. The SW-420 senses the amount of vibration in its surrounding. The SW-420 sensor can operate anywhere between 3.3 V – 5 V DC. The sensor converts the sensed data to a digital format and sends this data to a master device using its data communication pin.

The SW-420 is used in this study to sense the vibration level and then send this parameter to the ESP32 master device.

The ADSL Wireless Router: The ADSL wireless router is used to connect the NodeMCU device to the ThingSpeak cloud. The ADSL router used in this study is the N 300 ADSL2+4-Port Router [35].

The Adversarial system: The adversarial system used in this study is a DELL® Core-i3 laptop which runs on Kali Linux with a processor speed of 2.1GHZ and a total installed RAM of 4GB.

3.3.2 The IoT Test Bed's Software Components

ThingSpeak*:* ThingSpeak is an online IoT analytics platform that allows IoT developers to store, aggregate, visualize, and analyze live IoT data streams in the ThingSpeak cloud [33].

In this study, both data captured form sensors in the normal phase and attack phase are moved to the ThingSpeak cloud for training the ML models.

Kali Linux: Kali Linux is a Debian Linux distribution open source platform which is maintained and funded by Offensive Security that has built-in tools for network forensics, security auditing, reverse engineering, and penetration testing [34].

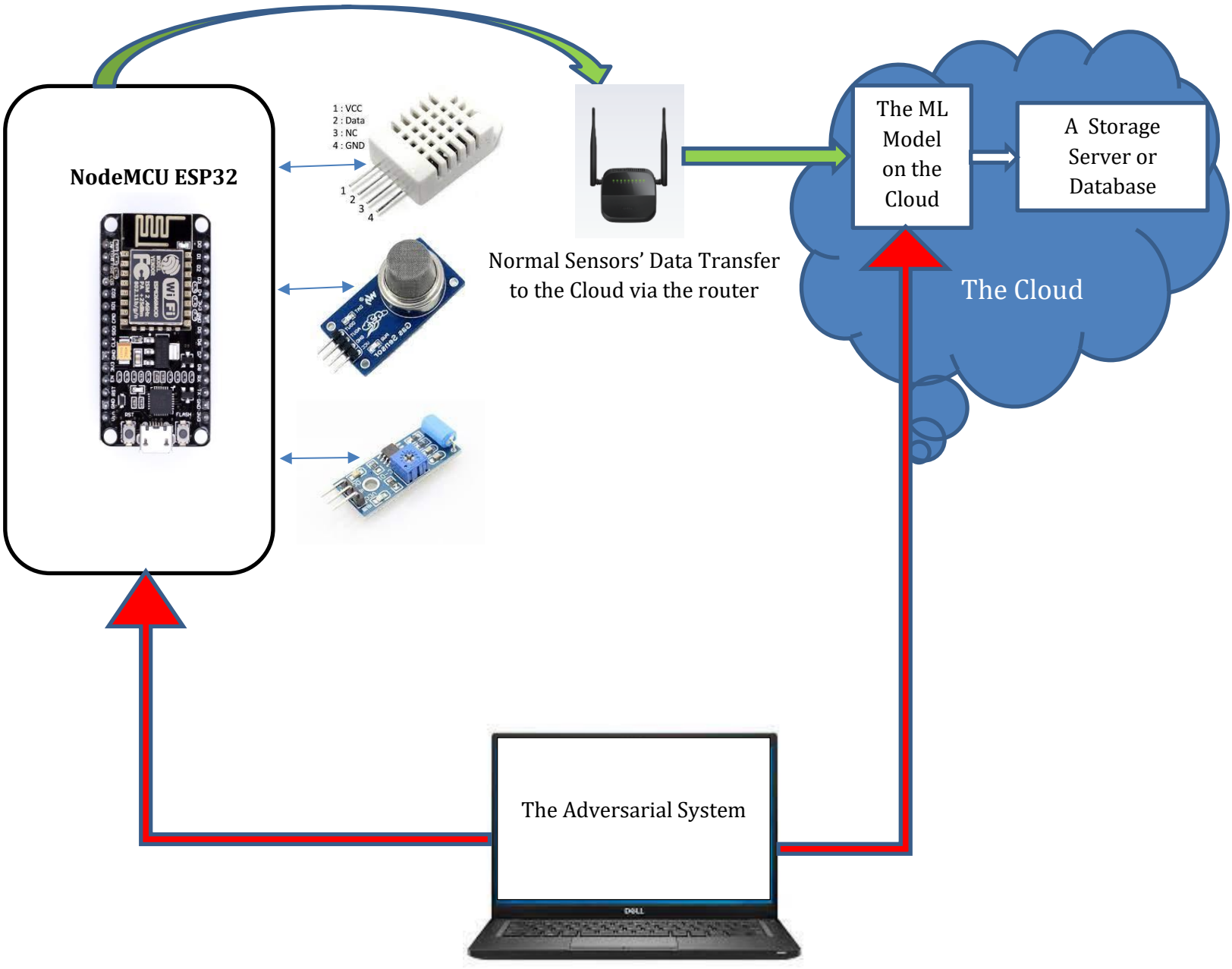
Kali Linux is used in this study to generate attacked sensors' records on the IoT test bed by performing ARP poisoning on the wireless router and the ESP32 NodeMCU IoT module.

WireShark: WireShark is the world's foremost and widely-used network protocol analyzer that lets you see what's happening on your network at a microscopic level and it is the de facto standard across commercial and non-commercial agencies across the world [37].

The Arduino IDE*:* The Arduino IDE is an open-source IDE used for electronic prototyping which makes writing code for Embedded Microcontrollers and IoT modules relatively easy. Active development of the Arduino software is hosted by GitHub [36]. Arduino IDE is used in this study to write program to the NodeMCU ESP IoT module to extract sensor data from the sensors and send these data to the ThingSpeak Cloud.

Weka Explorer: The Weka Knowledge Explorer is an easy-to-use graphical user interface software that harnesses the power of the Weka software. Each of the major Weka packages Filters, Classifiers, Clusterers, Associations, and Attribute Selection is represented in the Explorer along with a Visualization tools which allows datasets and the predictions of Classifiers and Clusterers to be visualized in two dimensions [43]

Figure 10 depicts a graphical representation of the IoT Test Bed used in this study. In Figure 10, the 3 bi-directional blue lines represent 2-way WIRED communication between the ESP32 NodeMCU Master device, the DHT22 Temperature & Humidity sensor, the MQ2 Gas Sensor, and the SW-420 Vibration sensors respectively. The green lines are used to represent a normal or non-attack wireless communication between the NodeMCU and the ADSL router and between the ADSL router and the ThingSpeak cloud. The red lines between the ESP32 NodeMCU and the cloud is used to represent the ARP poisoning, data interception, and modification actions performed by the adversarial system before sending these modified sensors' data to the cloud.



ARP poisoning, Sensors' data interception, and data modification performed by the Adversarial system prior to forwarding these modified Sensors' data to the cloud

Figure 10: Experimental setup of the IoT Test Bed

3.4 Dataset Generation

In order to train the machine learning models and make the models differentiate the IoT network's sensors' records as 'Normal' or 'Attacked', both 'Normal' dataset and 'Attacked' dataset was generated using the IoT testbed depicted in Figure 10 of this study. The detailed processes involved in generating both 'Normal' datasets and 'Attacked' datasets using the IoT testbed is briefly described in sections 3.4.1 and 3.4.2 of this study respectively.

3.4.1 Normal IoT Dataset Generation

To generate the 'Normal' dataset needed to train the ML models used in this study, the first and foremost task required was connecting the three sensors i.e. the DHT22 Temperature & Humidity sensor, the MQ2 gas sensor, and the SW-420 Gas sensor to the NodeMCU master device and write appropriate firmware to instruct the NodeMCU master device to send the sensors' readings to the ThingSpeak cloud.

According to Figure 10, the 3 bi-directional blue colored arrows are used to illustrate the communication between the three sensors and the NodeMCU IoT module which is the master device for all three slave sensor devices.

Once the three sensors collect physical phenomena from the environment they're immersed in, they will forward these environmental changes or Sensors' readings to the ESP32 NodeMCU master device which will perform appropriate raw-data processing on the sensors' records and later forward these sensors' readings to the ThingSpeak Cloud via the ADSL router wirelessly. The Green double-sided arrows of Figure 10, are used to illustrate this step i.e. forwarding Sensors' records from the NodeMCU module to the ThingSpeak Cloud. Hence, generating the 'Normal' dataset used for this study.

3.4.1.1 Hardware connection between the NodeMCU and the Sensors

The NodeMCU IoT module and the three sensors are connected to one another using jumper wires to establish communication with each other.

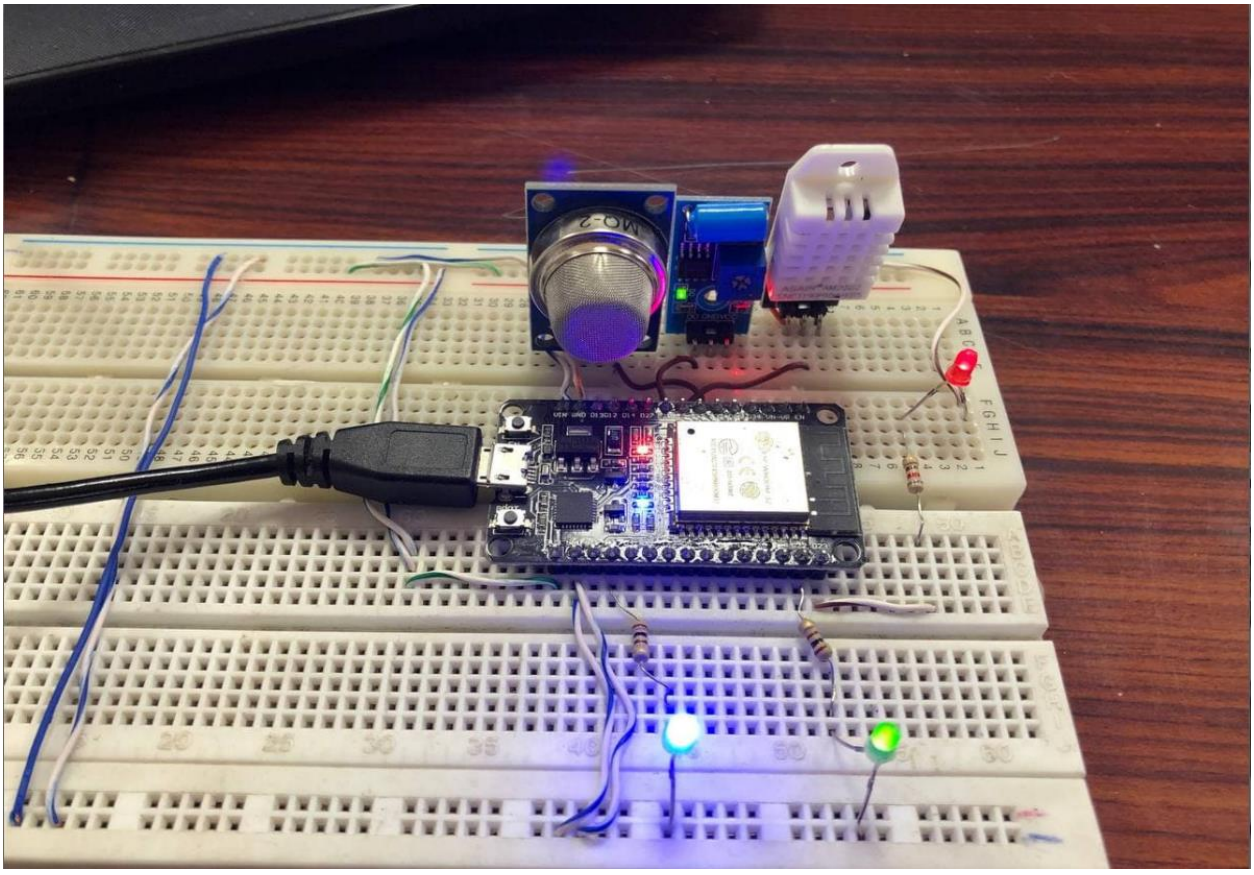


Figure 11**:** The IoT Testbed of This Study

According to Figure 11, Starting from the Left side of the Picture, the first component is the MQ2 Gas sensor, then comes the SW-420 Vibration sensor, and the last component is the DHT22 Temperature & Humidity sensor, and the central component is the ESP32 NodeMCU IoT module. All three sensors get their power from the NodeMCU Master device which draws 5Volts DC from a laptop using a Micro USB-B cable. The NodeMCU IoT module has voltage regulator that converts the 5V DC to 3.3V DC. The MQ2 Gas Sensor has a total of four pins and these are the AO pin for analog output, the DO pin (Digital Output), the GND pin for connecting with a common ground, and the VCC pin for connecting to a 3.3V power source from the breadboard. The AO pin of the MQ2 sensor is connected with the Analog pin 33 of the NodeMCU. The SW-420 Vibration sensor has a total of 3 Pins and these pins are the DO pin which is used to send data to the NodeMCU master device, the GND for connecting with a common ground, and the VCC pin for connecting to a 3.3V power source from the breadboard. The DO pin of the SW-420 vibration sensor is connected to the GPIO pin 26 of the NodeMCU

device. Likewise, the DHT22 has a total of three pins and these pins are the VCC, GND, and the Data out pin. The VCC pin is used to connect the sensor with a 3.3V power source from the bread board, the GND pin is used to connect with the circuit's common ground, and the Data out pin as the name implies is used to send data to the NodeMCU master device. The Data out pin of the DHT22 sensor is connected to the GPIO pin 25 of the NodeMCU master device.

Once all the hardware connections are established, the firmware or the NodeMCU Microcontroller's software is written using the Arduino IDE. Arduino IDE is used in this study to write firmware that instructs the NodeMCU to extract data form the three sensors and send the sensors' data to the ThingSpeak cloud. The version of Arduino IDE used in this study is Version 1.8.16. The following Arduino libraries were used while writing the firmware; "DHT.h", <WiFi.h>, and the "ThingSpeak.h" Arduino libraries.

The temperature and humidity data is extracted from the DHT22 sensor and sent to the ThingSpeak cloud. Likewise, the amount of Smoke in Parts-Per-Million (SPPM) is extracted from the MQ2 sensor and is sent to the cloud. Like other sensed data, the vibration data is extracted from the SW-420 sensor and is sent to the ThingSpeak cloud.

Once data from the three sensor is sent to the ThingSpeak cloud, it is downloaded as a .csv file for training the candidate ML models.

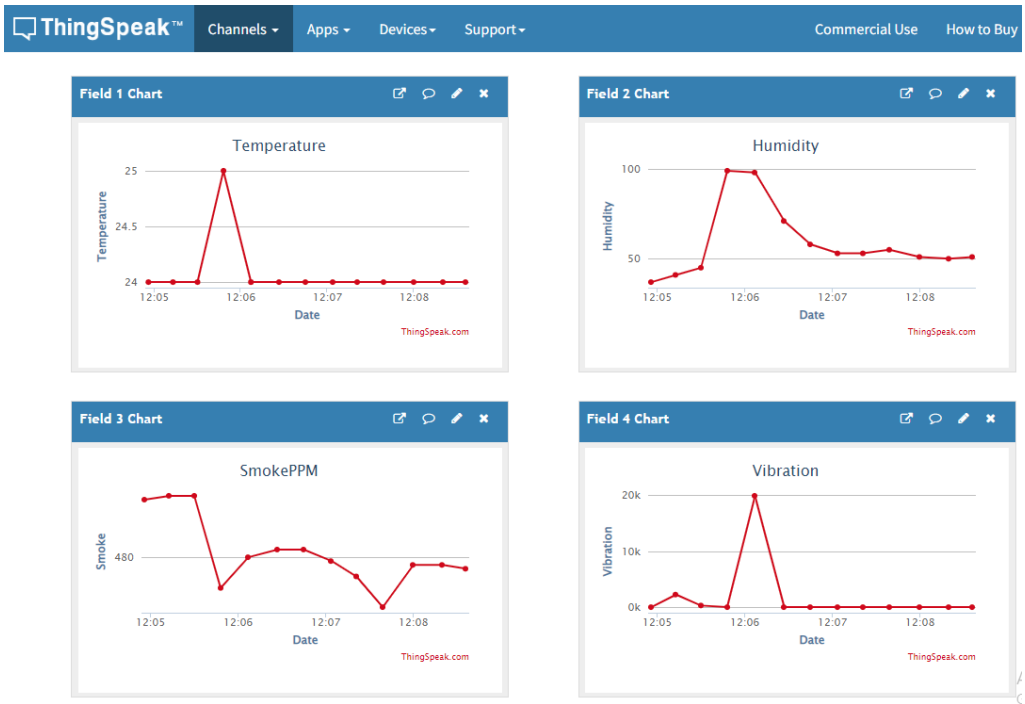


Figure 12: Sensors' data in ThingSpeak Server

3.4.2 Attacked IoT Dataset Generation

To generate 'Attacked' dataset using the IoT testbed, an adversarial system is built in this study to deliberately generate MitM attacks on the IoT test bed using ARP poisoning. To generate these 'Attacked' dataset, primarily the communicated packets were first visualized in WireShark for the purpose of identifying the IP addresses of the NodeMCU and the wireless gateway as well as analyzing the flow of traffic in the IoT network. The MiTM attack is achieved by using WireShark and Kali Linux which has built-in tools to generate MiTM attacks on the IoT test bed. The process of generating 'Attacked' dataset in this study is depicted using the two double faced Red arrows in Figure10.

Figure 13 below shows the entire steps used in this study for generating attacks on the IoT test bed by the adversarial system.

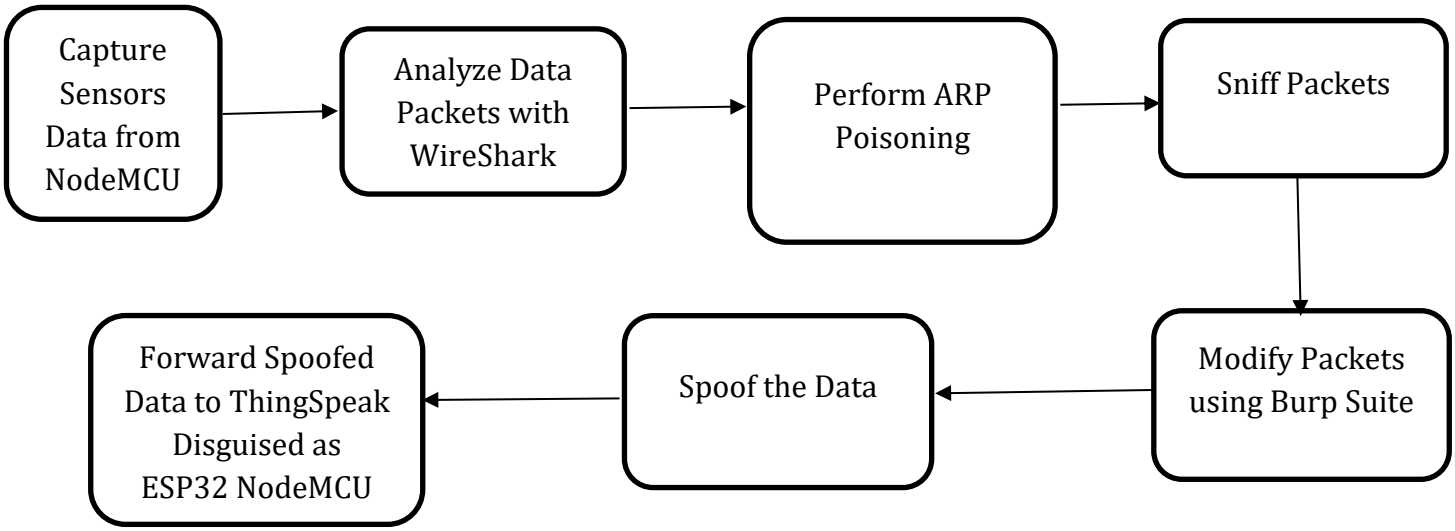


Figure 13: Attacked data generation using the Adversarial system [31]

To perform MitM attacks on the IoT test bed, primarily data packets containing the three sensors' readings is analyzed using WireShark. Once the sensors' data packets are analyzed with WireShark, Ettercap is used to perform ARP poisoning on the IoT testbed. Afterwards, the data packets are sniffed and later modified with Burp Suite. Lastly, the data packets modified by the adversarial system are forwarded to the ThingSpeak server under the disguise of the ESP32 NodeMCU device.

The adversarial tools used to execute MitM attack on the IoT test bed are WireShark, Ettercap, and Burp Suite. These software are built-in tools found in the Debian Distribution of Kali Linux 2021.1.

A detailed description of the tools and associated methods used for conducting the MitM attack is discussed below:

WireShark: WireShark is a network protocol analyzer which is primarily used for analysis of packets in a given network. Analysis can be done in various aspects like time stamps, source and destination IP addresses, ARP request & replies, HTTP communications etc.

In this study, WireShark is used to identify the IP address of all nodes in the IoT test bed and to identify the communication protocol used in transferring data between the nodes.

Ettercap: Ettercap is an open source network security tool found inherently on Kali Linux that is used for executing ARP poisoning in a network. Figure 14 below depicts the overall working principles of Ettercap to generate MitM attacks using ARP poisoning.

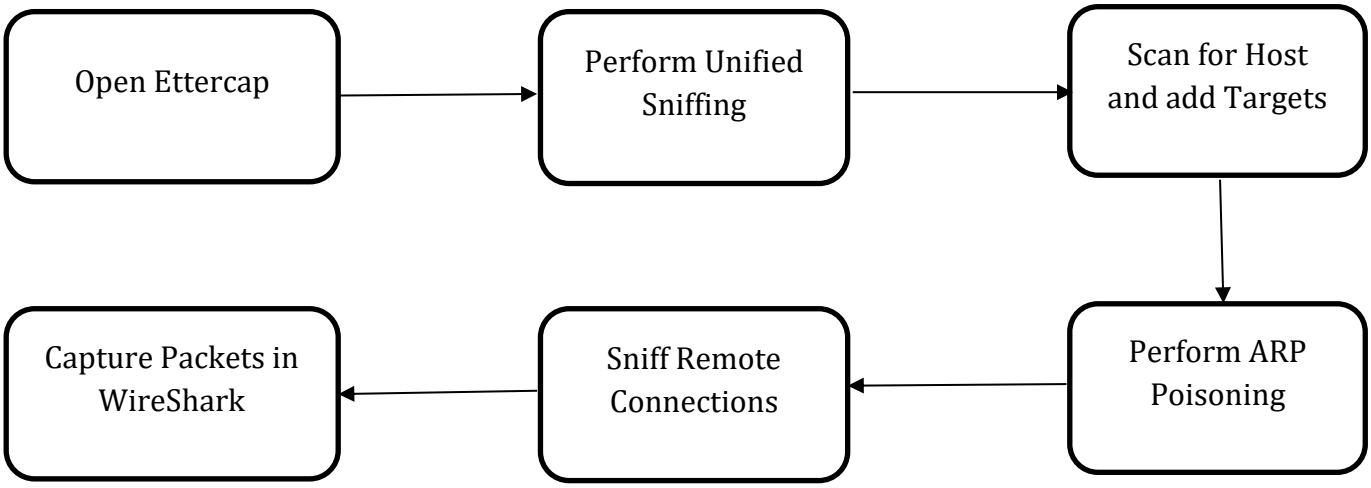


Figure 14: ARP poisoning using Ettercap [31]

Burp Suite: Burp Suite is an open source security tool used for performing and testing security features in web applications. In this study, Burp Suite is used to capture the flow of data between the ESP32 NodeMCU and the ThingSpeak server and later to modify the communicated data. This is done by configuring the web browser in the adversarial system as a proxy server to route packets to Burp suite.

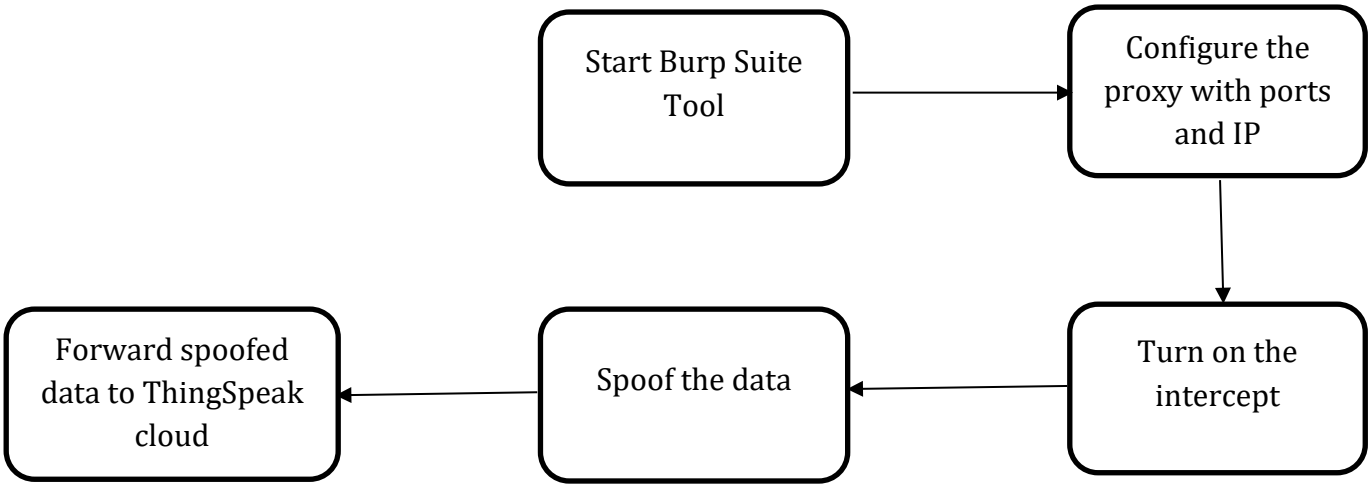


Figure 15: Sensors' data interception and modification using BurpSuite [31]

In Figure 15, data packets transferred between the ESP NodeMCU and ThingSpeak are routed to enter burp suite. Once the data packets are in burp suite, the sensors' records will be modified, and later, the modified sensors' readings will then be forwarded to the ThingSpeak server as if these sensors' records came directly from the ESP32 NodeMCU client.

Using steps and Kali Commands depicted form Figure 16 – Figure 26, a total of 510 'Attacked' records were generated for this study using an adversarial system. The Adversarial system used for this study is a DELL® Core-i3 laptop which runs on Kali Linux with a processor speed of 2.1GHZ and a total installed RAM of 4GB.

The following screen shots depict how the sniffing, ARP poisoning, and data modification was done in this study:

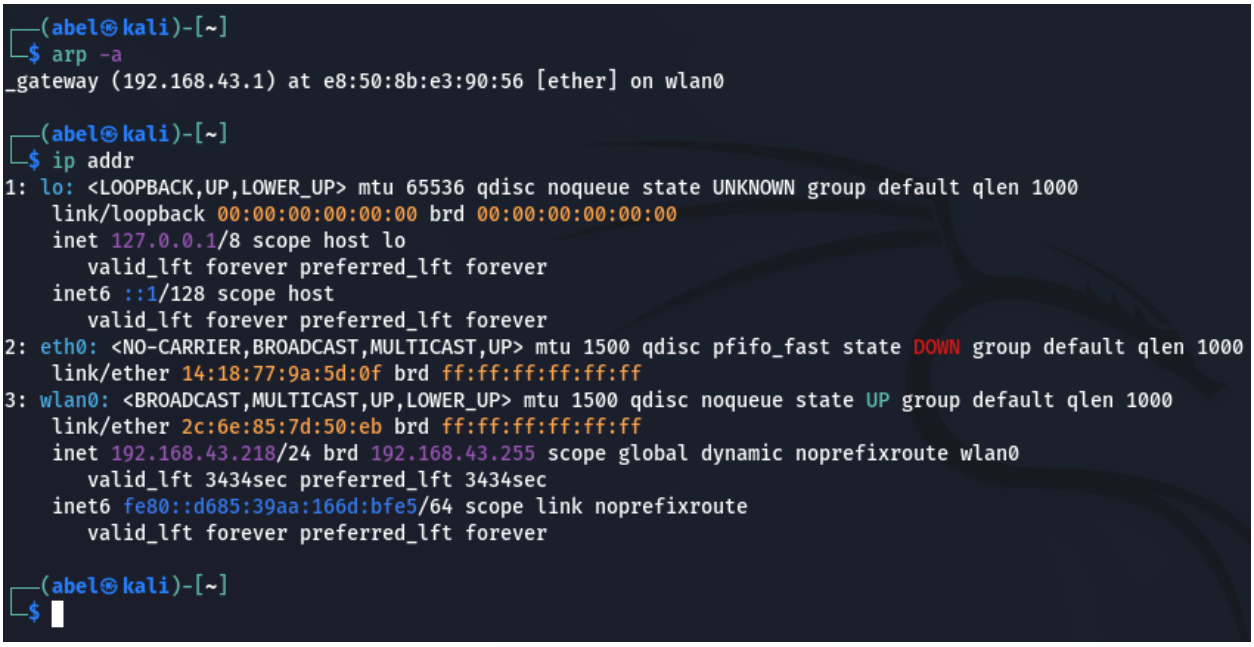


Figure 16: IP and MAC addresses of the default router and the Adversarial System

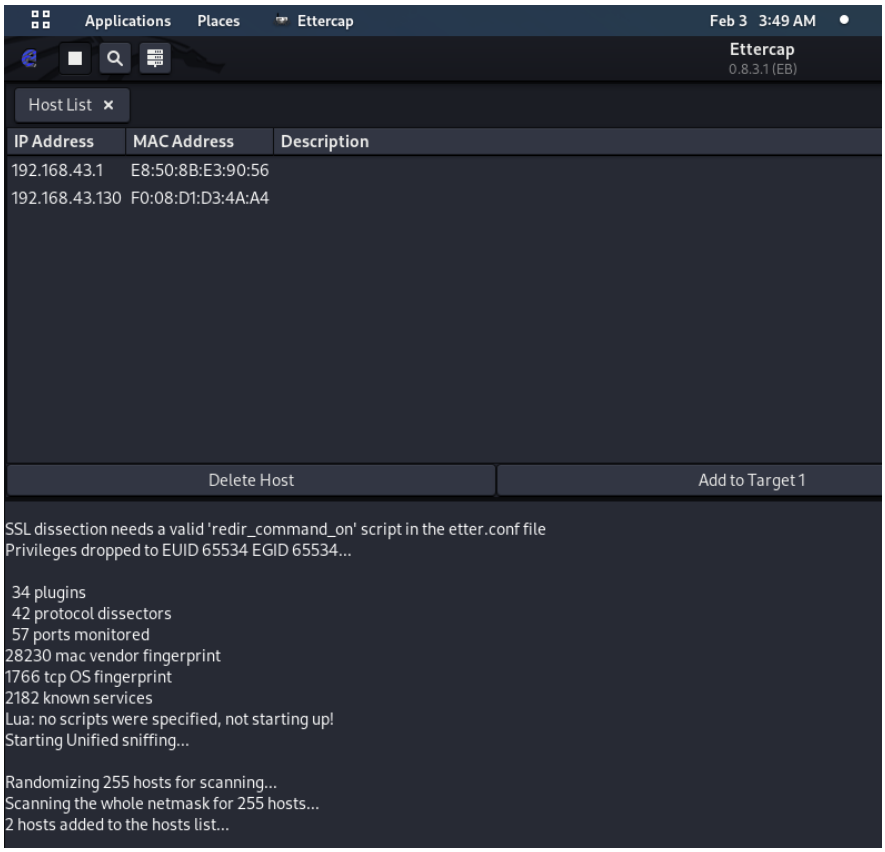


Figure 17: Host List in the IoT testbed

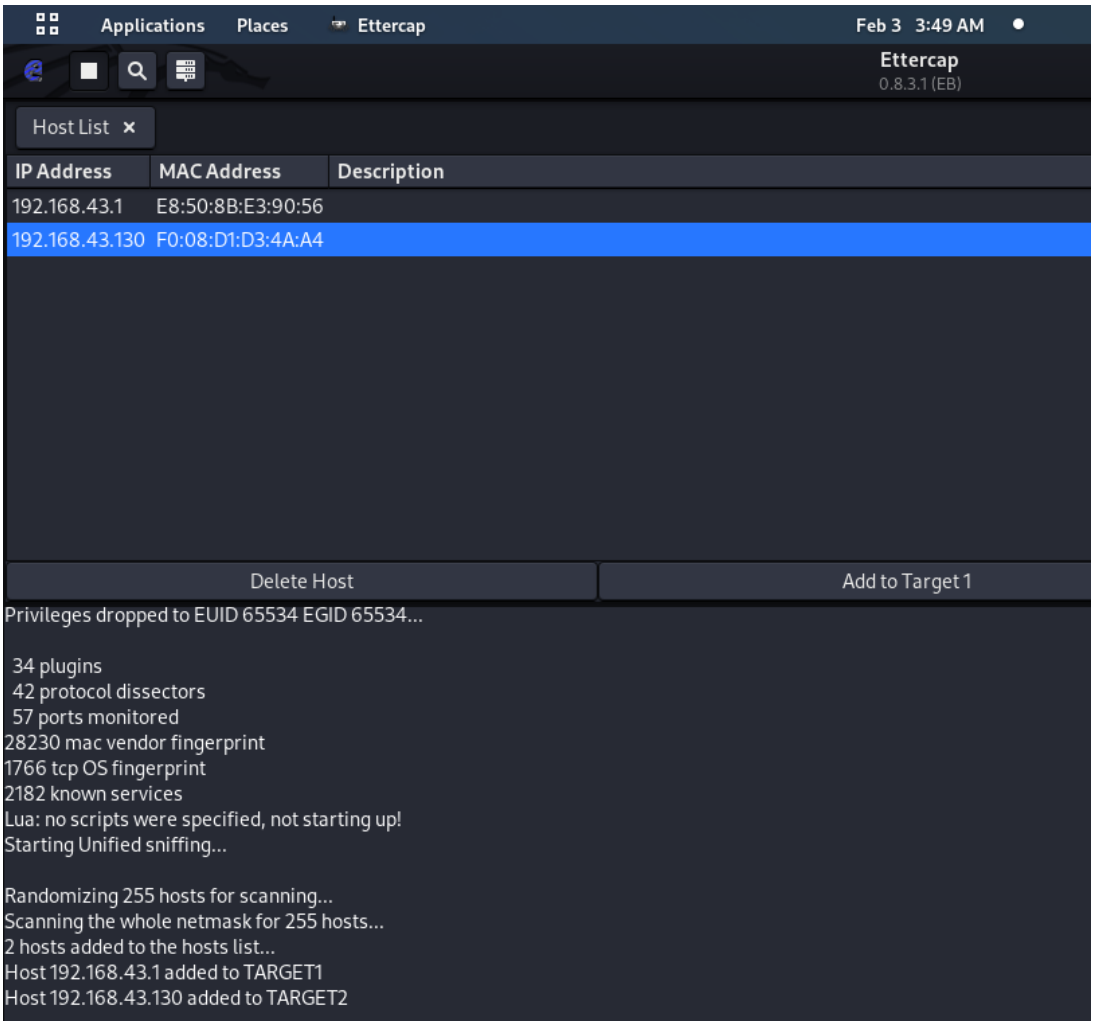


Figure 18: Adding Hosts to Target1 and Target2

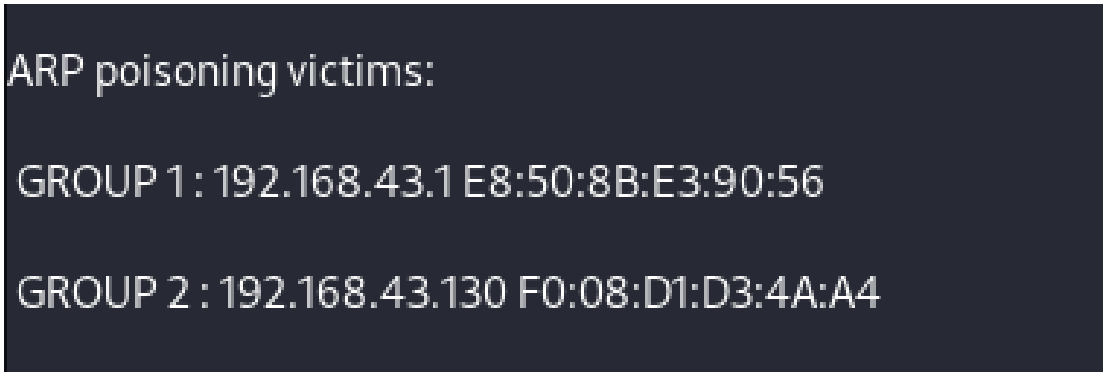


Figure 19: ARP Poisoning Victims

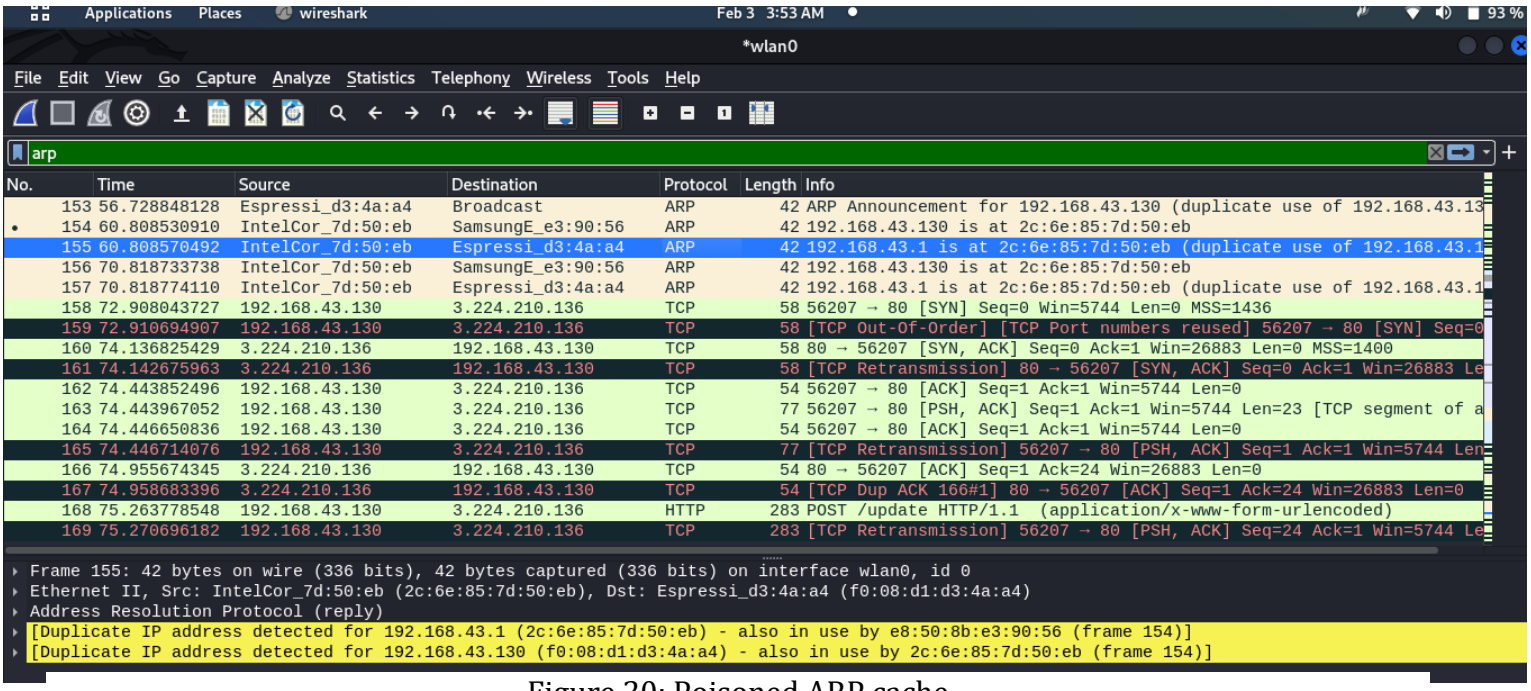


Figure 20: Poisoned ARP cache

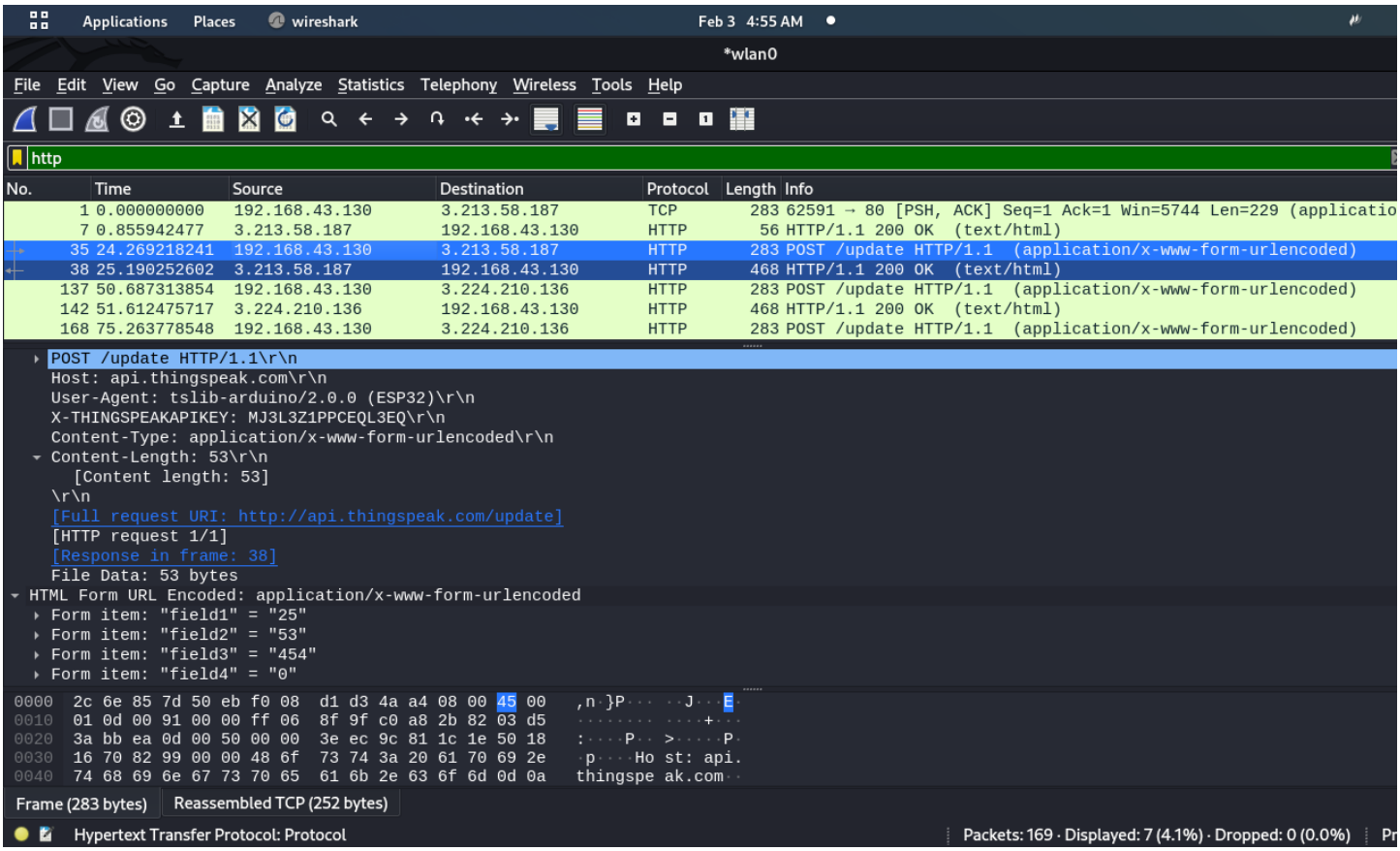


Figure 21: Sniffed Sensors' data

Figure 22: Setting Proxy Port and Destination Port for BurpSuite

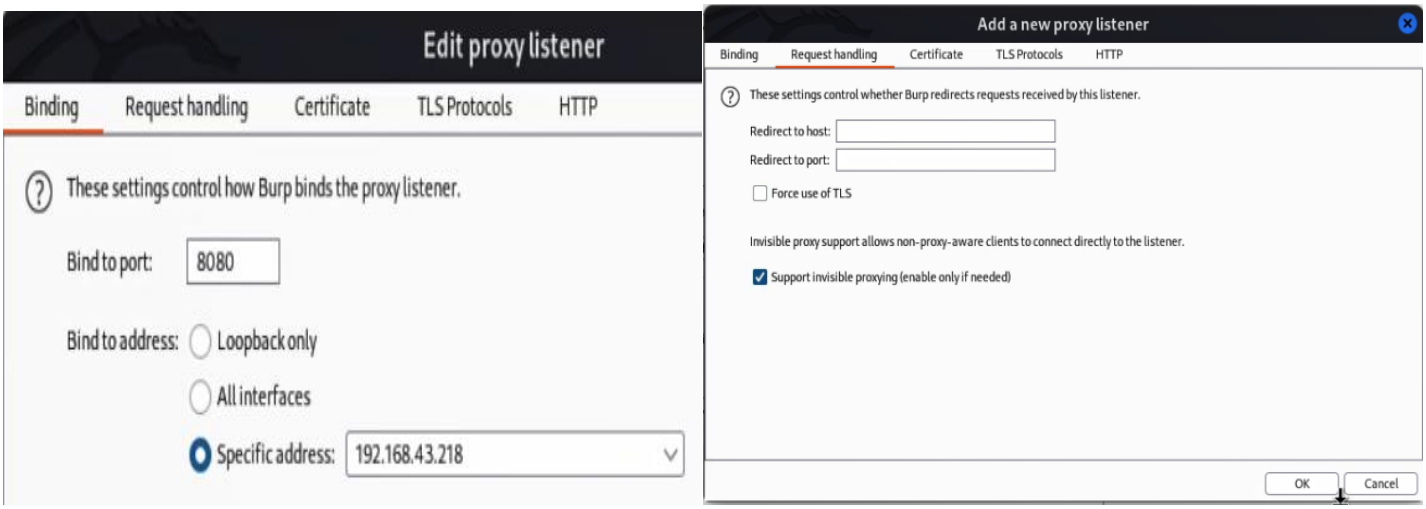


Figure 23: Setting up the BurpSuite Proxy listener Address

Figure 24: Writing Kali Commands to Forward HTTP requests to BurpSuite

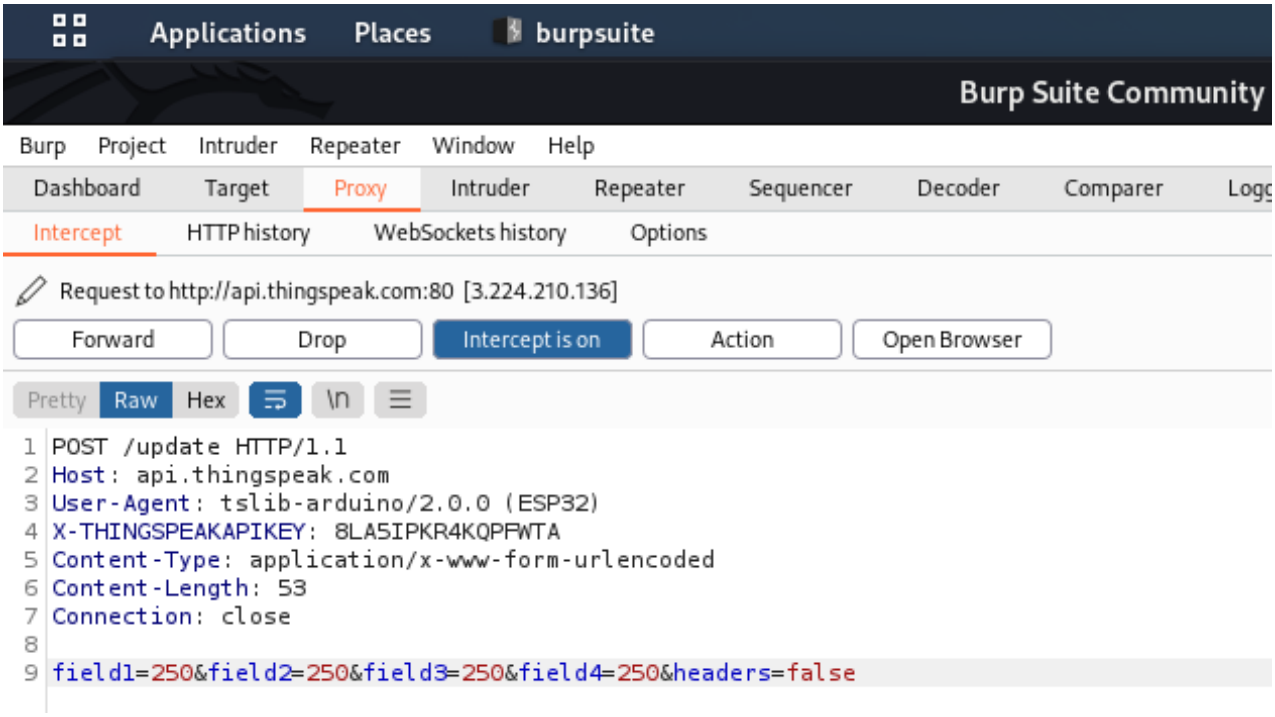


Figure 25: Forwarding BuprSuite Modified Sensors' readings to ThingSpeak Server

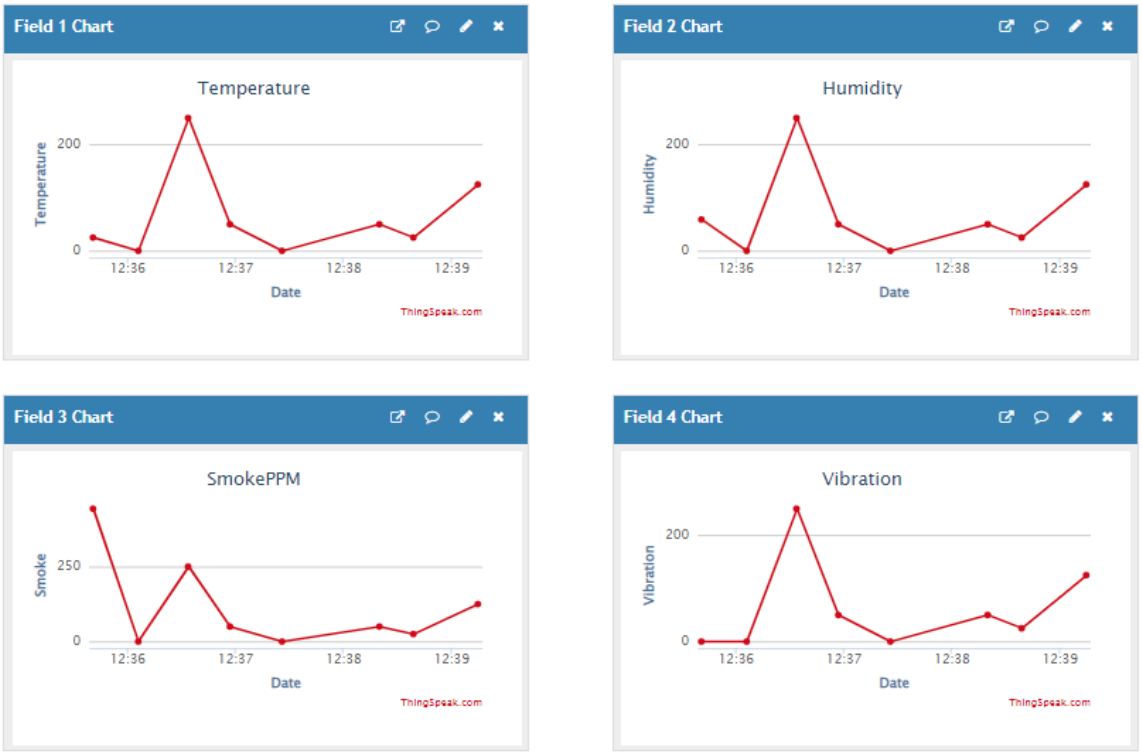


Figure 26: Altered and Modified Sensors' Data on ThingSpeak Server

3.5 Data Analysis Methods

In this study, five potential classifier supervised machine learning algorithms are used for identifying the best algorithm in detecting MitM attacks in IoT networks based on the IoT network's sensors' records. The algorithms used in this study are: SVM, Naïve Bayes, Decision-Trees, KNN, and the Adaboost algorithm. The aforementioned potential algorithms were chosen for this study as they are the most widely used classification algorithms and were recommended in the literature review section of this study.

These Five supervised classifier algorithms are compared to one another using standard confusion matrix metrics. This is achieved by generating a confusion matrix for each classifier. Using the confusion matrix, performance measures of each model is calculated. Figure 27 shows the evaluation metrics of the classifiers considered in this study.

Measure	Formula
Accuracy	$\frac{TP+TN}{P+N}$
Error rate	$\frac{FP+FN}{P+N}$
Precision	$\frac{TP}{TP+FP}$
Sensitivity (Recall)	$\frac{TP}{P}$
Specificity	$\frac{TN}{N}$
F1	$\frac{2*(Precision*Recall)}{(Precision+Recall)}$
Detection Rate	$\frac{TP}{TP+FN}$
False Alarm Rate	$\frac{FP}{TN+FP}$

Figure 27: Performance metrics for classifiers [31]

In this study, TP represents the number of records which have been correctly classified as 'Normal' records, TN represents the number of records correctly classified as 'Attacked' records, FP represents the number of Misclassified records as 'Normal' while in reality, these records were 'Attacked' records, and FN represents the number of Sensors' records Misclassified as 'Attacked' while in reality, they were 'Normal' Sensor's records.

3.6 Validity and Reliability

3.6.1 Research Validity

Research validity is used to determine whether a research truly measures what it was intended to measure or how truthful the research results are [40]. To validate this study, the best machine learning model will be tested if it accurately differentiates sensors' records as either 'Attacked' or 'Normal'. The validity of this study is measured by testing the accuracy of this study's best machine learning model in predicting whether an IoT network's Sensor's record is either 'Normal' or 'Attacked'.

3.6.2 Research Reliability

Research reliability is the extent to which research results are consistent over time [40]. Research reliability also checks if the results of a research can be reproduced under a similar methodology, and if the result is reproduced, the research is considered to be reliable [40]. To check the reliability of this study, each process used in this study is briefly described as well as all hardware and software components used in the study are clearly depicted in a pictorial manner. In addition the NodeMCU's firmware written to extract data from the 3 sensors and sending these sensors' records to the ThingSpeak cloud is given at the appendix section of this study.

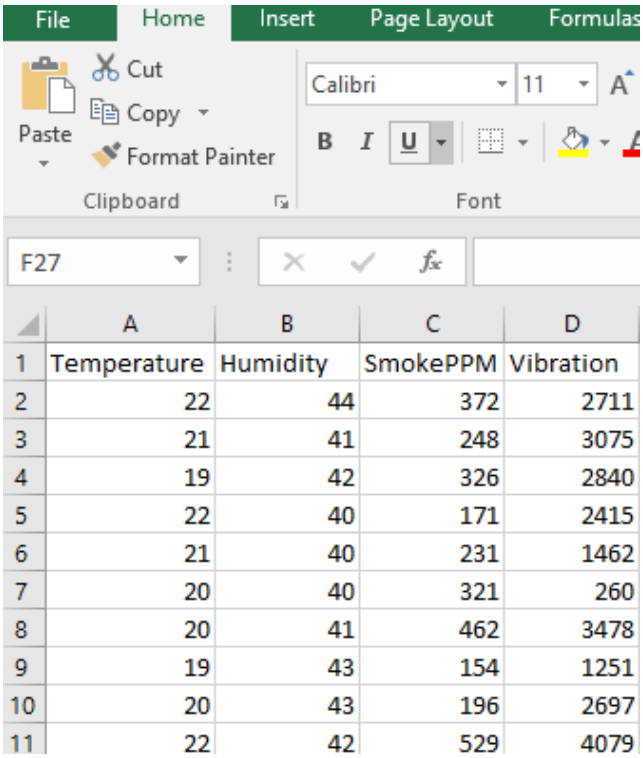
3.7 Communication

According to [41], communication is the last core phase of a design research study. The primary goal of communication is communicating the research problem and the designed artifact to other researchers and scholars interested in the research area. This study communicates the research area to Saint Mary's University via this research paper.

3.8 Dataset Used in this Study

To compare the 5 supervised machine learning classification algorithms, both 'Normal' data and 'Attacked' data were used. In this phase, both the Normal records (Non-Attack data) and Attacked sensors' records were downloaded from the ThingSpeak server separately in a .csv format for the sole purpose of training and testing the candidate ML models.

Using these datasets, five candidate Machine Learning models are generated using the datasets. The five candidate algorithms used in this study are: SVM, Naïve Bayes, Decision-Trees, KNN, and Adaboost.



	A	B	C	D
1	Temperature	Humidity	SmokePPM	Vibration
2	22	44	372	2711
3	21	41	248	3075
4	19	42	326	2840
5	22	40	171	2415
6	21	40	231	1462
7	20	40	321	260
8	20	41	462	3478
9	19	43	154	1251
10	20	43	196	2697
11	22	42	529	4079

Figure 28: Sample DataSet

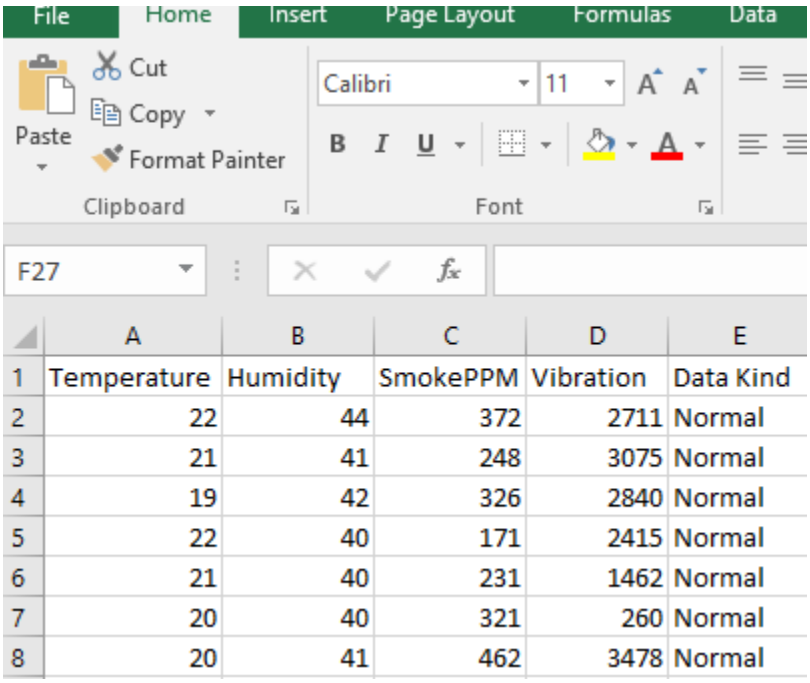
In this study, a total of 1,510 Normal records were generated and downloaded from the ThingSpeak server. Besides the 'Normal' sensors' records, additional 510 Attacked and spoofed records were downloaded from the ThingSpeak cloud. Of each record, 10 records were cutoff form each record in the training and testing phase of the ML algorithms. The reason being; these 20 sensors' records are later used as prediction records for the purpose of testing the accuracy of the best ML model in "Labeling" these sensors' records as either 'Normal' or 'Attacked'.

Hence, a total of 2000 (Two thousand) Original records were used to train and test the five candidate algorithms.

3.9 Data Preprocessing and SMOTEing

Once the sensors' records were downloaded from the cloud, it was imperative to label sensors' records as 'Normal' and 'Attacked' for the purpose of training and testing the supervised ML algorithms. Hence, each record was classified as either 'Normal' or 'Attacked' under an attribute named "Data Kind". As their name vividly implies, records labeled 'Normal' are records that are NOT tampered with by the Adversarial system. On the other hand, records labeled 'Attacked' are records that are tempered by the Adversarial system in the IoT Testbed. Later before the records were fed to the Weka Explorer, each numeric value was changed to nominal using the Weka Explorer.

As clearly stated in section 3.8 of this study, a total of 2,000 records were generated and collected from the ThingSpeak cloud. Of the 2,000 records, 1,500 records were 'Normal' records and the remaining 500 are 'Attacked' records. Nevertheless, for the sake of balancing the data and increasing the amount of records SMOTE technique was used in this study. SMOTE (Synthetic Minority Oversampling Technique) is a statistical technique used for increasing the number of records in a given dataset in a synthetic and balanced manner. Hence once the data was SMOTEd, a total of 4,000 records were generated. Amongst which 2000 are 'Normal' records and the remaining 2000 are 'Attacked' sensors' records.



	A	B	C	D	E
1	Temperature	Humidity	SmokePPM	Vibration	Data Kind
2	22	44	372	2711	Normal
3	21	41	248	3075	Normal
4	19	42	326	2840	Normal
5	22	40	171	2415	Normal
6	21	40	231	1462	Normal
7	20	40	321	260	Normal
8	20	41	462	3478	Normal

Figure 29: Labeled Dataset

3.10 Detail about the Experimentation

The following sections describe how each experiment is conducted and every step involved in the Weka Explorer in a very detailed manner.

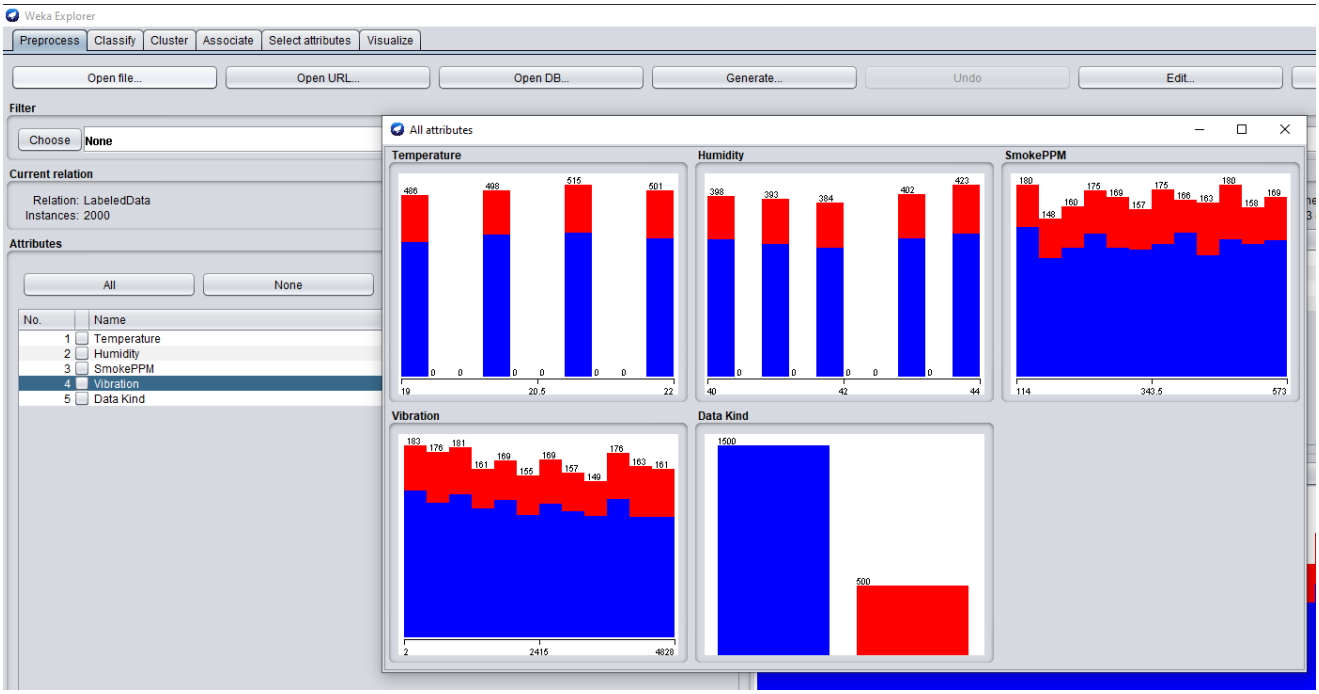


Figure 30: Original Dataset Used in this Study with Attributes

As vividly depicted in Figure 30, the dataset is comprised of five attributes; namely 'Temperature', 'Humidity', 'Smoke PPM (Part-Per-Million)', 'Vibration', and 'Data Kind'. The first four attributes are self-explanatory but the last attribute i.e., 'Data Kind' is used to label sensors' records as 'Normal' and 'Attacked'. On the upper left corner of figure 30, it is vividly depicted that at this point, the dataset is comprised of just 2,000 records as the data has not been SMOTEd yet.

The next step taken in this study was performing preprocessing on the dataset and balancing the dataset.

3.11 Preprocessing the data:

The original data format of the Attacked and Normal data downloaded from the ThingSpeak cloud was in a numeric format also the ratio of ‘Normal’ sensors’ records to ‘Attacked’ sensors’ records was 3:1. Hence, primarily the Numeric records were converted to their Nominal counterparts. The dataset also needed to be resampled to balance the ‘Attacked’ records and the ‘Normal’ sensors’ records. Figures 31 and 32 do exactly that respectively.

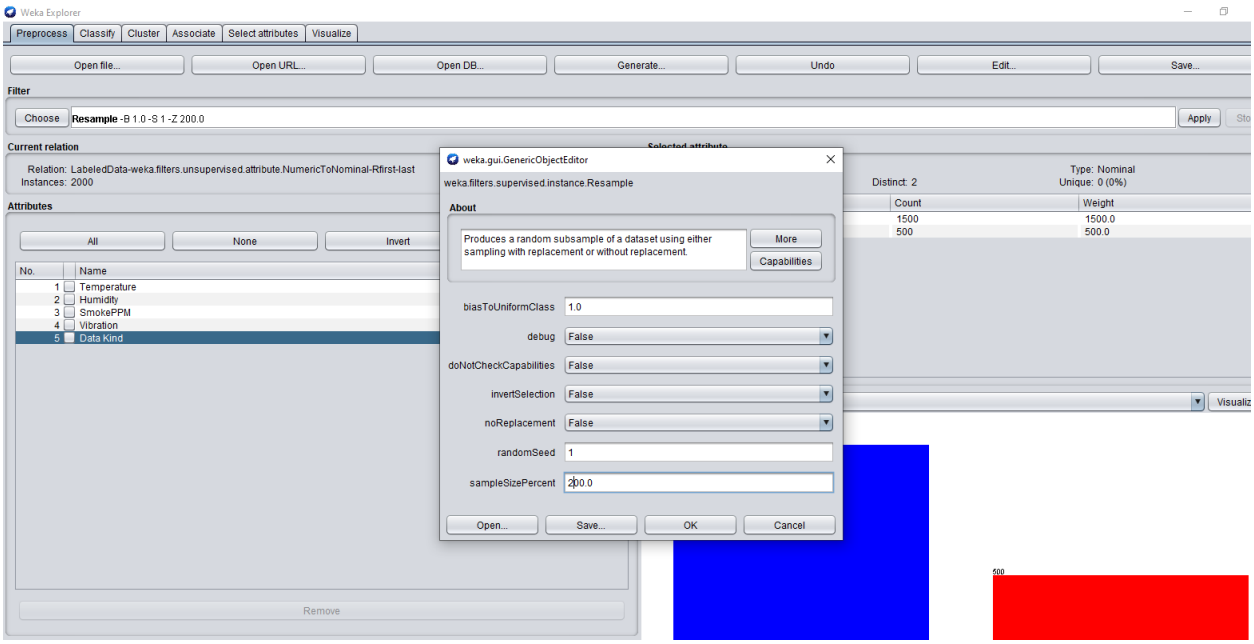


Figure 31: *Tuning Parameters to double size of Dataset*

After being preprocessed, all numeric values were converted to nominal values using built-in tools of the Weka explorer and the ration of 'Normal' sensors' records to 'Attacked' sensors' records have now been balanced as depicted in Figure 32.

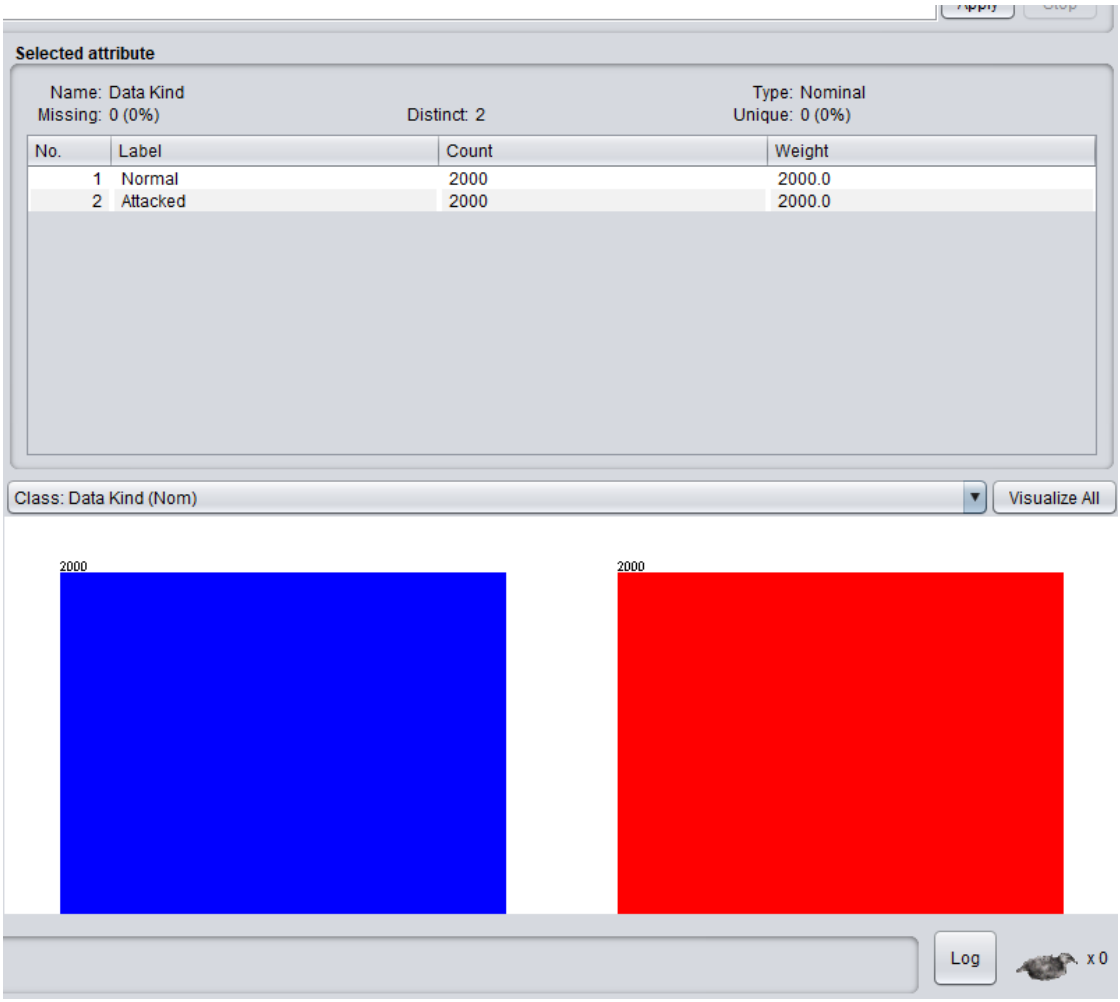


Figure 32: Normal and Attacked Sensors' records after being balanced

As depicted on Figures 31, 32, and 33, the dataset has now been converted to nominal, doubled, and the ‘Normal’ sensors’ records and ‘Attacked’ sensors’ records have been balanced; hence, the ML models will be trained better.

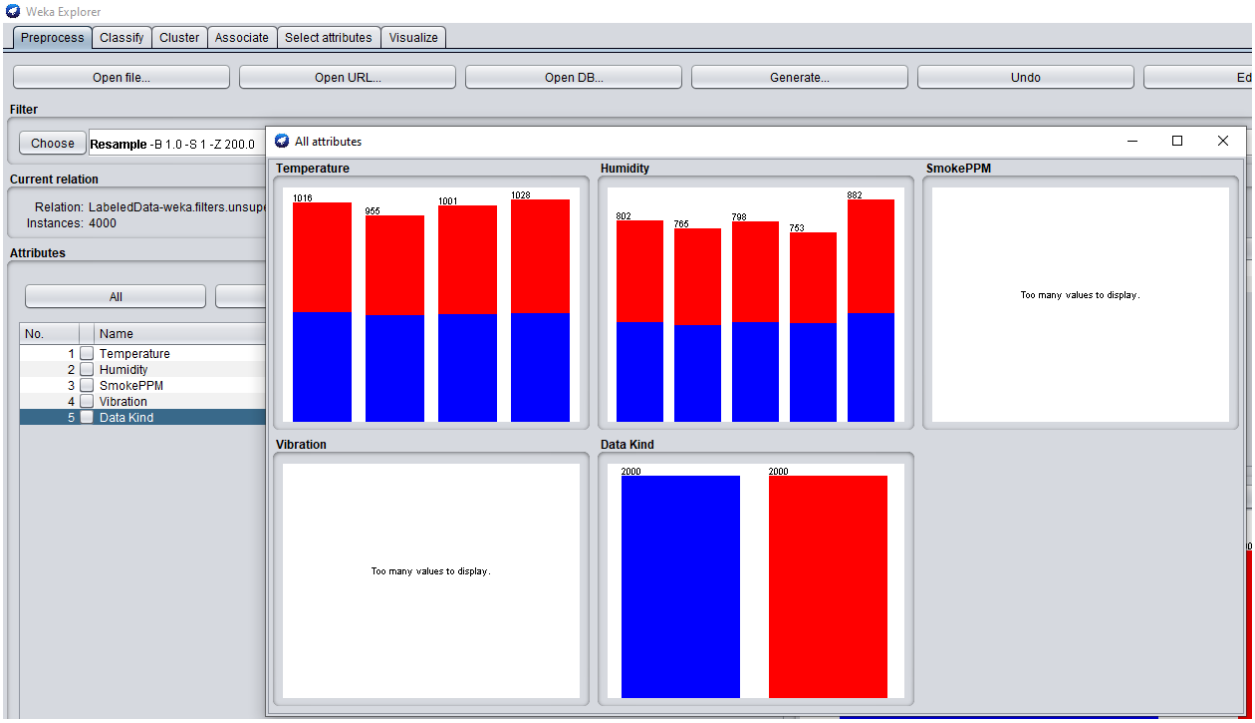


Figure 33: Dataset after being doubled and Balanced

3.12 Cross-Validation of the Dataset

For the sake of creating good machine learning models and preventing both Over-Fitting and Under-Fitting, Cross-Validation was used during the training phase of each candidate classifier algorithm. And the number of Folds or 'K' during the training of all candidate algorithms used was the default value of 10.

3.13 Experiments:

The following sections show how each candidate classifier algorithms performed in differentiating the 'Normal' sensors' records from 'Attacked' records with the generated dataset. The 5 Candidate algorithms used in this study are: SVM, Naïve Bayes, Decision-Trees, KNN, and Adaboost algorithms. Performance of each candidate algorithm are given below.

3.13.1 Experiment using the SVM Algorithm

Using the SVM algorithm, the results appearing in Figure 34 has been found.

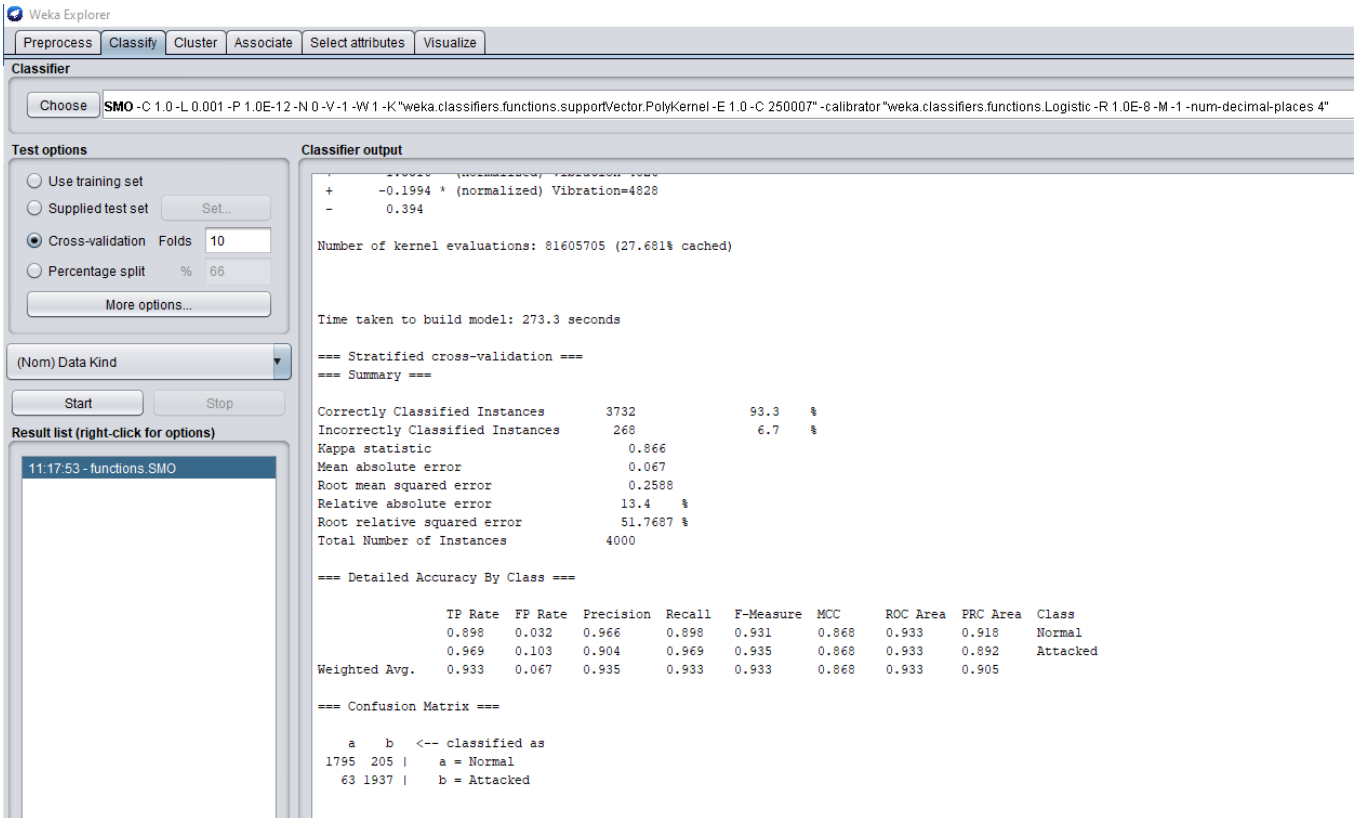


Figure 34: Experiment using the SVM Algorithm

3.13.2 Experiment Using the Naïve Bayes Algorithm

Using the Naïve Bayes algorithm, the results appearing in Figure 35 has been found.

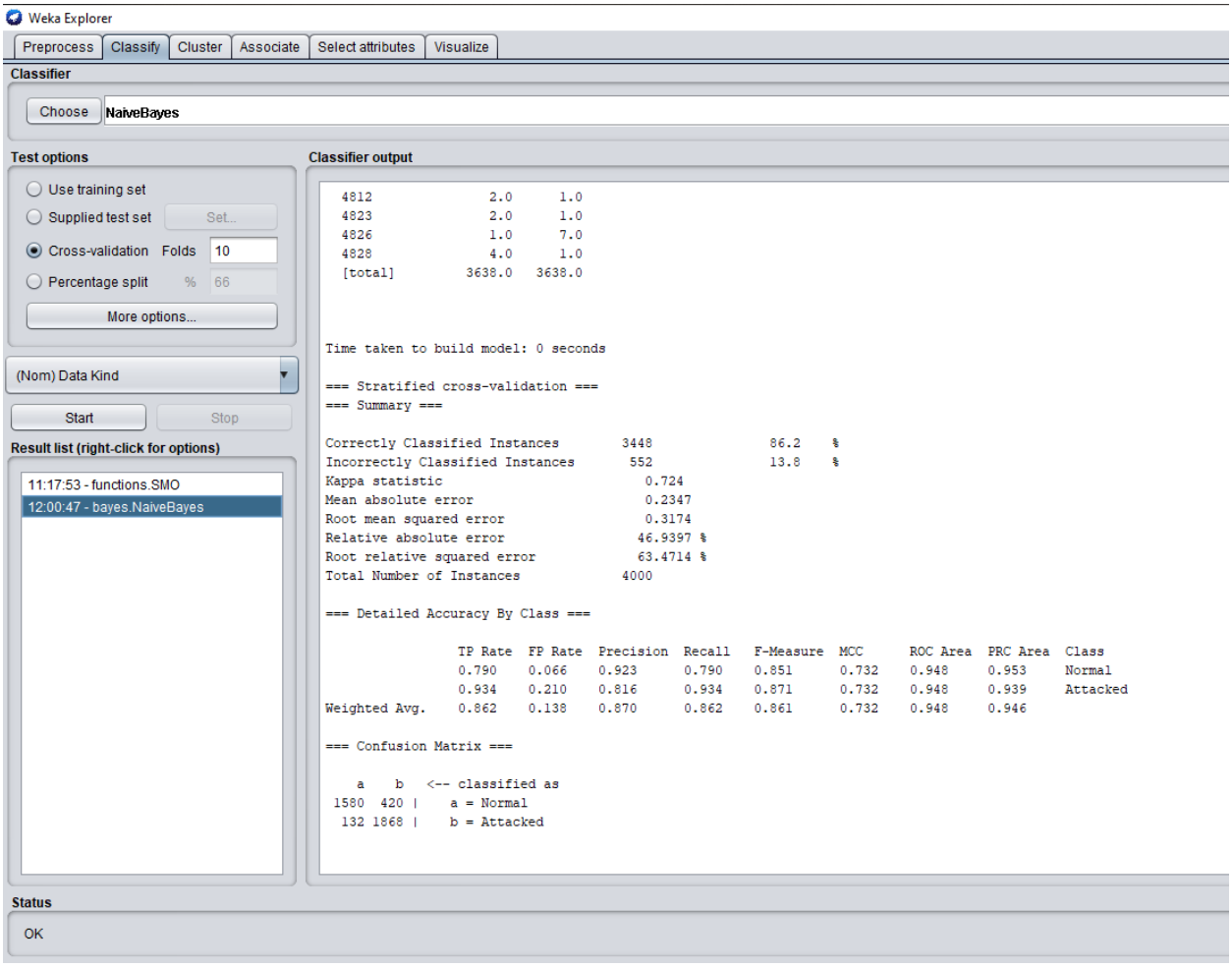


Figure 35: Experiment Using the Naïve Bayes Algorithm

3.13.3 Experiment using the Decision-Trees Algorithm

Using the Decision-Trees algorithm, the results appearing in Figure 36 has been found.

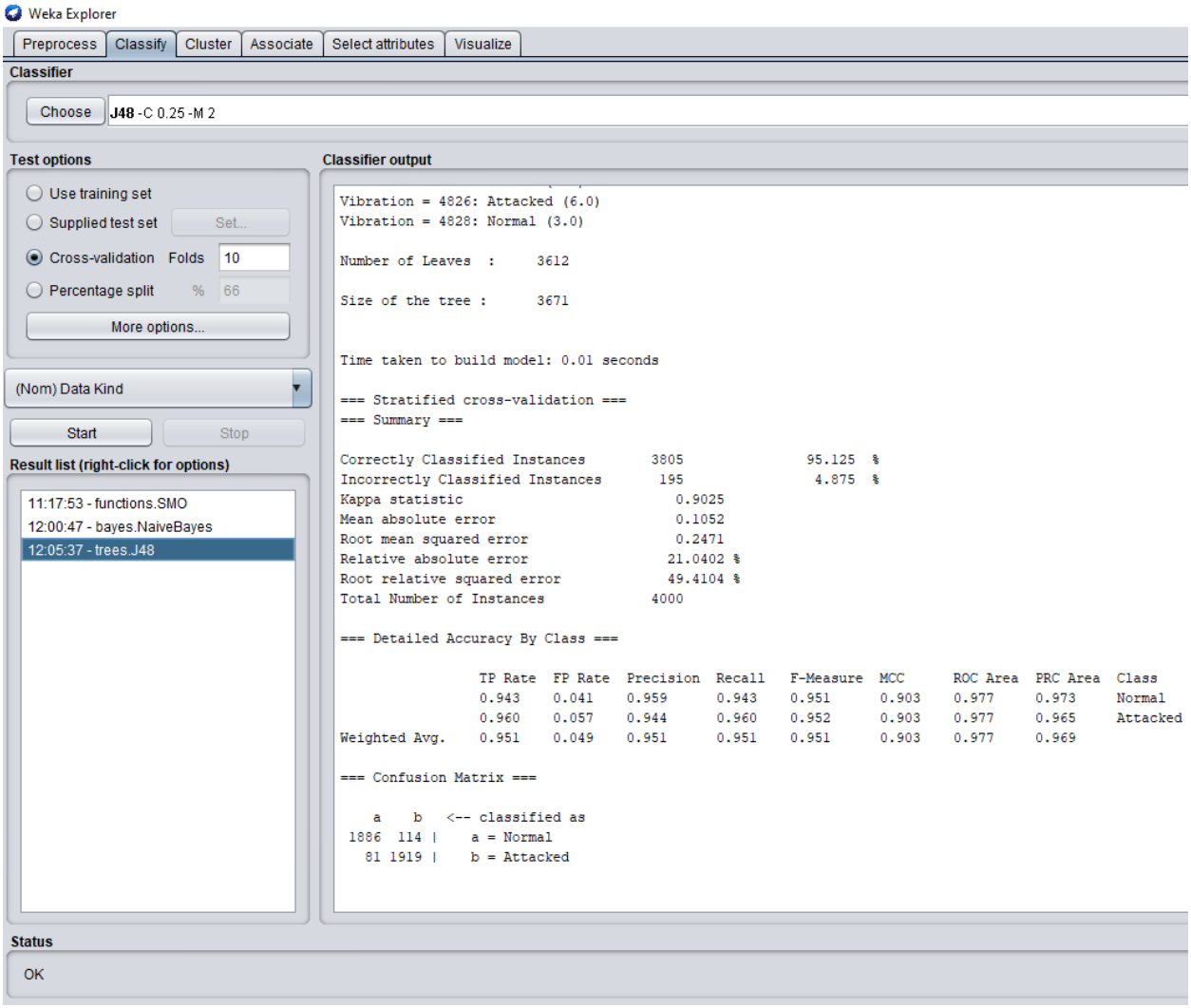


Figure 36: Experiment using Decision-Trees

3.13.4 Experiment using the KNN Algorithm

Using the KNN algorithm, the results appearing in Figure 37 has been found. While using the KNN algorithm, the value 63 was used as the value of K.

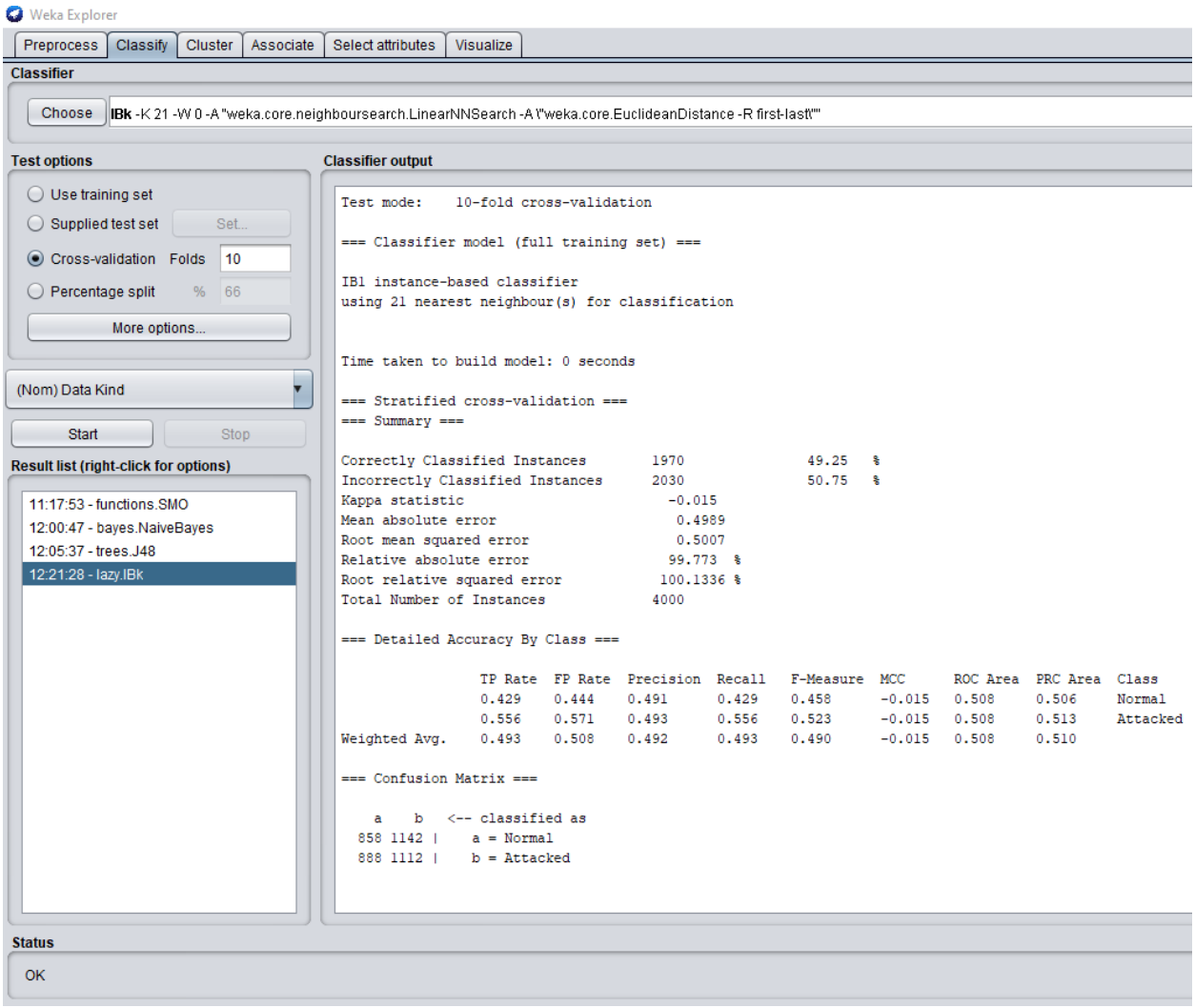


Figure 37: Experiment Using the KNN Algorithm

3.13.5 Experiment using the Adaboost Algorithm

Using the Adaboost algorithm, the results appearing in Figure 38 has been found.

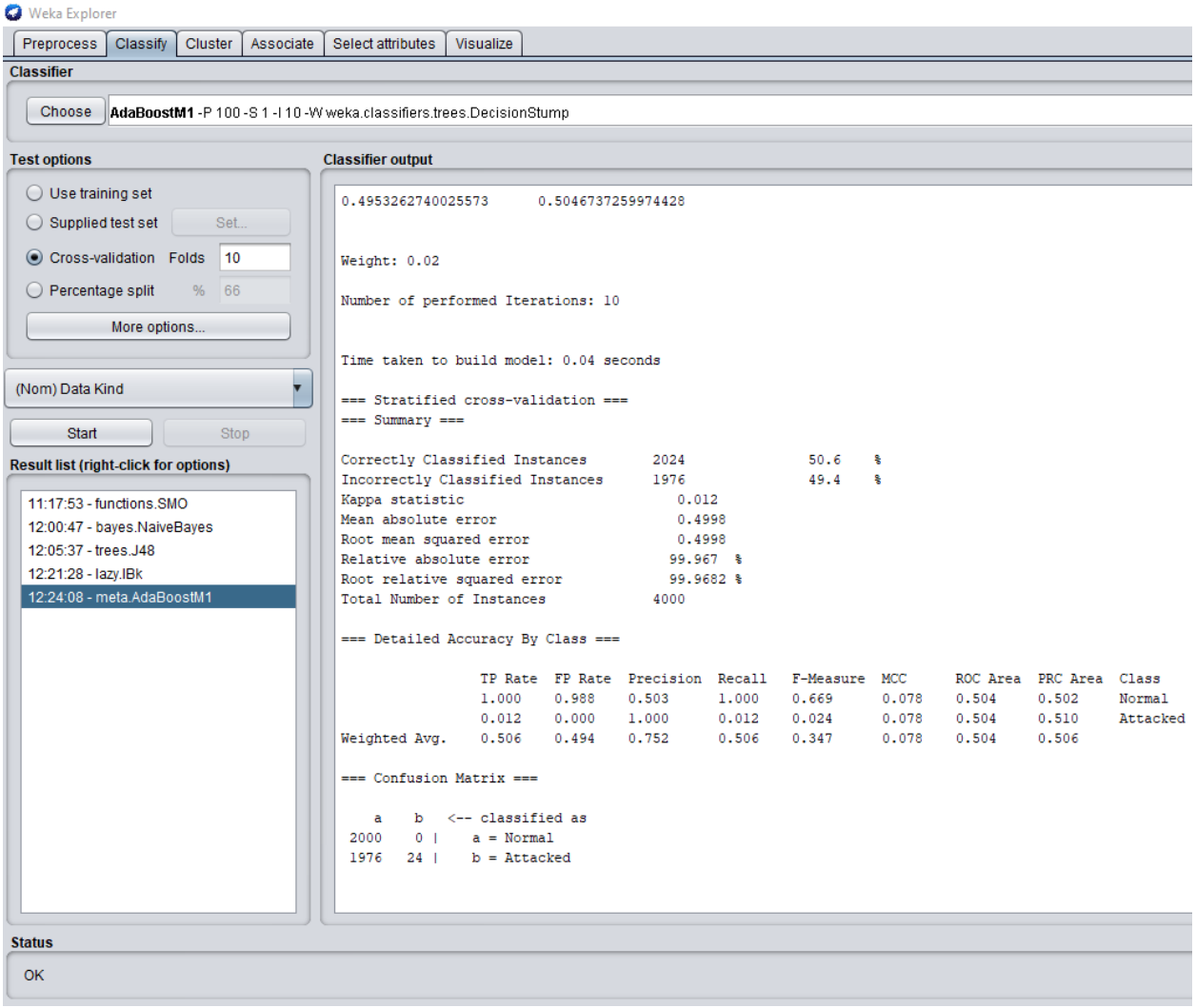


Figure 38: Experiment using the Adaboost Algorithm

3.14 Comparison of Experiments

To compare the results obtained from the five classifier algorithms in differentiating the 'Normal' sensors' records from 'Attacked' records, the accuracy of each algorithm is compiled to Table 1 of this study.

Hence, according to Table 1, of the five potential algorithm used in this study with the given dataset. Decision-Trees turned out to be the one that is most accurate and KNN turned out to be the one with the lowest accuracy in differentiating 'Normal' IoT network's sensors' records form 'Attacked' sensors' records.

Chapter Four

Demonstration and Evaluation

4.1 Evaluation of Experiments

According to [41], the demonstration and evaluation phase is used to demonstrate, test, observe, and evaluate the extent to which the designed artifact provides solution to the identified research problem. In this study the five supervised classifier algorithms are tested using the experimental setting listed in Table 1. And each Algorithm is evaluated as per its accuracy in differentiating 'Normal' IoT network's sensors' records from 'Attacked' records.

Algorithm	Accuracy in %
SVM	93.3 %
Naïve Bayes	86.2 %
Decision Trees	95.125 %
KNN	49.25 %
Adaboost	50.6 %

Table 1. Evaluation of Algorithms

4.2 Demonstration

As vividly stated on Table 1, the decision-Tree algorithm is the one with the highest level of accuracy hence, the decision-tree's algorithm's ML model was used to predict the attribute 'Data Kind' using the 20 records excluded from being used in training & testing the ML models as discussed in section 3.8 of this study. Hence, the 20 excluded sensors' records were used to test the prediction accuracy of the Decision-Tree's machine learning model.

To test the accuracy of prediction of the decision-tree's ML model in accurately predicting the 'Data Kind' attribute of the 20 sensors' records, these sensors' records were deliberately excluded from the training dataset for this sole purpose.

Of the 20 records, the first 10 IoT network's sensors' records are of type 'Normal' while the later are of type 'Attacked'. Using these records, the Decision tree ML model's predicted output is shown in figure 39.

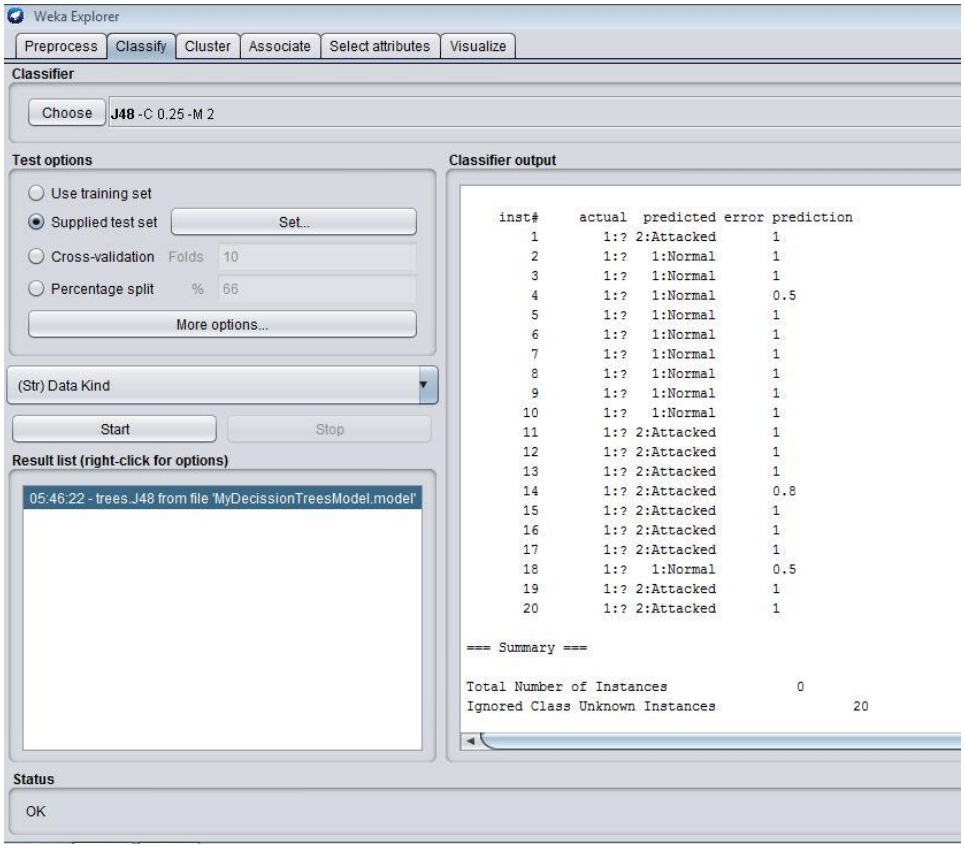


Figure 39: Predicted Outcome using the Decision Tree's Model

As depicted in figure 39, the decision-tree's model was able to accurately predict 18 sensors' records based on the provided test records. Of the 20 sensor's records, the Decision trees ML model was able to accurately predict 18 and incorrectly predicted 2 records; one from each record kind.

Chapter Five

Conclusion and Recommendation

5.1 Conclusion

This study focuses on building a ML model that detects Man in the Middle attack in IoT networks using machine learning techniques based on the IoT network's sensors' records. This study have discussed the background, application, ubiquity, limitations and use of IoT applications in our day to day lives. Because of resource constraints inherent to the IoT network's microcontrollers and vulnerabilities that arise due to their wireless communication medium, this study also discussed that it is not wise to use traditional cyber security mechanisms like encryption as a counter measure to combat MiTM attacks in these resource constrained IoT devices. Hence using their data as input for security analysis together with machine learning techniques was chosen as an efficient alternative to detect MitM attacks in the IoT network.

For this research, for the purpose to acquiring training and testing datasets, an IoT testbed comprised on an ESP32 NodeMCU IoT Module, a DHT22 Temperature & Humidity sensor, An MQ2 gas sensor, an SW-420 Vibration sensor, and an Adversarial system with Kali Linux was used. Once the IoT testbed was developed, both Normal data and Attack data were generated for training and testing the machine learning models.

Using the dataset from the aforementioned IoT testbed five candidate supervised classifier algorithms were used to differentiate between 'Normal' data and 'Attacked' data. Of the five algorithms decision-trees came out to be the one with the highest accuracy of 95.125%.

Hence this study concludes that even if we do not deploy traditional cyber security mechanisms like encryption to sustain the integrity of communicated data in these power and resource constrained IoT devices, it is still possible to use machine learning techniques in verifying whether sensors' records coming from these IoT networks is either legitimate or tampered with using the Man-in-the-Middle attack using ARP cache poisoning.

This study shows that it is possible to actively attack and modify sensors' reading in an IoT network using ARP poisoning. Hence, unless machine learning models like the one used in this study lie in the middle of the IoT network and a central server or database located in a cloud, sensors' readings from such IoT networks should never be used to control actuators embedded in Mission-Critical infrastructures as they might cause real-time real-life catastrophes.

5.2 Recommendation

IoT devices are generally designed to be connected to multiple sensors at any given time and in this study, only three sensors were used in developing the IoT testbed. These sensors are the DHT22 Temperature and Humidity sensor, An MQ2 gas sensor and a SW-420 Vibration sensor. These three sensor combined had a total of four sensor outputs namely; Temperature, Humidity, Smoke in Parts-Per-Million, and level of vibration.

Therefore, increasing the number of sensor in the IoT testbed or increasing the number of sensed attributes in the IoT testbed is a possible area for future thesis work. Another possible area for future work is increasing the amount of sensors' data and develop a machine learning model using artificial neural networks.

References:

[1] IBM, 15, July 2020, "Machine Learning" [online]. Available: https://www.ibm.com/cloud/learn/machine-learning

[2]Statista, 2022, "Internet of Things (IoT) and non-IoT active device connections worldwide from 2010-2015" [online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[3] Meng, Y.; Zhang, W.; Zhu, H.; Shen, X.S, "Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures" IEEE Wirel. Commun. 2018, 25, 53–59. [CrossRef]

[4] Xiao, Liang, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. (2018) "IoT security techniques based on machine learning." *arXiv preprint arXiv:1801.06275*

[5] Abd El-Monem A. El-Bawab "Untangle Network Security" 2014, pp 26

[6]ZDNet, February 3, 2020, "What is IoT? Everything you need to know about the internet of Things right now" [online]. Available: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

[7] Oracle, 2022, "What is IoT" [online]. Available: https://www.oracle.com/internet-of-things/what-is-iot/

[8] IoT Agenda, Alexander S. Gillis, "What is internet of things (IoT)?" [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[9] Aeris, "What is IoT? Defining the Internet of Things (IoT)" [online]. Available: https://www.aeris.com/in/what-is-iot/

[10] IBM, November 17, 2016, Jen Clark "What is the Internet of Things"? [online]. Available: https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/

[11] Dogan Ibrahim, "Advanced Microcontroller projects in C, From USB to ZIGBEE with the PIC 18F series" 2008, pp 17-19

[12] Evan Randall, "End to End Large scale Machine learning with Keystone ML" 2016, pp 13-15

[13]Analytics Vidhya, September 9, 2017, Sunil Ray, "Commonly used Machine Learning Algorithms (With Python and R Codes)" [online]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

[14] Francois Chollet, "Deep Learning with Python", 2014, pp 16-17

[15] Peter Harrington, "Machine Learning in Action", 2012, pp 130-132

[16] Toby Sergan, "Programming collective intelligence Building Smart Web 2.0 Applications", 2007, pp 29-30

[17] Feng Niu et al. “DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference.” In:VLDS12 (2012), pp. 25–28.

[18] Micro.ai, November 12, 2019, “10 Types of Cyber Security Attacks in IoT” [online]. Available: https://www.micro.ai/blog/10-types-of-cyber-security-attacks-in-the-iot

[19] Kevin Lam, David LeBlanc, and Ben Smith “Assessing Network Security”, 2004 pp 301-310

[20] Eric Cole, “Network Security Bilble” 2009, second edition, pp 332

[21] Ross J. Anderson, “Security Engineering a guide to building dependable distributed systems”, 2008, pp 635

[22] Imperva, “What is the ARP Protocol” [online]. Available: https://www.imperva.com/learn/application-security/arp-spoofing/

[23] Miller, Lawrence. (2016) "IoT Security for Dummies”, Carrie A. Johnson, Ed.

[24] N. Vlajic and D. Zhou, "IoT as a Land of Opportunity for DDoS Hackers," in *Computer*, vol. 51, no. 7, pp. 26-34, July 2018, doi: 10.1109/MC.2018.3011046..

[25] Andrea, Ioannis, Chrysostomos Chrysostomou, and George Hadjichristofi. (2015) "Internet of Things: Security vulnerabilities and challenges." In *2015 IEEE Symposium on Computers and Communication (ISCC)*, 180-187.

[26] Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. (2018) "IoT security techniques based on machine learning." *arXiv preprint arXiv:1801.06275*

[27] Mohammad Noor Injadat, Adballah Moubayed, Abdallah Shami (2020), “Detecting Botnet Attacks in IoT Environments: An Optimized Machine Learning Approach”

[28] Gao, Weihua, Yuhao Sun, Qingying Fu, Zhouzhe Wu, Xiao Ma, Kai Zheng, and Xin Huang. (2018) "ARP Poisoning Prevention in Internet of Things." In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, 733-736.

[29] Pingle, Bhargav, Aakif Mairaj, and Ahmad Y. Javaid. (2018) "Real-World Man-in-the-Middle (MITM) Attack Implementation Using Open Source Tools for Instructional Use." In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 0192-0197.

[30] Chaabouni, Nadia, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. (2019) "Network Intrusion Detection for IoT Security based on Learning Techniques." *IEEE Communications Surveys & Tutorials*.

[31] K.V.V.N.L Sai Kiran, R.N. Kamakshi Devisetty, N.Pavan Kalyan, K.Mukundini, R.Karthi (2020) “Building a Intrusion Detection System for IoT Environment using machine Learning Techniques” Dept of Computer Science and Engineering, Amirita School of Engineering, Combatore, Amirita Vishwa Bidyapeetham, India

[32] ESPRESSIF, ESP32, "ESP32 Features" [online]. Available: https://www.espressif.com/en/products/socs/esp32

[33] ThingSpeak, "ThingSpeak for IoT Projects" [online]. Available: https://thingspeak.com/

[34] Wikepedia, "Kali Linux" [online]. Available: https://en.wikipedia.org/wiki/Kali_Linux

[35] D-Link, "Wireless N 300 ADSL2+ 4-Port Router" [online]. Available: https://dlinkmea.com/index.php/product/details?det=dU1iNFc4cWRsdUpjWEpETFlSeFlZdz09

[36] Arduino, "Arduino IDE 1.8.19" [online]. Available: https://www.arduino.cc/en/software

[37] WireShark, "WireShark" [online]. Available: https://www.wireshark.org/

[38] Jan vom Brocke, Alan Hevner, Alexander Maedche, Introduction to Design Science Research, 2020

[39] Hevner, A.R., March, S.T., Park, J., & Ram, S. (2004). Design Science in Information Systems. MIS Quarterly, 28(1), 75-105.

[40] Golafshani. (2003). Understanding Reliability and Validity in Qualitative Research

[41] Ken Peffers, Turre Tuunanen, Marcus A.Tothenberger, Samir Chaatterjee. A Design Science Research Methodology for Information Systems Research

[42] Towards data science, September 10, 2018, Onel Harrison "Machine Learning Basics with the K-Nearest Neighbors Algorithm" [online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[43] Waikato, "Weka Knowledge Explorer" [online]. Available: https://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html#:~:text=The%20Weka%20Knowledge%20Explorer%20is,power%20of%20the%20weka%20software

Appendix:

```
/*

This is the firmware written to extract data from the three sensors and send the sensors' readings to the ThingSpeak Server. The three sensors are the DHT22 Temp & Humidity sensor, the MQ2 Gas sensor, and the SW-420 Vibration sensor.

  Author: ABEL ASHENAFI TADESSE

  Date: Jan, 2022

  belaashu23@gmail.com

 */

#include <Adafruit_Sensor.h>

#include "DHT.h"

#include <WiFi.h>

#include "ThingSpeak.h"

#define DHTPIN 25

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

int PwrPin = 2;

int VibLED = 22;

int RedLED = 23;

int MQ2Pin = 33;

int SW_420Pin = 26;

const char* ssid = "Put Your SSID HERE";

const char* password = "Put your SSID’s Password Here";

WiFiClient client;

unsigned long myChannelNumber = 1;

const char* myWriteAPIKey = "Put Your ThingSpeak Server’s Write API Key Here";
```

```
unsigned long lastTime = 0;
unsigned long timerDelay = 3000;
int Temperature;
int Humidity;
void BlinkLED (void)
{
   for (int i = 0; i < 5; i++)
   {
      digitalWrite (PwrPin, HIGH);
      delay (50);
      digitalWrite (PwrPin, LOW);
      delay (50);
   }
}
long SensedVibration ()
{
   long Measurement = pulseIn (SW_420Pin, HIGH);
   return Measurement;
}
long SmokeValuePPM ()
{
   long Smoke = analogRead (MQ2Pin);
   Smoke *= 0.20125;
   return (Smoke);
}
```

```
void setup()
{
   pinMode (PwrPin, OUTPUT);
   digitalWrite (PwrPin, LOW);
   pinMode (RedLED, OUTPUT);
   digitalWrite (RedLED, HIGH);
   pinMode (MQ2Pin, INPUT);
   pinMode (SW_420Pin, INPUT);
   Serial.begin (115200);
   Serial.print ("Send Sensors' Data! \n");
   delay (1500);
   if (WiFi.status () != WL_CONNECTED)
   {
      Serial.print ("Attempting to Connect");
      while (WiFi.status () != WL_CONNECTED)
      {
         WiFi.begin (ssid, password);
         delay (5000);
      }
      Serial.println ("Connected and the ");
      Serial.print ("IP Address is:  " );
      Serial.print (WiFi.localIP ());
      Serial.print ("\n"); Serial.print ("\n"); Serial.print ("\n");
      delay (3000);
   }
```

```
    dht.begin ();
    WiFi.mode (WIFI_STA);
    ThingSpeak.begin (client);
}
void loop()
{
    BlinkLED ();
    if ((millis () - lastTime) > timerDelay)
    {
        if (WiFi.status () != WL_CONNECTED)
        {
            Serial.print ("Attempting to Connect");
            while (WiFi.status () != WL_CONNECTED)
            {
                WiFi.begin (ssid, password);
                delay (5000);
            }
            Serial.println ("\nConnected");
        }
        int Temperature = dht.readTemperature ();
        int Humidity = dht.readHumidity ();
        if (isnan(Temperature) || isnan(Humidity))
        {
            Serial.println(("Failed to read from DHT sensor!"));
            return;}
```

```
  long SmokeValue = SmokeValuePPM ();
  long Vibration = SensedVibration ();
  Serial.print ("Temperature: ");
  Serial.print (Temperature);
  Serial.print (" °C");
  Serial.print ("  Humidity: ");
  Serial.print (Humidity);
  Serial.print (" % \n");
  Serial.print ("Smoke in Part-Per-Million: ");
  Serial.print (SmokeValue);
  Serial.print (" PPM \n");
  Serial.print ("Vibraion: ");
  Serial.print (Vibration);
  ThingSpeak.setField (1, Temperature);
  ThingSpeak.setField (2, Humidity);
  ThingSpeak.setField (3, SmokeValue);
  ThingSpeak.setField (4, Vibration);

  int RNo = ThingSpeak.writeFields (myChannelNumber, myWriteAPIKey);
  if (RNo == 200)
  {
     Serial.println ("Channel Update Sucessful"); }
  else {Serial.println("Problem updating channel. HTTP error code " + String(RNo));
  }lastTime = millis ();
}}
```