

HAWK EYE SECURITY – SECURING IOT DEVICES FROM CYBER ATTACKS



By

NC Muhammad Talha

NC Muneeb Malik

NC Ameer Hamza

NC Komal Batool

NC Mizna Akram

Supervisor: Asst. Prof. Dr. Mir Yasir Umair

Co- Supervisor: Lec Narmeen Shafqat

Submitted to the Faculty of Electrical Engineering, Military College of Signals,
National University of Sciences and Technology, Rawalpindi in partial fulfillment for
the requirements of a B.E. Degree in Telecom Engineering

JULY 2018

CERTIFICATE OF CORRECTNESS AND APPROVAL

Certified that work contained in this thesis titled “**HAWK EYE SECURITY**- Securing IoT Devices from cyber-attacks”, carried out by NC Muhammad Talha, NC Mizna Akram, NC Muneeb Malik, NC Ameer Hamza and NC Komal Batool under the supervision of Asst. Prof. Mir Yasir Umair and Lec. Narmeen Shafqat for partial fulfillment of Degree of Bachelors of Telecommunication Engineering, is correct and approved. The plagiarism of the document is _____ %

Approved By

Asst. Prof. Mir Yasir Umair

EE Dept, MCS

Lec. Narmeen Shafqat

IS Dept, MCS

Dated: July 2018

ABSTRACT

IoT devices have low computational power due to which security protocols demanding high speed computation cannot be implemented on preprocessing layer. This can compromise the confidentiality, integrity and availability of information collected through IoTs. The project “Hawk Eye Security” aims at securing IoT devices against common security attacks like DDOS (Distributed Denial of service attacks), Man in the Middle attack (MITM) and Spoofing (IP and MAC spoofing) by providing security at Perception, Network and Application Layer and disallowing intrusion of any unauthorized user. The proposed module can act as an Access point (AP) for the IoT devices which is then connected with the internet. A user-friendly Graphical user interface (GUI) and an Android application for remotely controlling IoT devices have also been developed. HAWK EYE Security has a wide variety of applications. It can secure IoT devices ranging from home network to health care and surveillance of military and civilian infrastructure. In countries like Islamic Republic of Pakistan where security is an untouched topic, this module can be useful in protecting national projects like Smart Cities.

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification either in this institution or anywhere else.

In the Name of ALLAH, the Most Benevolent, the Most Merciful.

DEDICATION

To our beloved parents and teachers
who educated us and enabled us to reach at this level.

ACKNOWLEDGEMENTS

All praises for Allah Almighty on the successful completion of this project, Who gave us strength, courage and guidance to accomplish this task.

We want to thank our parents, whose prayers helped us to successfully complete this project. They provided us everything they could manage for us.

We are also extremely thankful to our project Supervisor: Asst. Prof. Mir Yasir Umair and Co-Supervisor: Lec. Narmeen Shafqat who always gave us their kind attention. They encouraged us and gave their precious knowledge regarding project. We are extremely grateful to them for providing such a consistent support, despite their busy schedule. They took keen interest in our project work and guided us till the completion of our project work by providing all the mandatory data for developing a decent system.

Table of Contents

CHAPTER 1: INTRODUCTION	14
1.1. PROJECT OVERVIEW	14
1.2. PROBLEM STATEMENT	15
1.3. OBJECTIVES.....	16
1.3.1. <i>Academic Objectives</i>	16
1.3.2. <i>Industrial Objectives</i>	16
1.4. APPROACH.....	17
1.5. FUTURE POSSIBILITIES	18
CHAPTER 2: LITERATURE REVIEW	20
2.1. PROJECT DOMAIN	20
2.2 IOT ARCHITECTURE	20
2.3 SECURITY ISSUES REGARDING IOTs.....	21
2.4 IOTs SECURITY FEATURES	22
2.5 IOTs SECURITY SOLUTIONS	24
CHAPTER 3: TECHNOLOGICAL REQUIREMENTS	29
3.1. CORE FEATURES OF HAWKEYE	29
3.2. HARDWARE REQUIREMENTS.....	29
3.2.1. <i>Raspberry Pi 3 Vol B:</i>	30
3.2.2. <i>Micro SD Cards (Class 10)</i>	31
3.2.3. <i>Rechargeable Battery</i>	31
3.2.4. <i>IP Camera</i>	32
3.2.5. <i>Router</i>	32
3.2.6. <i>ESP8266 Wi-Fi Module</i>	33
3.3. SOFTWARE REQUIREMENTS.....	34
3.3.1. <i>Linux</i>	34
3.3.2. <i>UDHCPD</i>	34
3.3.3. <i>Hostapd</i>	35
3.3.4. <i>Iptables</i>	35
3.3.5. <i>Pen Testing Tools</i>	35
3.3.6. <i>Arduino Software</i>	36
3.3.7. <i>Android Studio for implementation of smart home on Android phone:</i>	36
CHAPTER 4: DESIGN OF MAIN MODULE	38
4.1 ATTACKS PREVENTION	38
4.1.1. <i>Reverse Path Filter:</i>	39
4.1.2. <i>ICMP Redirection Prevention:</i>	39
4.1.3. <i>TCP/IP Syn Cookies:</i>	40
4.1.4. <i>IPv4 Masquerading:</i>	40
4.2. PREVENTION OF ILLEGITIMATE ACCESS:	41
4.2.1. <i>Random Password Changing:</i>	41
4.2.2. <i>MAC-IP Binding:</i>	42
4.2.3. <i>Static ARP:</i>	43
4.3. HAWKEYE'S GUI.....	44
CHAPTER 5: DESIGN OF EXTENDED AUTOMATION MODULE	50
5.1. ANDROID APPLICATION DEVELOPMENT FOR IOTs MANAGEMENT:.....	50

5.2. NODE MCU.....	53
CHAPTER 6: FUTURE WORK AND CONCLUSION	56
6.1. FUTURE WORK	58
6.1.1. <i>Smart cities</i>	58
6.1.2. <i>SESSION LAYER</i>	58
6.1.3. <i>Use of multiple firewall</i>	58
6.2. CONCLUSION	59
APPENDIX-A	62
APPENDIX-B	64
APPENDIX-C	67
APPENDIX-D	68
APPENDIX-E	69
APPENDIX-F.....	70
APPENDIX-G	72
APPENDIX-H	74
BIBLIOGRAPHY	76
REFERENCES	77

LIST OF FIGURES

FIG. 1-1 SMART HOME	13
FIG. 1-2 IOT SECURITY MECHANISM	15
FIG. 2-1 IOT LAYERS	19
FIG. 2-2 IOT SECURITY	20
FIG. 2-3 IPV4 MASQUERADING	22
FIG. 2-4 MITM ATTACK	23
FIG. 2-5 SYN FLOOD ATTACK	23
FIG. 3-1 RASPBERRY PI	26
FIG. 3-2 IP CAMERA	28
FIG. 3-3 ROUTER PERFORMANCE	28
FIG. 3-4 WI-FI MODULE	29
FIG. 4-1 RANDOM PASSWORD CHANGING OUTPUT	37
FIG. 4-2 MAC-IP BINDING OUTPUT	38
FIG. 4-3 STATIC ARP OUTPUT	39
FIG. 4-4 GRAPHICAL USER INTERFACE	41
FIG. 4-5 GUI LOGIN WINDOW	42
FIG. 5-1 WI-FI MODULE, RASPBERRY PI, FAN, BULB, CHARGER, RELAYS	45
FIG. 5-2 CIRCUIT DIAGRAM	45
FIG. 5-3 ANDROID APPLICATION DEVELOPMENT	46
FIG. 5-4 WI-FI MODULE	47

LIST OF ABBREVIATIONS

DDOS	Distributed Denial of Service
MITM	Man in the Middle
IoT	Internet of things
NIC	Network interface controller
GUI	Graphical user Interface
NAT	Network address translation
LAN	Local area network
IP	Internet protocol
OSI	Open system interconnection
ICMP	Internet control message protocol

Chapter 1: Introduction

1.1 Project Overview

1.2 Problem Statement

1.3 Objectives

1.4 Approach

1.5 Future possibilities

Chapter 1: Introduction

A comprehensive overview of the project “HAWK EYE SECURITY” is presented in this chapter. The problem encountered, objectives, detailed specifications, targeted audience, final solution and deliverables are elaborated.

1.1. Project Overview

Internet of Things (IoTs) refer to the interconnection of computing devices via the Internet. IoTs are expected to reach 200 Billion in number by 2020¹ with an expected budget of 6.2 trillion USD by 2025² which mostly includes devices from healthcare and manufacturing sector. With such an intense increase in the number of IoT devices, the security issues regarding IoTs are imposing a major threat to privacy of the data.

The project, HAWK EYE SECURITY, is aimed at providing a solution to these security issues at perception Layer (Data and Acquisition Layer), network Layer and application Layer in general. IoT devices are the major components of smart homes and smart cities, where not just mobile phones and computers but all the other devices such as door bells, surveillance cameras, lights, clocks and other appliances not just communicate with each other but also send us information and take commands from us. So, HAWK EYE SECURITY can be used to make smart homes more secured as it is based on securing an IoT network.

Figure 1.1 shows a home network consisting of different IoT devices that are connected together. Such networks can be protected from various attacks by our security solution.



Fig. 1-1 Smart Home

1.2. Problem Statement

IoT devices are low power consumption devices due to which security protocols demanding high speed computation cannot be implemented on them. This can compromise the confidentiality, integrity and availability of data collected through IoTs. Moreover, attacks like “MIRAI” attack (back in 2016)³ could convert the IoT devices into botnets, consuming processing power for completion of tasks assigned by the hacker which may be in the form of Distributed Denial of Service (DDOS) Attack or Denial of Sleep Attack. Looking deep in the scenario, these types of attacks would not only be the cause of wastage of power resources and wear and tear of devices, reducing their lifetime but also due to increase in their use in health sector and many other departments in the near future, the vulnerabilities of these devices might result in major crimes, starting a new era of cyber terrorism.

1.3. Objectives

The project objectives can be broadly divided into academic and industrial objectives.

1.3.1. Academic Objectives

The major academic objective is to gain knowledge about IOTs, their architecture, security issues regarding IOTs and networking basics. Other academic objectives include learning of:

- Penetration testing tools
- Basics of Linux and its features
- Programming language - Python
- Linux packages (UDHCPCD and Hostapd) and Iptables
- Arduino Software and Android Studio

1.3.2. Industrial Objectives

This project can help secure IoT devices ranging from home network to healthcare and surveillance of military and civilian infrastructure. Its industrial objective is to secure IoT devices against common attacks like DDOS, Man in the Middle (MITM) and Spoofing by providing security at perception, network and application Layer. The objectives of the project are as follows:

- Devise a secure authentication scheme for IOT devices
- Protect IOT devices from spoofing attacks
- Prevent Denial of Service (DOS) attack on IOT devices

1.4. Approach

There are three scenarios of deploying the security module within the IoT infrastructure.

1. Module is deployed before router (or any device connecting to the internet)

2. Module is deployed on the router
3. Module is deployed after the router

HAWK EYE SECURITY prefers 1st approach over others since this scenario could provide protection from attacks within the home network also. The module acts as an Access Point (AP) for the IoT devices which is then connected to the Internet. The proposed module consists of a Raspberry Pi 3, operating on Linux environment for implementing security controls. Implementation of secure authentication techniques disallows intrusion of any unauthorized user to the IoT Network. Access Point is programmed to provide smooth connection of IoT devices to Internet. This module insulates authorized IoT network from our home network providing protection against any attack from even within the home network. By limiting the number of connection requests per second to the network, the solution provides protection against the DDOS Attacks. Fig. 1-2 shows different security techniques used in this project.

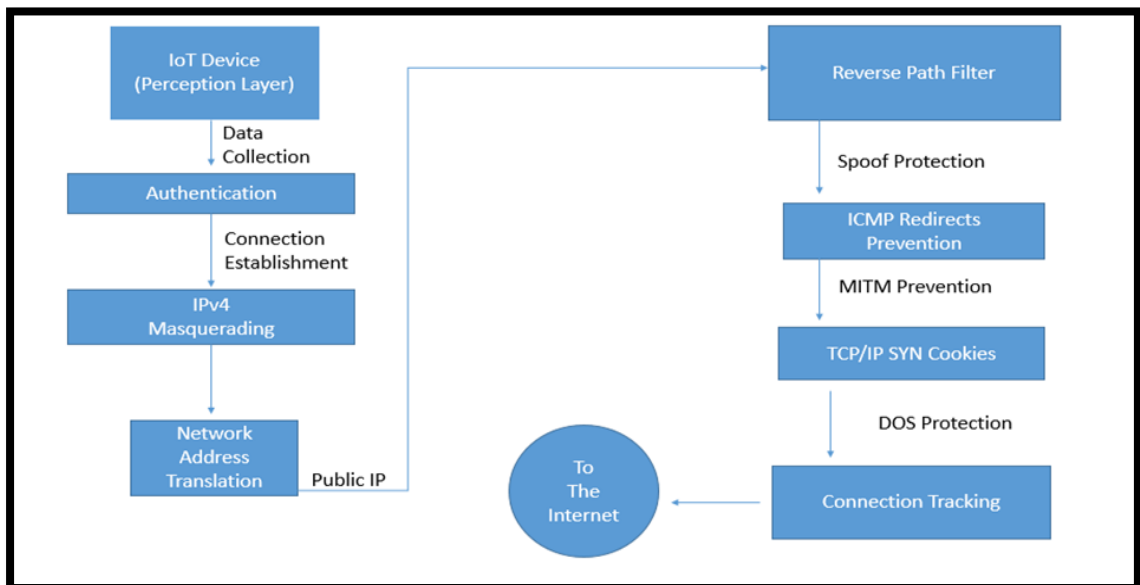


Fig. 1-2 IOT Security Mechanism

1.5. Future Possibilities

Due to rapid increase in the number of IoT devices and their applications in different fields, HAWK EYE SECURITY can be very useful for securing smart cities and smart homes from external as well as internal attacks. It can also be extended to large networks containing numerous IoT devices, by increasing its processing power and connection handling capacity. It can simply be done by using more than one microprocessor.

Chapter 2: Literature Review

2.1 Project Domain

2.2 IOT Architecture

2.3 Security Issues Regarding IOTs

2.4 IOTs Security Features

2.5 IOTs Security Solutions

Chapter 2: Literature Review

In this section, a brief overview of existing literature related to the project is given. A basic knowledge regarding the architecture of IoT devices as well as fundamental understanding of different security threats to IoT was a prerequisite before starting this project, hence various research papers and technological articles were studied. A brief description of all the topics that were studied for the project is given below.

2.1. Project Domain

The scope of the project revolves around securing IoT devices connected within a home network from potential internal and external network threats. The module is portable and adaptive to any type of IoT Environment. In countries like Pakistan, where security is an untouched topic, this module can be helpful in protecting national projects like Smart Cities, to be initiated under the umbrella of mega projects like China Pakistan Economic Corridor (CPEC).

2.2 IOT Architecture

Different IoT layers have different devices and perform different functions in an IoT architecture. A number of opinions regarding IoT layers exist. However, according to most of the researchers, the IoT primarily consists of three layers⁴:

- **Perception layer:** Sensors collect or gather data from the surroundings or environment in perception layer also known as “Sensor” layer. It detects, collects, and processes information gathered by sensors. This processed data is then transferred to the Network Layer.

- **Network layer:** Transmission of data to different IoT devices as well as IoT hubs is the main task of network layer. Network technologies like Bluetooth, 3G, LTE, Wi-Fi are used in the devices including switching devices, internet gateways and routing devices.
- **Application layer:** Application layer addresses the three main components of cyber security i.e. Confidentiality, Integrity and Authenticity of digital information. At this layer, the purpose of IoT which is the creation of smart environments is achieved. Smart environment is created at this layer of IoT. Figure 2-1 shows a general block diagram of our system, specifying IoT layers.

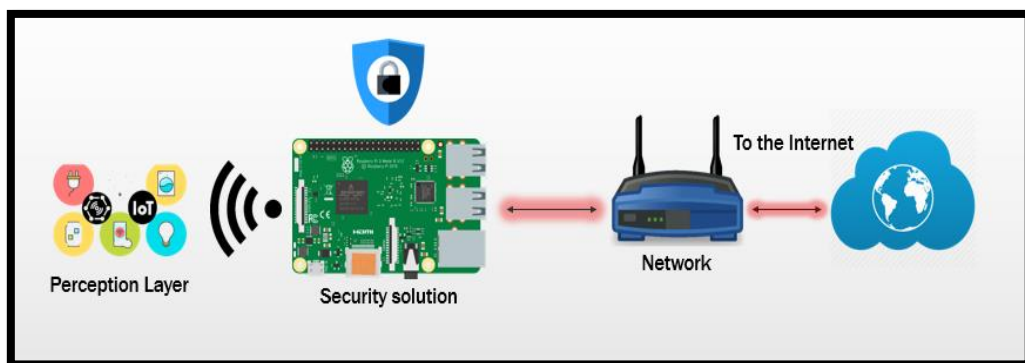


Fig. 2-1 IOT Layers

2.3 Security Issues Regarding IOTs

IoT needs similar security goals as those required for other kinds of communications. These goals are Confidentiality, Integrity and Availability (CIA) of digital information and IT assets. IOTs, however, have very confined computational and power resources. The omnipresent nature of IoT demands some supplementary security measures. Apart from CIA Triad, IoT network can be made secure by adding the following features:

- IoT devices within the network should be added after due authentication in order to prevent unauthorized and malicious devices from connecting to the secure network.

- As the IoT devices have low memory and less computational resources, employment of firewall is essential in IoT networks to screen packets directed to and from the IoT devices.

A diagram depicting the security of IoTs in a smart home is shown in Fig 2-2.



Fig. 2-2 IOT Security

2.4 IOTs Security Features

Given below are the overall security features that have been enforced in the proposed solution to achieve a safe communication network for IoTs:

- **Confidentiality:**

The key factors in IoT communication are: security of data and its accessibility to authorized parties. External objects, which are not part of the network, internal objects, which are part of the network, services and machines can be the users of an IoT network. One might not want to show the collected data to anyone else in the neighborhood. So, confidentiality of data is required.

- **Integrity:**

Interchange of data between different devices forms basis of IoT. Here arises the question of data accuracy i.e. whether legitimate user is sending the data or the data is altered intentionally or unintentionally during transmission. In IoT communication, feature of integrity can be made obligatory by upholding end-to-end security.

- **Authentication:**

Every device should be able to authenticate as well as identify itself to the other IoT devices. The ubiquitous nature of IoT can make this method a very challenging one. A procedure to communally authenticate the entities is needed in the IoT whenever they interact.

- **Lightweight Solutions:**

Due to limitations in power and computational resources of IoT devices, an inimitable security feature is introduced. In case of authentication or encryption of the devices as well as the data of the IoT, this restriction (limited memory resources) must be taken into consideration while scheming and employing protocols. In other words, all the authentication algorithms should be compatible with the device capabilities.

2.5 IOTs Security Solutions

To ensure all the above-mentioned security principles in an IoT network, thorough literature review has been carried out to propose a security solution that is capable of

providing a secure communication network for IoTs. A brief description of some of the security solutions are given below:

- **IPv4 Masquerading** can be enabled, so that packets coming inside or going outside the network are assigned a different IP address thereby hiding the original one. IP masquerading allows devices that don't have an officially assigned IP address to communicate to other networks and especially the internet. It uses a form of Network Address Translation to accomplish IPv4 Masquerading as shown in Fig. 2-3

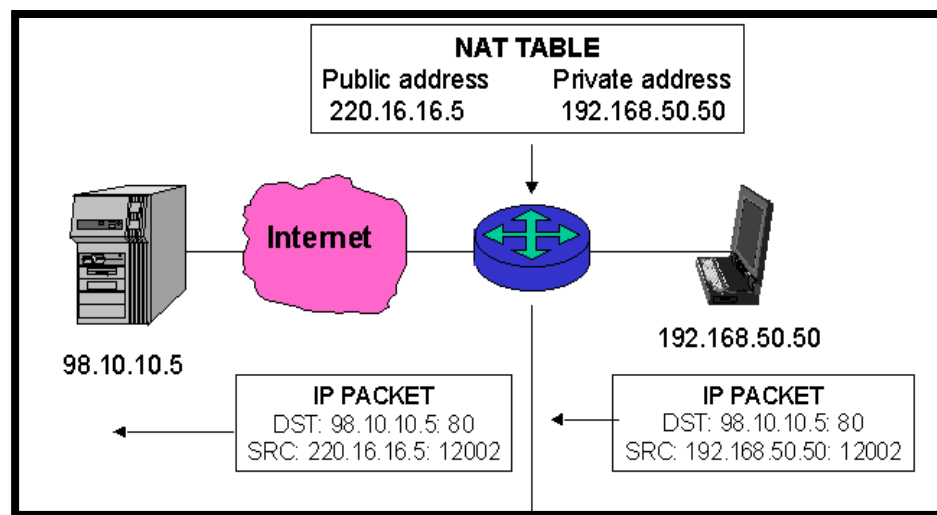


Fig. 2-3 IPv4 Masquerading

- **Spoof protection** can be provided using reverse path filter. A reverse path filter checks whether the source address of a packet is routable or not. In case a packet is not routable, i.e. not originated from legitimate source, it is dropped.
- **Man-in-the-Middle (MITM)** attacks can be prevented by blocking ICMP redirect messages. If a packet is not being routed optimally according to the router, it will send a redirect error message to the source or the sender. In this error, router tells the source to send that particular packet through a different gateway, as summarized in the Fig. 2-4.

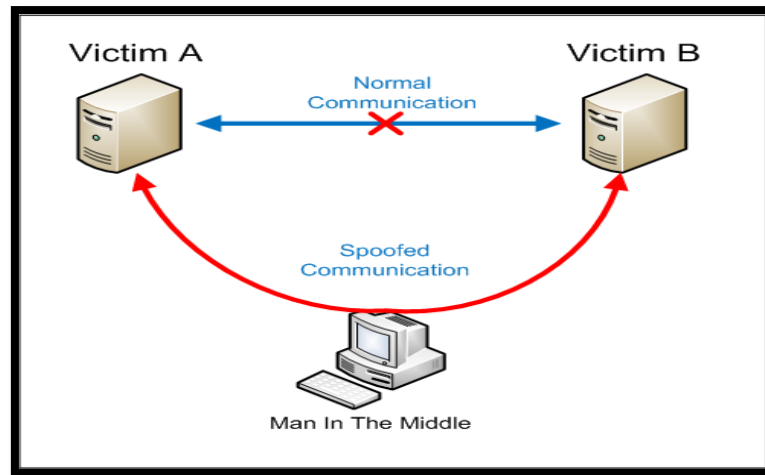


Fig. 2-4 MITM Attack

- **SYN flood attack**, which is a form of DOS attack, can be avoided by enabling TCP/IP SYN cookies. In DOS attack, the main purpose of attacker is to consume the resources of server by sending a series of SYN requests.

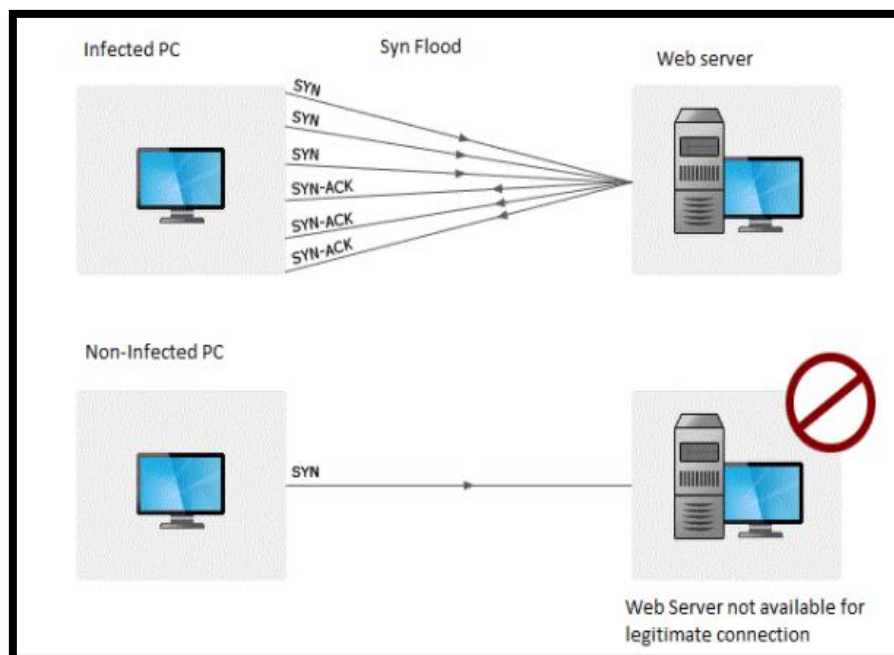


Fig. 2-5 SYN Flood Attack

This concludes the literature review carried out for the project.

Chapter 3: Technological Requirements

3.1 Introduction to Technological Requirements

3.2 Hardware Requirements

3.3 Software Requirements

Chapter 3: Technological Requirements

3.1. Core Features of Hawk Eye

The project provides a solution named as “HAWK EYE SECURITY” which protects IoT devices from potential threats like Distributed Denial of Service, Denial of Sleep, Spoofing, Unauthorized Access, Privacy Breaches and all other related active and passive attacks. Using the solution, attacks which include unauthorized and unethical use of computing resources for achieving criminal goals, can be prevented. With evolved and upgraded cyber threats, these attacks have become too effective that attack on one device can lead to distributed attacks using multiple devices within the vicinity of the attacked device without revealing any information about the actual intruder. The Hawk Eye module when deployed before the router counterfeits such attacks by blocking suspicious connections. Any unused port that can be exploited is blocked. Hence, only intended clients can access the IoT devices. Moreover, the module will keep track of all types of connections and activities.

3.2. Hardware Requirements

The hardware required for implementation of the project includes:

3.2.1. Raspberry Pi 3 Vol B:

It is a Single-board computer. Raspberry Pi 3 Vol B has Bluetooth connectivity and wireless LAN which is acting as a platform with Linux environment. The Raspberry Pi 3 Model B is a 3rd generation model. Board format identical to Raspberry Pi B+ and Raspberry Pi 2 is maintained by this 3rd generation model, but this model vaunts a faster 64 bit 1.2 GHZ SoC, with the addition of on-board Wi-Fi and Bluetooth. Raspberry Pi 3 Model B is also backward compatible. It maintains the same design and layout as

Raspberry Pi Model 2 and Raspberry Pi Model B+ so it is able to fit all the items that were included in the previous model. Power LEDs and ACT are moved to the opposite side of the SD card slot due to the addition of the Bluetooth and Wi-Fi antenna. For its specifications, refer to the appendix section A. Shown below is a Raspberry pi 3 that we have employed.



Fig. 3-1 Raspberry Pi

3.2.2. Micro SD Cards (Class 10)

SDHC is the only device in which an SD card normally works (possibly with lower performance). SD devices (such as a camera) are incompatible with SDHC card. SDHC cards are available with different memory which includes 4GB, 8GB, 16GB and 32GB. We are using Micro SD card Class 10 in our project because Class 10 memory cards are the fastest memory cards which are available as per the Speed Class rating system. Minimum speed of 10 Mega Bits per second can be retained in the Class 10 SD Card. It can also be used for storing the logs and information in our project.

3.2.3. Rechargeable Battery

Rechargeable battery is required to power the Raspberry Pi (the module on which HAWK EYE SECURITY runs). The main difference between a rechargeable and a non-rechargeable one is that the rechargeable battery can be discharged many times into a load but a non-rechargeable one is discarded after its first and last use. Electrochemical cells are the main composition of rechargeable battery. A reversible electrochemical reaction is used to hoard and store energy. Lead–acid, nickel–cadmium (NiCd), nickel–metal hydride (NiMH), lithium-ion (Li-ion), lithium-ion polymer (Li-ion polymer) and many other variety of combinations of electrolytes and electrode materials are used.

3.2.4. IP Camera

IP camera has been used as an IoT device for testing of HAWK EYE SECURITY. An IP camera is an Internet Protocol camera that works on the principle of digitalization of video. Its main advantage is that it can send or receive data through the internet which cannot be done in the analog closed-circuit television camera (CCTV). Webcams are also capable of communicating data but IP cameras can be directly accessed over any network. IP camera used in HAWK EYE SECURITY is shown below:



Fig. 3-2 IP Camera

3.2.5. Router

A router is basically used to connect networks. When compared to the OSI model, router tends to be the layer 3 device i.e. network layer of the OSI model. It has also been used for testing of HAWK EYE SECURITY. Multiple networks are joined and traffic between them is routed by the router. Network cards (NICs) are connected physically to the networks through the router. If a dedicated NIC is available for each network, then any number of networks can be connected to a particular router. Router performs routing and routing is the process of forwarding IP packets from one network to another. Fig. 3-1 shows the basic performance of routers.

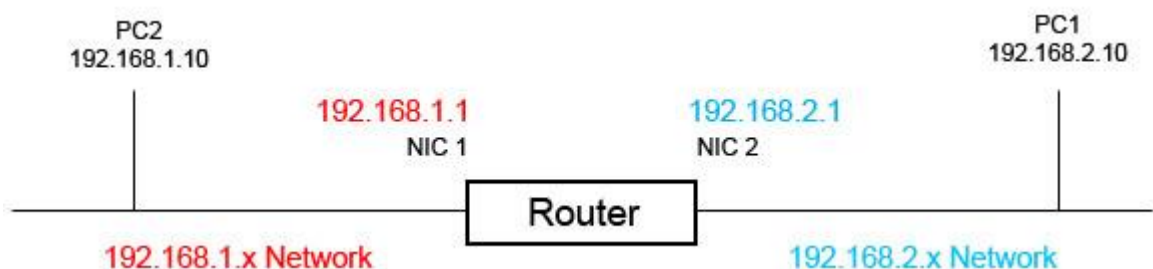


Fig. 3-3 Router Performance

3.2.6. ESP8266 Wi-Fi Module

The ESP8266 is a Wi-Fi module having an integrated SOC incorporated with TCP/IP protocol that may get any device such as a microcontroller to get into Wi-Fi community. The ESP8266 is capable of hosting an application as well as offloading all Wi-Fi networking capabilities from other software, which is used to program the processor. Every ESP8266 module is already programmed with different AT command sets called as ‘Wireless RM WARE’, which means that it can be hooked with any device and get approximately a large Wi-Fi capacity similar to the Wi-Fi shield.

This module has an integrated strong onboard processing and storage capacity that allows it to be integrated with different sensors and various application via its General-Purpose Input Outputs also called GPIO with minimum development of the front and minimal loading throughout its whole runtime. Due to its high degree of onboard chip integration, it requires minimal external circuitry which also includes the front-stop module. The module ESP8266 assists APSD for Voice over IP packages and Bluetooth interfaces. Both of these can exist simultaneously. It includes a self-adjusted RF allowing it to paint underneath all functioning conditions and needs no outside radio frequency parts. This module is shown below:



Fig. 3-4 Wi-Fi Module

3.3. Software Requirements

The software(s) required for the implementation includes:

3.3.1. Linux

Linux is an operating system used because of a need of Linux environment on the platform. Desktop or Laptop, which associates the hardware resources uses an operating system to manage it. It can be simply defined as an operating system that brings about the communication between the hardware and software. For the functionality of software, an operating system is needed.

3.3.2. UDHCPCD

It is a package to appoint IPs dynamically.

3.3.3. Hostapd

Hostapd⁵ is a package used for making an authentication server and access point. FreeBSD (net80211) and Linux (Host AP, madWi-Fi, mac80211-based drivers) are supported by the current version. Hostapd can perform the following functions:

- Creation of an Access Point.
- Creation of multiple Access Points (can be up to 8 if the card supports it).
- Creation of one Access Point on one and another Access Point on the other, (can be done through Hostapd's single instance).
- Can use 5 GHz and 2.4 GHz at same instant as well as on the same card but it requires two radios.

3.3.4. Iptables

For configuration of incoming and outgoing traffic through platform. Commands and rules are defined by generic table structure known as “iptables”. In Linux 2.4 or later, packet mangling, packet filtering and Network Address Translation (NAT) are facilitated by iptables as a part of netfilter framework. Different protocols use different programs and kernel modules. Iptables applies to IPv4. Root execution is required by the iptables as it requires root privileges to operate effectively.

3.3.5. Pen Testing Tools

For penetration testing different tools are used. In our project we are using kali Linux tool. Many steps are involved in penetration testing. The most critical among them is the phase which includes planning. Various cases of network usage, network specifications and user documentations are reviewed during the phase of planning. A series of test cases are then designed by using the given information for the purpose of penetration testing. It is a network security service, which provides one of the several methods used to prevent unauthorized network intrusion.

3.3.6. Arduino Software

Electronics projects can be made by using an open source platform known as Arduino Software. Arduino consists of two major things. First one is a microcontroller which is a programmable circuit board. Second is an Integrated Development Environment (IDE) that runs on your Personal Computer (PC) for the purpose of writing and uploading of the computer code to the physical board.

3.3.7. Android Studio for implementation of smart home on Android

phone:

Android Studio is an android application inventor. For smart homes successful implementation, the devices are needed to be controlled through another platform i.e. an android cellphone. Android Studio was used to build an application that was able to control the devices.

The next chapter is concerned with the design and development of the proposed Hack Eye Security module.

Chapter 4: Design of Main Module

4.1 Attacks Prevention

4.1.1 Reverse Path Filter

4.1.2 (ICMP) Redirection Prevention

4.1.3 TCP/IP SYN Cookies

4.1.4 IPv4 Masquerading

4.2 Prevention from illegitimate device entrance

4.2.1 Random Password Changing

4.2.2 MAC-IP Binding

4.2.3 Static ARP

4.3. GUI for the solution

Chapter 4: Design of Main Module

This chapter covers details regarding the design and development of the main module of the project that aims to secure the IOT network from aforementioned cyber-attacks. Details of the sub-modules are as follow.

4.1 Attacks prevention

The main objective of HAWK EYE SECURITY is to masquerade IP addresses of IoT devices present in the network and prevent cyber-attacks such as:

- Spoofing
- Man in the Middle (MITM) Attack (see Appendix C)
- Distributed Denial of Service (DDOS) Attack (see Appendix C)

The approach selected for the implementation of this sub-module is to use Raspberry Pi as an Access Point and a gateway through which the internet traffic to and from IoTs shall pass. Raspbian (Linux) is installed on Raspberry Pi. Two packages are also installed for the creation of Access Point and DHCP Server i.e.:

- Hostapd (for the Access Point)
- DHCPCD (for the DHCP Server)

For attacks prevention iptables are used. Iptables is a firewall which can overlook the internet traffic passing through it. Raspberry Pi generally has two important interfaces:

- Wlan0 (for IoT network)
- Eth0 (for the internet)

All the ports except SSL (port 22) are blocked for secure connection. Through iptables, ICMP packets are blocked from outside the IoT Network i.e. Eth0 so that a device outside the network cannot ping any device present inside the network. In this way, no one from outside can obtain network topology of the IoT network. Other than this, iptables are configured in a way that no intruder from outside the network can establish TCP or UDP connection with the IoTs. All connections could only be made either from the Raspberry Pi or from inside the IoT Network. Only ICMP packets and all other connections are allowed to be established from inside the IoT Network. These are the preliminary steps taken for the implementation of this sub-module.

4.1.1. Reverse Path Filter:

For Spoof Protection, Reverse Path Filter is enabled. By default, everything is routed through the router. Reverse Path Filter checks whether the packet arriving at the Raspberry Pi is routable or not. If the packet is not routable then the packet is dropped. In this way, protection from spoofing is achieved.

4.1.2. ICMP Redirection Prevention:

ICMP redirects are disabled to prevent Man in the Middle Attack. ICMP redirect error messages are sent by the router to the sender of an IP packet. When router detects that a packet coming from a source is not being routed optimally, it sends a message to the sender of that packet to transmit the packets through different gateways which can result in Man in the Middle Attack. So ICMP redirects are disabled in this way.

4.1.3. TCP/IP Syn Cookies:

Distributed Denial of Service Attack (DDOS) is a type of attack in which the availability of a given service is compromised. Attacker sends illegitimate connection establishment request, when server acknowledges that request, attacker does not send the Re-acknowledgement message and sources of the server are wasted. As a result, legitimate devices or users are deprived of the service. TCP/IP Syn Cookies are used for DDOS Attack prevention. TCP/IP Syn Cookies limit number of connection establishment requests as per the requirements. In this way DDOS attack is prevented. As our network has limited number of devices attached and all of them send re-acknowledgement message while establishing connections, so their connections will be established easily.

4.1.4. IPv4 Masquerading:

Ipv4 Masquerading is enabled to let the IoT devices interact with the outside network (internet) with their original IP addresses hidden. IPs of all the IoT devices (Private IPs) interacts with outer network through a single IP (Public IP) and different port numbers so that Raspberry Pi can differentiate incoming data as per the port number and send it to the corresponding device. Its code has been appended in Appendix B.

4.2. Prevention of illegitimate access:

In the first phase described above, attacks were prevented using different techniques. Second phase consists of preventing any illegitimate device from getting into the IoT Network. This phase consists of following parts:

- Random Password Changing

- Mac Binding
- Static ARP

4.2.1. Random Password Changing:

In Random Password Changing, a python-based code has been written to change password of the access point randomly. As a result of this code, it becomes difficult for the attacker to crack password as he will have to take some time to crack it and during that time, password will be changed. In this way network will remain secure from password cracking. Python program created for this task selects a random password from an array consisting of all capital alphabets (A-Z), small alphabets (a-z) and digits (0-9). Password length is set to 8 characters; however, it can be changed. The password can also have symbols if required. The new password will be a random one thus making it almost impossible for anyone to break. The python program changes the configuration file of Hostapd which in turn changes the password of access point. This Random Password changing feature is the first line of defense in HAWK EYE SECURITY. For random password generation code, see Appendix F. Output for random password generation is shown in Fig. 4-1.

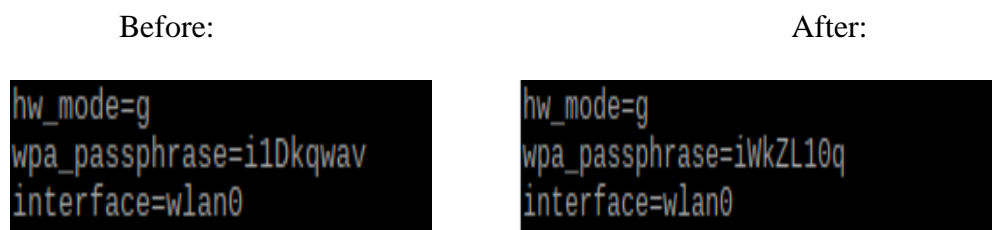


Fig. 4-1 Random Password Changing Output

4.2.2. MAC-IP Binding:

Another feature of HAWK EYE SECURITY is MAC binding. Through this feature, only the mac addresses of legitimate devices have been added into the network, thereby ensuring only legitimate devices connect to the network. If the attacker somehow breaches the “First Line of Defense”, he will not be assigned any IP from DHCP server. If the attacker uses a static IP address, even then he will not be able to access the Internet.

Logically, an illegitimate device cannot be part of this IoT network. The assignment of IP addresses through DHCP also takes place through a particular mechanism. If a new device is added in the network, that particular device is provided an IP from free IP pool. If a device is offline, then IP assigned to that device is reserved and not given to any other device. Similarly, if a device is removed from the network, then there are two scenarios. 1st scenario is that the device being removed was assigned last IP from the pool. In this Scenario, the device will simply be removed from the network and its IP will become free IP. 2nd Scenario is that the device being removed was assigned any other IP except the last one. In this scenario, the devices that have IPs after the IP of the particular device, will change their IPs upward and the last IP will move to the free IP pool. In this way, range of DHCP server will remain limited thus preventing any other device to get any dynamic IP. Output of MAC binding code is shown in Fig. 4-2, however, the code can be seen in Appendix E.

Before:

```
GNU nano 2.7.4 File: /etc/dnsmasq.conf
interface=wlan0
dhcp-range=10.3.141.50,10.3.141.52,255.255.255.0,12h
dhcp-host=14:1f:78:83:b3:de,10.3.141.51
```

After:

```
GNU nano 2.7.4 File: /etc/dnsmasq.conf
interface=wlan0
dhcp-range=10.3.141.50,10.3.141.52,255.255.255.0,12h
dhcp-host=14:1f:78:83:b3:de,10.3.141.51
dhcp-host=aa:bb:cc:dd:ee:ff,10.3.141.52
```

Fig. 4-2 MAC-IP Binding Output

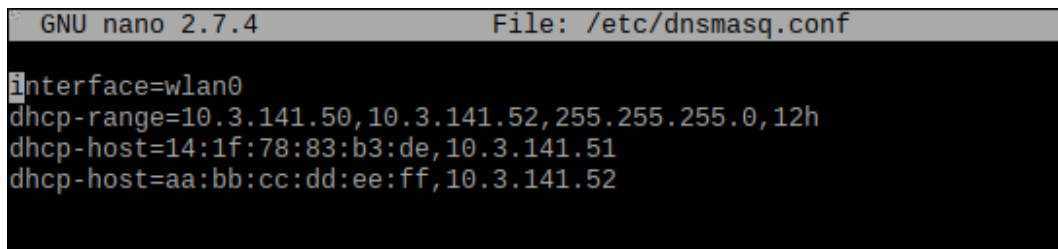
4.2.3. Static ARP:

Static ARP (address resolution protocol) technique has also been used in HAWK EYE SECURITY. By default, an ARP request is sent to the devices and in return devices reply with their corresponding MAC Addresses. Any Attacker can spoil the ARP table by associating illegitimate MAC Address with a legitimate IP address. So, a packet which is meant for a legitimate device can be sent to the illegitimate device. Static ARP table employed in this approach has helped resolve this issue. Since, ARP table will have static ARP entries, ARP request will not be sent. In this way, any packet meant for a particular legitimate device is prevented from reaching any illegitimate user. The Static ARP code has been appended in Appendix D. Output of static ARP code is shown in Fig. 4-3.

Before:

```
GNU nano 2.7.4 File: /etc/dnsmasq.conf
interface=wlan0
dhcp-range=10.3.141.50,10.3.141.52,255.255.255.0,12h
dhcp-host=14:1f:78:83:b3:de,10.3.141.51
```

After:



```
GNU nano 2.7.4 File: /etc/dnsmasq.conf
interface=wlan0
dhcp-range=10.3.141.50,10.3.141.52,255.255.255.0,12h
dhcp-host=14:1f:78:83:b3:de,10.3.141.51
dhcp-host=aa:bb:cc:dd:ee:ff,10.3.141.52
```

Fig. 4-3 Static ARP Output

4.3. Hawk Eye's GUI

HAWK EYE SECURITY provides an effective security solution for IOT devices, consisting of various features and protection mechanisms. These features play a collective role in providing a considerable defense against different types of cyber-attacks. As cyber-attacks are growing at a very rapid rate and also the attackers have become more inventive, it has now become a need to focus on cyber security.

Our module is specifically focused on the security of IOT networks as IOT devices are largely increasing in number and due to low computational power and memory resources, complex security algorithms cannot be implemented on them. HAWK EYE SECURITY has a wide range of applications as it is not limited to a single audience. It can be used in home and office networks, research and development projects, military and civilian surveillance, healthcare industry, smart cities and many more areas where security of IOTs is required. We have designed a graphical user interface of HAWK EYE SECURITY designed in order to integrate all the protection mechanisms and security features on a single platform. This graphical user interface (GUI) makes it easy for a common person, who has an average understanding in the field of networks security, to use different features of HAWK EYE SECURITY without any complexity.

GUIs generally allow users to interact with electronics devices, make them user friendly and provide better accessibility. Users do not need to know any programming languages, they can simply use icons and images instead of text, to accomplish tasks. Hawk Eye's GUI has all these amazing features.

We have designed GUI of HAWK EYE SECURITY by writing a code on python 3, as it is easy to understand and supports multiple functions and platforms. In python 3 we have used Tkinter, which is an open source GUI Programming toolkit for python. It has numerous functions and is reliable and quick. Tkinter consists of a large number of widgets including labels, frames, buttons, entries, radiobuttons, checkbuttons, scale widgets (horizontal and vertical), text, canvas, menu and listbox widgets and much more. Moreover, it consists of a database of colors containing 65536 different kind of colors and it also supports a wide variety of fonts. So, we designed a user intuitive presentable GUI of our own choice according to our requirements. Snap of this Graphical User Interface for HAWK EYE SECURITY is shown in Fig. 4-4.

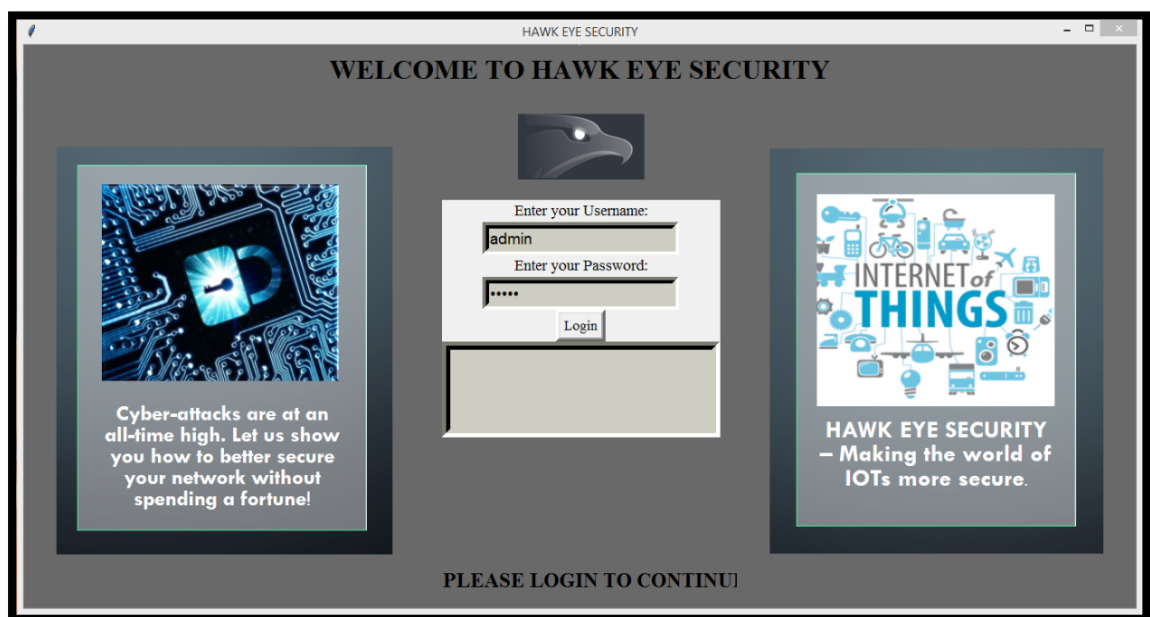


Fig. 4-4 Graphical User Interface

This GUI can be used by the administrator. After entering username and password, the administrator will be able to access HAWK EYE SECURITY and perform different operations, as shown in the figure. The default Username and password has been set to 'admin'. User will be able to use the module only after logging in by inserting correct username and password otherwise it will show an error. After logging in, a new window will be displayed as shown in Fig. 4-5 below.

This window consists of different buttons performing different functions as their name indicates. HAWK EYE SECURITY will be enabled by clicking on 'Enable' button. Administrator can now control all the IoT devices connected in the network, he can check the devices present in the ARP table (by clicking on "Show ARP table"). New devices can be added in the ARP table by clicking on the button 'Enter a new device'. Similarly, a device present in the network can be removed by clicking 'Remove a Device'. The code of the GUI of Hawk Eye Security can be seen in the Appendix G.

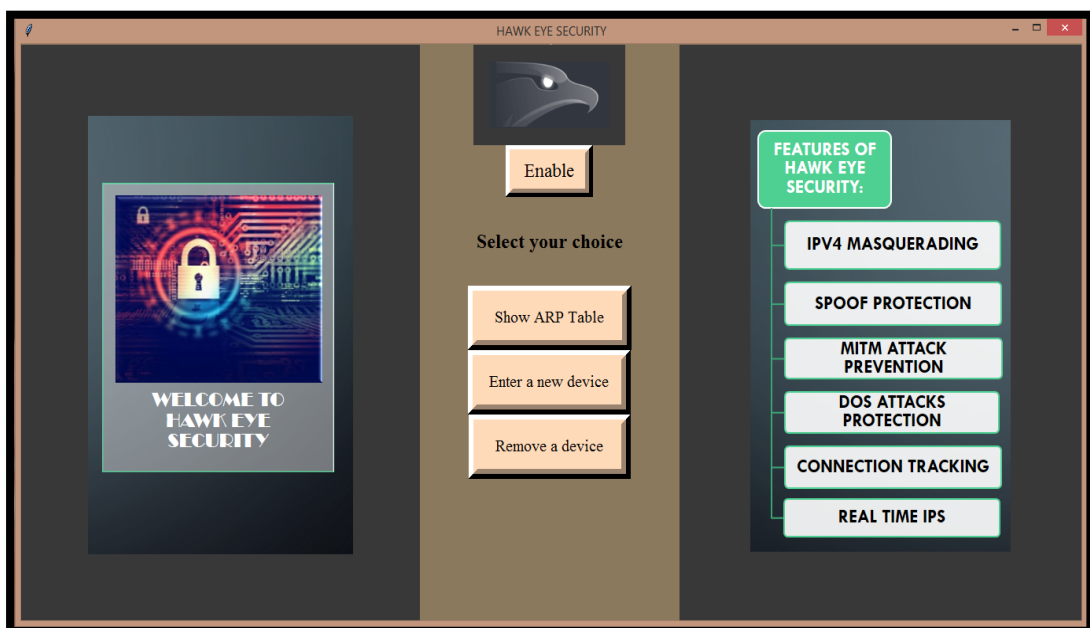


Fig. 4-5 GUI Login Window

Chapter 5: Design of Extended Automation Module

5.1 Android Application development for IoTs management

5.2 Node MCU

Chapter 5: Design of Extended Automation

Module

This chapter presents details pertaining to the second module of project “HawkEye Security”. It covers the design and development of the automation module that is equipped with an Android application and helps IoT users access and manage their device at the comfort of their mobile phone anytime anywhere.

5.1. Android Application development for IoTs management:

This Android App is developed for the IOT management and solution providing access to all legitimate users only. This app consists of username and passwords lists and from these lists, access will be granted to users if the actual user is accessing it. In the GUI, the user is required to give/ select the IP Address of the device and Pi through which it wants to communicate. Once the devices are available on the android screen, the user can switch them On and Off. Here, when the connection is established, HTTP server is initialized that communicates via Wi-Fi through the given IP Address. The whole communication process is through HTTP server which is created by the NodeMCU at default port 80.

Fig. 5-1 actually represents the whole flow diagram of the circuit of this module. In this circuit, Raspberry Pi is acting as an access point just like a router. NodeMCU which acts as an IOT device is connected to the same Raspberry Pi. Now when NodeMCU is connected successfully with the Pi, it is assigned an IP address only known to Raspberry Pi and the legitimate user. NodeMCU is further connected to the relay module which

has 4 channels through which we can be turned On and Off to enable or disable home appliances. This relay module uses four digital pins of the MCU board which is then connected to the Load devices.

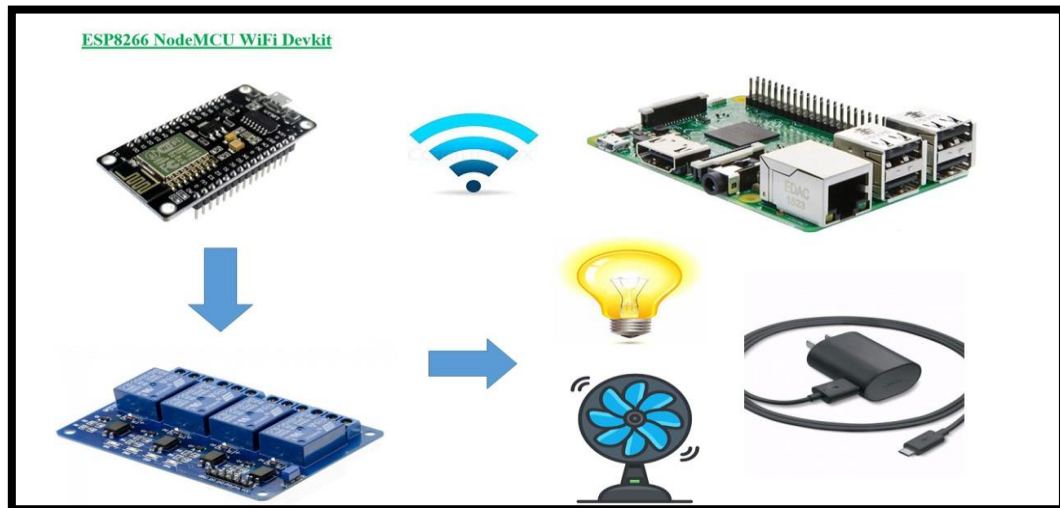


Fig. 5-1 Wi-Fi Module, Raspberry pi, Fan, Bulb, Charger, Relays

The Circuit diagram is as follows:

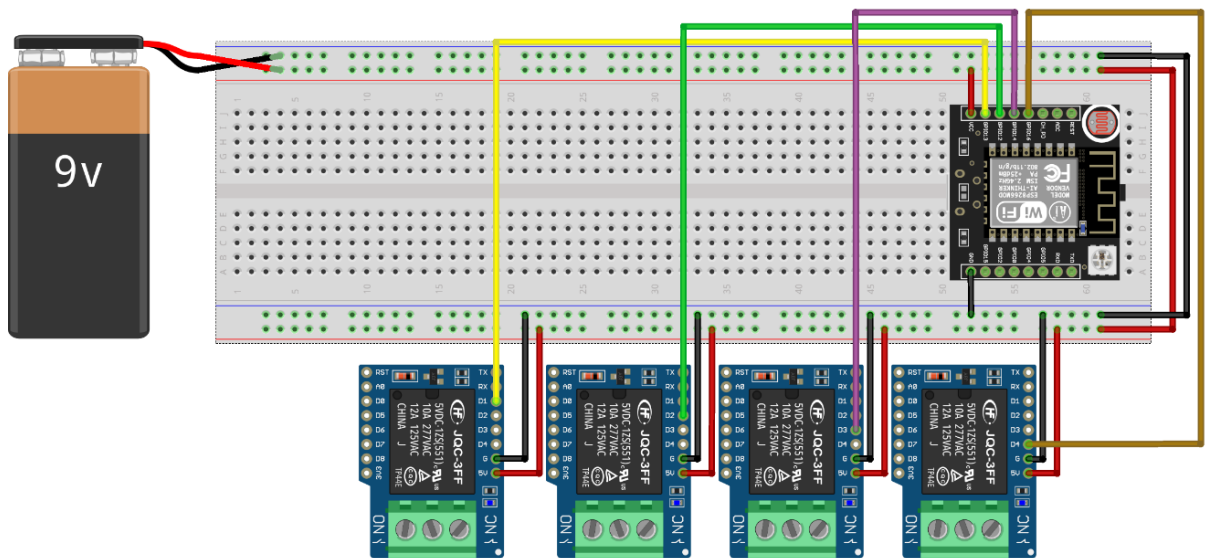


Fig. 5-2 Circuit Diagram

Once all the connections have been made, we move towards the android application. This application is actually responsible for switching on and off different home

appliances based on a request sent by the user. User can either send HTTPS requests or simply a command over the internet by using a cloud service like AWS and adafruit or Thingspeak etc. Fig. 5-2 shows the interface of our android application in which only registered or legitimate users can be added or deleted and can use the services of the application.

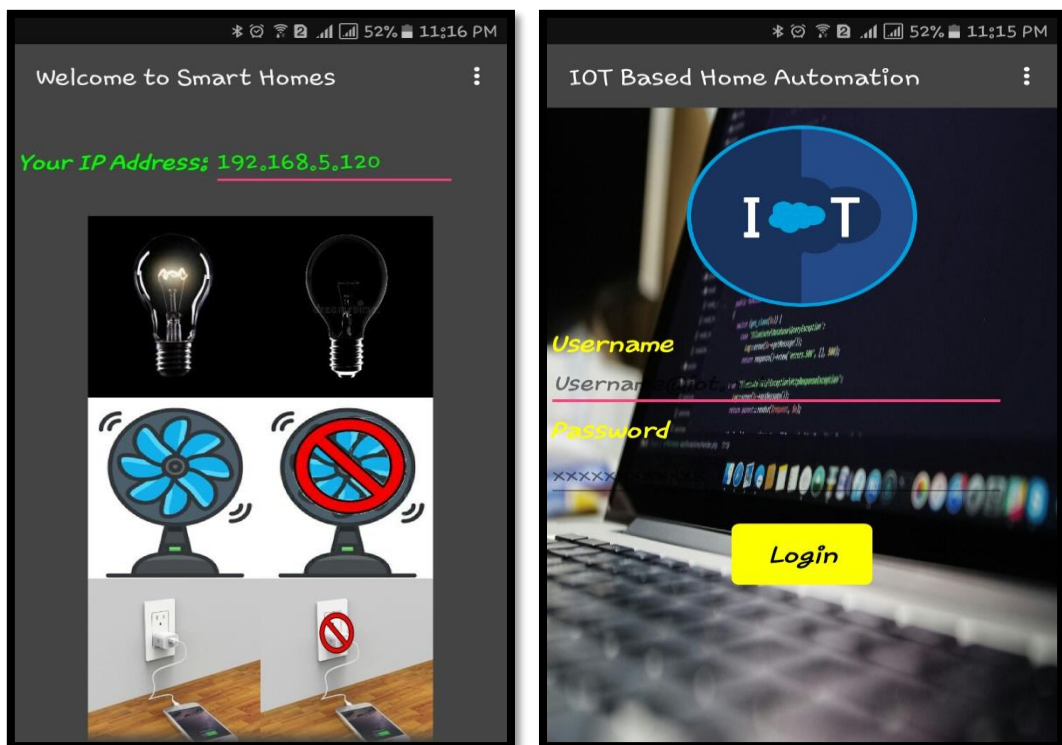


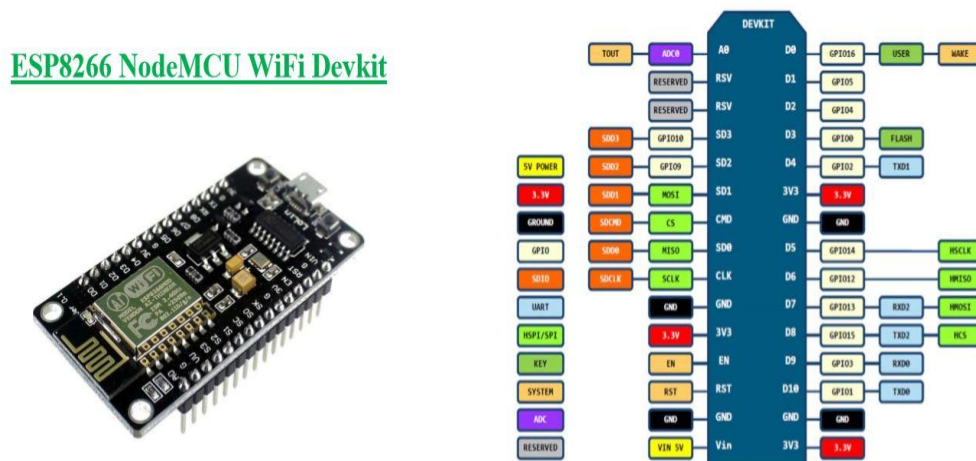
Fig. 5-3 Android Application Development

Code for Android Application development for IoTs management has been attached in Appendix H.

5.2. Node MCU

NodeMCU is a Wi-Fi module with an easily programmable platform just like an Arduino board consisting of digital and analog pins as well. A rich hodge-podge of pins out and a USB connector is present in NodeMCU. NodeMCU can be connected to the

Personal Computer and it can be flashed as well through the use of a micro USB cable. It can act as Access Point Station Mode, can connect to the Wi-Fi Routers or Internet routers and can act as a client. In our case, we are actually using it to make an IOT based SMART HOME network, in which it is connected securely to the Raspberry Pi, acting as an access point. The HTTP Server is initialized by the NODE MCU on Port 80 and where it starts listening to the incoming commands. Also, it is connected to the relay module in which the devices are plugged in, according to the name of device added in the application. Digital Pins are used for the Switching Application of relays. Different appliances are then connected to the Relays. Fig. 5-3 shows a NodeMCU WiFi Devkit.



Chapter 6: Future Work and Conclusion

6.1 *Future Work*

6.1.1 *Smart cities*

6.1.2 *Session Layer*

6.1.3 *Use of multiple firewalls*

6.2 *Conclusion*

Chapter 6: Future Work and Conclusion

In this chapter we will discuss some of the future aspects of HAWK EYE SECURITY to further protect IoT devices from potential threats like:

- Distributed Denial of Service(DDOS) attack
- Man in the middle (MITM) attack
- Spoofing
- Unauthorized Access
- Privacy Breaches
- Active and passive attacks

By using this solution, attacks that embrace unauthorized and unethical use of computing resources for achieving criminal goals, can be prevented. With evolved and upgraded cyber threats, these attacks have become too effective that an attack on one device can cause distributed attacks on multiple devices in the neighborhood of the attacked device, not revealing any knowledge about that particular intruder. The deployed module before the router can counterfeit such attacks by blocking the connections concerning the suspected activities. Any non-purposed port which can cause vulnerabilities is blocked. Only intended clients should be able to access the IoT devices. Moreover, the module can keep track of all sorts of connections and activities.

Following are the tasks which were accomplished to fulfill security concerns:

- Raspberry Pi as an Access Point
- IPv4 Masquerading

- Spoof Protection and Authentication Using 4-digit pin
- Spoof Protection (Using Reverse Path Filter)
- TCP/IP SYN Cookies
- ICMP Redirection Prevention
- MAC Binding and Static AR
- Protection against WPA2 Cracking
- GUI for the solution
- Android Application development for IoTs management

6.1. Future Work

6.1.1. Smart cities

The proposed solution can secure the IoT devices connected within a home network from potential threats from inside and outside the network. The module is portable and adaptive to any type of IoT Environment. In countries where security is an untouched topic, this module can be helpful in protecting national projects like Smart Cities. IoT presently employs variety of protocols, services and devices to realize a typical objective. However, to assimilate a network of IoT frameworks to realize a much bigger framework, for example, to make a sensible city by the mixing of the many smart homes, there must be a collection of standards that ought to be followed from small to macro levels of IoT realization. The present day demand of IoT is to possess well outlined architecture standards comprising of information models, interfaces, and protocols.

6.1.2. SESSION LAYER

The IoT structure containing 3 layers is not capable of managing and organizing sessions between different devices according to many researchers. To overcome this shortcoming, there is a need for implementing certain protocols to make communication between devices more effective. An additional session layer can be implemented in IoT structure to particularly control connections and sessions and improve communication between different devices.

6.1.3. Use of multiple firewall

We have devised a module to make home network secured from intruders and unauthorized access. We can make it more effective by employing a dashboard on pi which can store information regarding location of devices, amount of sent data and other such parameters in a home network. We have employed a single raspberry pi in our module, which acts as a gateway for all the incoming and outgoing traffic. But one weakness of using a lone pi is that it can act as a single point of failure. Moreover, it does not possess sufficient processing power to manage a large number of connections and additional load. So, we can make it more efficient by using multiple firewalls. Also, a load balancer can be used to completely remove the chance of DDOS attacks.

6.2. Conclusion

There are different security problems and challenges that we need to focus, as every layer in IoT structure is vulnerable to different kind of attacks. In our security solution, we have particularly focused on securing the IoT network from several attacks and unauthorized access. Due to rapid progress in technology, it's necessary to implement advanced protocols such as 5G and IPv6. Most of the major developments in IoTs are

observed on small scales for examples in small industries and firms. Several security threats have to be addressed for scaling the IoT network from a small company to a large group of different systems and companies. IoT has the power to completely change our way of living. Security is the most significant concern in this regard while considering an IoT framework. In the near future, use of IoTs will become more common if security issues such as authentication, confidentiality, integrity, trust management, access management and international policies are focused. New hardware and software technologies need to be implemented to address the present IoT challenges.

Appendix

1. *Appendix A - Raspberry Pi 3 Vol B Specifications*
2. *Appendix B - Code for IPv4 Masquerading*
3. *Appendix C-Code for MITM & DOS attack prevention*
4. *Appendix D-Code for Connection tracking & Static ARP*
5. *Appendix E- Code for MAC-IP Binding*
6. *Appendix F-Code for Random Password Changing*
7. *Appendix G- Code for GUI*
8. *Appendix H-Code for Android Application Development*

Raspberry Pi 3 Vol B Specifications

Specifications:

- **Processor**
 - Broadcom BCM2387 chipset.
 - 1.2GHz Quad-Core ARM Cortex-A53 (64Bit)
 - 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
- **GPU**
 - Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode.
 - Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
- **Memory**
 - 1GB LPDDR2
- **Operating System**
 - Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
- **Dimensions**
 - 85 x 56 x 17mm
- **Power**
 - Micro USB socket 5V1, 2.5A

Connectors

- **Ethernet**
 - 10/100 BaseT Ethernet socket
- **Video Output**
 - HDMI (rev 1.3 & 1.4)
 - Composite RCA (PAL and NTSC)
- **Audio Output**
 - Audio Output 3.5mm jack
 - HDMI

- USB 4 x USB 2.0 Connector
- **GPIO Connector**
 - 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
 - Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
- **Camera Connector**
 - 15-pin MIPI Camera Serial Interface (CSI-2)
- **Display Connector**
 - Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
- **Memory Card Slot**
 - Push/pull Micro SDI

APPENDIX-B

Code for IPv4 Masquerading

```
#1. Install the necessary software-----
sudo apt-get update
sudo apt-get install hostapd udhcpd -y
sudo apt-get install iptables -y
sudo apt-get install zip unzip -y
sudo apt-get update
#2. Configure DHCP-----
x=tem.temtouch $x
sudo rm -rf /etc/default/udhcpd
sudo mkdir /etc/default
sudo touch /etc/default/udhcpd
echo "start 192.168.42.2 " >> $x
echo "end 192.168.42.20" >> $x
echo "interface wlan0" >> $x
echo "remaining yes" >> $x
echo "opt dns 8.8.8.8 4.2.2.2" >> $x
echo "opt subnet 255.255.255.0" >> $x
echo "opt router 192.168.42.1" >> $x
echo "opt lease 864000" >> $x
sudo mv $x /etc/udhcpd.conftouch $x
echo "# Comment the following line to enable" >> $x
echo "#DHCPD_ENABLED=\"no\"" >> $x
echo "# Options to pass to busybox' udhcpd." >> $x
echo "# -S    Log to syslog" >> $x
echo "# -f    run in foreground" >> $x
echo "DHCPD_OPTS=\"-S\"" >> $x
sudo mv $x /etc/default/udhcpd
#give the Pi a static IP address
sudo ifconfig wlan0 192.168.42.1
#-----SETUP Station Interface for Rt5370-----
touch $x
sudo cp /etc/network/interfaces /etc/network/interfaces.bk
echo "source-directory /etc/network/interfaces.d" >> $x
echo "auto lo" >> $x
echo "iface lo inet loopback" >> $x
echo "" >> $x

echo "allow-hotplug wlan0" >> $x
echo "iface wlan0 inet dhcp" >> $x
echo "    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf" >> $x
echo "    wireless-power off" >> $x
echo "" >> $xecho "iface default inet dhcp" >> $x
sudo chown --reference=/etc/network/interfaces $x
sudo chmod --reference=/etc/network/interfaces $x
sudo mv $x /etc/network/interfaces
```



```

sudo cp /etc/network/interfaces /etc/network/interfaces.sta
sudo chmod --reference /etc/network/interfaces.bk /etc/network/interfaces.sta
#-----SETUP AP Interface for Rt5370-----
touch $x
echo "source-directory /etc/network/interfaces.d" >> $x
echo "auto lo" >> $x
echo "iface lo inet loopback" >> $x
echo "" >> $x
echo "auto eth0" >> $x
echo "iface eth0 inet dhcp" >> $x
echo "" >> $x
echo "iface wlan0 inet static" >> $x
echo "    address 192.168.42.1" >> $x
echo "    netmask 255.255.255.0" >> $x
echo "    wireless-power off" >> $x
echo "" >> $x
echo "iface default inet dhcp" >> $x
echo "up iptables-restore < /etc/iptables.ipv4.nat" >> $x
sudo chmod --reference=/etc/network/interfaces $x
sudo chown --reference=/etc/network/interfaces $x
sudo mv $x /etc/network/interfaces.ap
sudo chmod --reference /etc/network/interfaces.bk /etc/network/interfaces.ap
#Config /etc/wpa_supplicant/wpa_supplicant.conf
touch $x
sudo rm -rf /etc/wpa_supplicant/wpa_supplicant.conf
sudo mkdir /etc/wpa_supplicant
sudo touch /etc/wpa_supplicant/wpa_supplicant.conf
echo "ctrl_interface=/var/run/wpa_supplicant" >> $x
echo "update_config=1" >> $x
echo "network={" >> $x
echo "    ssid=\"Robotbase\"" >> $x
echo "    psk=\"Do@nket201234\"" >> $x
echo "}" >> $x
sudo mv $x /etc/wpa_supplicant/wpa_supplicant.conf
#3. Configure HostAPD-----
touch $x
echo "interface=wlan0" >> $x

echo "driver=nl80211" >> $x
echo "ssid=My_AP" >> $x
echo "hw_mode=g" >> $x
echo "channel=6" >> $x
echo "macaddr_acl=0" >> $x
echo "auth_algs=1" >> $x
echo "ignore_broadcast_ssid=0" >> $x
echo "wpa=2" >> $x
echo "wpa_passphrase=hawkeyesecurity" >> $x

```

```

echo "wpa_key_mgmt=WPA-PSK" >> $x
echo "wpa_pairwise=TKIP" >> $x
echo "rsn_pairwise=CCMP" >> $x
sudo mv $x /etc/hostapd/hostapd.conf
#4. Configure NAT-----
touch $x
echo "DAEMON_CONF=\"/etc/hostapd/hostapd.conf\"" >> $x
sudo mv $x /etc/default/hostapdtouch $x
sudo sh -c "echo 1 >> /proc/sys/net/ipv4/ip_forward"
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo sh -c "iptables-save >> /etc/iptables.ipv4.nat"
#-----5. Fire it up! R-----
sudo service hostapd start
sudo service udhcpd start
#-----6.get the hotspot to start on boot-----
sudo update-rc.d hostapd enable
sudo update-rc.d udhcpd enable
#-----Create wifiConnect.py-----
sudo apt-get install dnsmasq -y
sudo service dnsmasq start
sudo update-rc.d dnsmasq enable
sudo apt-get install udhcpc -y
sudo cp ap.sh /usr/bin/ap

```

APPENDIX-C

Code for MITM & DOS attack Prevention

MITM attack prevention

```
#!/simple

iptables -P INPUT -j DROP
iptables -P OUTPUT -j ACCEPT
iptables -P FORWARD -j DROP
iptables -N SERVICES
iptables -t NAT -P OUTPUT -j ACCEPT
iptables -t NAT -P PREROUTING -j ACCEPT
iptables -t NAT -P POSTROUTING -j ACCEPT

#input

iptables -A INPUT -p icmp -i eth0 -j DROP
iptables -A INPUT -p icmp -i wlan0 -j ACCEPT
iptables -A INPUT -p TCP -i eth0 -j DROP
iptables -A INPUT -p UDP -i eth0 -j DROP
iptables -A INPUT -p ALL -i eth0 -j DROP
iptables -A INPUT -p TCP -i wlan0 -j ACCEPT
iptables -A INPUT -p UDP -i wlan0 -j ACCEPT
iptables -A INPUT -p ALL -i wlan0 -j ACCEPT
iptables -A INPUT -p ALL -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p TCP -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p UDP -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p ALL -i eth0 -m state --state NEW -j DROP
iptables -A INPUT -p UDP -i eth0 -m state --state NEW -j DROP
iptables -A INPUT -p TCP -i eth0 -m state --state NEW -j DROP
iptables -A INPUT -p TCP --tcp-flag ALL,SYN,ACK -j DROP

#output

iptables -A OUTPUT -p icmp -o eth0 -j ACCEPT
iptables -A OUTPUT -p icmp -o wlan0 -j ACCEPT
iptables -A OUTPUT -p TCP -o eth0 -j ACCEPT
iptables -A OUTPUT -p UDP -o eth0 -j ACCEPT
iptables -A OUTPUT -p ALL -o eth0 -j ACCEPT
iptables -A OUTPUT -p TCP -o wlan0 -j ACCEPT
iptables -A OUTPUT -p UDP -o wlan0 -j ACCEPT
iptables -A OUTPUT -p ALL -o wlan0 -j ACCEPT
iptables -A OUTPUT -p ALL -o eth0 -m state --state ESTABLISHED,RELATED,NEW,INVALID -j ACCEPT
iptables -A OUTPUT -p TCP -o eth0 -m state --state ESTABLISHED,RELATED,NEW,INVALID -j ACCEPT
iptables -A OUTPUT -p UDP -o eth0 -m state --state ESTABLISHED,RELATED,NEW,INVALID -j ACCEPT
```

DOS prevention

```
#!/tcp/syn

iptables -A FORWARD -p TCP --syn -m limit --limit 1/s -j ACCEPT
iptables -A FORWARD -p TCP --tcp SYN,ACK,RST,FIN -m limit --limit 10/s --limit-burst 5 -j ACCEPT
iptables -A FORWARD -p TCP --tcp SYN,ACK,RST,FIN -m limit ! --limit 10/s -j DROP
```

APPENDIX-D

Code for Connection Tracking & Static ARP

Connection tracking

```
#Connection Tracking

$ modprobe ip_conntrack
$ modprobe ip_conntrack_ftp

iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p udp --dport 53 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 53 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 10/second -j ACCEPT
```

Spoof protection

```
GNU nano 2.7.4 File: Desktop/static-arp.py
#!/bin/python3
import os.path

mac=input("Enter the mac address of the device in 00:00:00:00:00:00 format:\n")
length=len(mac)
if((mac[2]and mac[5] and mac[8] and mac[11] and mac[14])!=':'):
    print('The MAC Address is not valid')
elif(length!=17):
    print('The MAC Address is not valid')
else:
    counter=1

    with open("Desktop/static_arp.sh","r") as in_file:
        buf=in_file.readlines()
        for line in buf:
            print(counter)
            counter=counter+1

    ctr=str(counter)
    x="sudo arp -s 192.168.137."+ctr + " "+mac+"\n"

    print(x)
    with open("Desktop/static_arp.sh","a") as out_file:
        out_file.write(x)
```

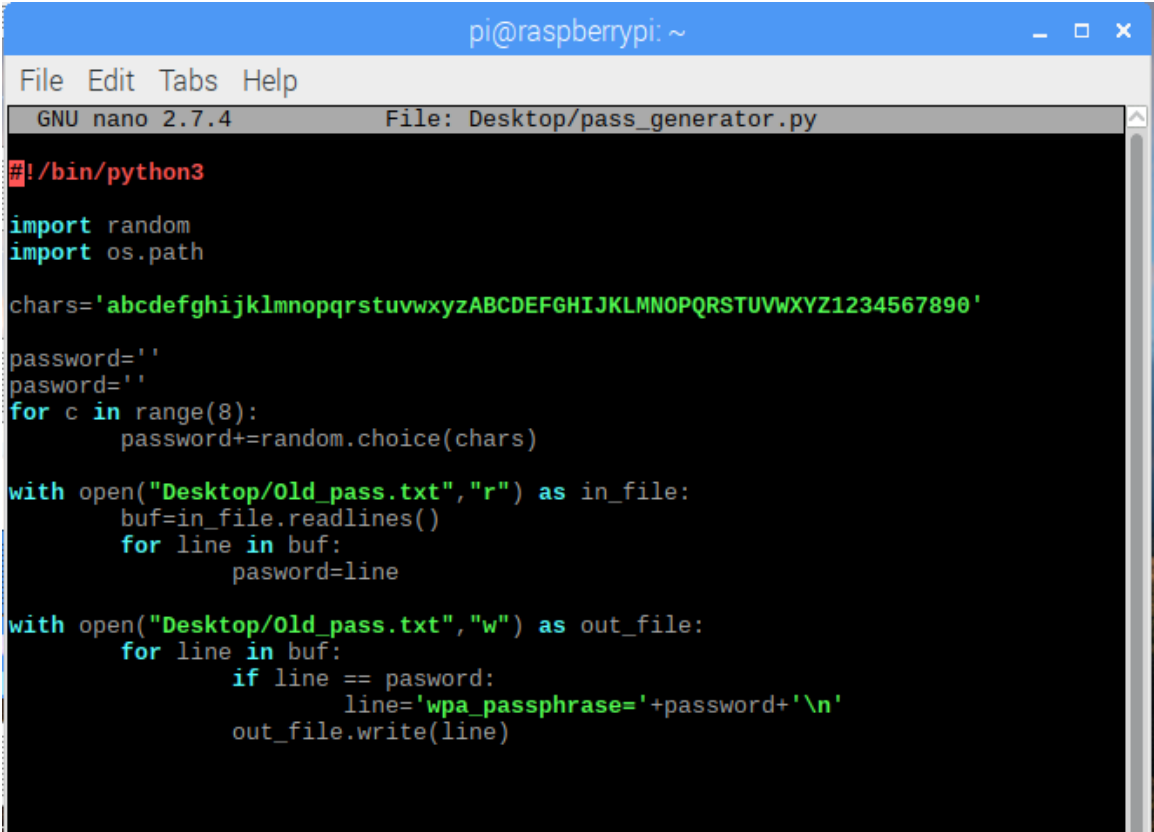
APPENDIX-E

Code for MAC-IP Binding

```
GNU nano 2.7.4 File: Desktop/mac_bind.py
#!/bin/python3
import os.path
mac=input("Enter the mac address of the device in 00:00:00:00:00:00 format:\n")
length=len(mac)
if((mac[2]and mac[5] and mac[8] and mac[11] and mac[14])!=':'):
    print('The MAC Address is not valid')
elif(length!=17):
    print('The MAC Address is not valid')
else:
    last_line=''
    last_line1=''
    with open("/etc/dnsmasq.conf","r") as in_file:
        buf=in_file.readlines()
        for line in buf:
            last_line=line
    with open("Desktop/free_ips.txt","r") as in_file1:
        buf1=in_file1.readlines()
        for line in buf1:
            last_line1=line
    ip_str=last_line1[9:11]
    ip=int(ip_str)
    with open("Desktop/free_ips.txt","w") as out_file1:
        for line in buf1:
            if line==last_line1:
                line=''
            out_file1.write(line)
    ip=ip-1
    ip1=str(ip)
    temp='10.3.141.'+ip1
    x='dhcp-host='+mac+','+last_line1
    with open("/etc/dnsmasq.conf","w") as out_file2:
        for line in buf:
            if line=='dhcp-range=10.3.141.50,'+temp+',255.255.255.0,12h'+'\n':
                line='dhcp-range=10.3.141.50,10.3.141.'+ip_str+',255.255.255.0,12h'+'\n'
            out_file2.write(line)
    with open("/etc/dnsmasq.conf","a") as out_file:
        out_file.write(x)
```

APPENDIX-F

Code for Random Password Changing

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal shows the GNU nano 2.7.4 editor editing a file named 'Desktop/pass_generator.py'. The code is a Python script that generates a random password and updates it in a file named 'Desktop/Old_pass.txt'. The script imports 'random' and 'os.path', defines a set of characters, generates an 8-character password, reads the current password from the file, and then writes the new password back to the file, replacing the old one.

```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4 File: Desktop/pass_generator.py
#!/bin/python3
import random
import os.path

chars='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'

password=''
password=''
for c in range(8):
    password+=random.choice(chars)

with open("Desktop/Old_pass.txt","r") as in_file:
    buf=in_file.readlines()
    for line in buf:
        password=line

with open("Desktop/Old_pass.txt","w") as out_file:
    for line in buf:
        if line == password:
            line='wpa_passphrase='+password+'\n'
        out_file.write(line)
```

APPENDIX-G

Code for GUI

```
file Edit Format Run Options Window Help
from tkinter import *
import random
import time

root = Tk()

root.geometry("1600x700+0+0")
root.title("HAWK EYE SECURITY")
root.configure(background='dim gray')
app=Frame(root)
app.pack()
text_Input=StringVar()
text_Input1=StringVar()
Tops = Frame(root, width=1600, height=0,bg="dim gray", relief=SUNKEN)
Tops.pack(side=TOP)
f1=Frame(root, width=360, height=800, bg="dim gray", relief= SUNKEN)
f1.pack(side=LEFT)
f2=Frame(root, width=360, height=800, bg="dim gray", relief=SUNKEN)
f2.pack(side=RIGHT)
f3=Frame(root, width=1600, height=50, relief=SUNKEN)
f3.pack(side=BOTTOM)
p1=PhotoImage(file='Eagle.png')
L1=Label(Tops,font=('Times', 25, 'bold'), text="WELCOME TO HAWK EYE SECURITY", fg="black", bd=10, bg='dim gray')
L1.grid(row=0, column=0)
l3=Label(root, image=p1, bg="dim gray",bd=25)
l3.pack()
L5=Label(f3, font=('Times', 19, 'bold'), text="PLEASE LOGIN TO CONTINUE ", fg='black',bd=20, bg='dim gray')
L5.grid(row=1, column=0)
p2=PhotoImage(file='ty.png')
l6=Label(f1, image=p2, bg='dim gray',bd=40).grid()
p3=PhotoImage(file='gu.png')
l7=Label(f2, image=p3, bg='dim gray',bd=40).grid()
bullet="\u2022"

class Application(Frame):

    def __init__(self, master):
        Frame.__init__(self, master)
        self.pack()
        self.create_widgets()
```

```

File Edit Format Run Options Window Help

def create_widgets(self):
    self.instruction = Label(self, font=('Times', 14),text = "Enter your Username:")
    self.instruction.pack()

    self.choice = Entry(self, font=('arial',14),bd=8, bg='ivory3')
    self.choice.pack()
    self.instruction1=Label(self, text="Enter your Password:", font=('times',14))
    self.instruction1.pack()

    self.choice1=Entry(self, show=bullet, font=('arial',14),bd=8, bg='ivory3')
    self.choice1.pack()

    self.submit_button = Button(self, font=('Times',13),text = "Login", bd=5, command = self.reveal,activebackground='cyan2')
    self.submit_button.pack()

    self.text = Text(self, font=('Times',13), width = 35, height = 5, wrap = WORD,bd=10, bg='ivory3')
    self.text.pack()

def reveal(self):
    content = self.choice.get()
    content1=self.choice1.get()

    if (content == "admin" and content1 == "admin"):
        def change():
            b7.configure(text='HAWK EYE Enabled')
        def new():
            n=TopLevel(root)
            n.geometry('500x200')
            n.title('Enter a new device')
            n.configure(background='gray1')
            L1=Label(n, text="Enter MAC address:", font=('Times',12, 'bold'),bd=15,bg='khaki').pack()
            E1=Entry(n, font=('Times',12), bd=12, bg='khaki').pack()
            self.destroy()
            Tops.destroy()
            f1.destroy()
            f2.destroy()
            f3.destroy()
            L1.destroy()
            L5.destroy()
            L3.destroy()
            fr1=Frame(root, width=400, height=800, bg='khaki')

```

```

fr1.pack(side=LEFT)
fr2=Frame(root, width=400, height=800, bg='khaki')
fr2.pack(side=RIGHT)
Ts=Frame(root,width=1600,height=200, bg='NavajoWhite4')
Ts.pack(side=TOP)
pi=PhotoImage(file="Eagle.png")
lab=Label(Ts, image=pi, bg='gray22',bd=20,pady=10).grid()
ph=PhotoImage(file="xb.png")
lab=Label(fr1, image=ph, bg='gray22',bd=85).grid(row=0,column=0)
b7=Button(Ts, font=('Times',17), text='Enable',bg='peach puff',bd=10,command=change,padx=7,activebackground='cyan2')
b7.grid()
C=Label(root, font=('Times',18,'bold'), text='Select your choice', bg='NavajoWhite4',bd=10,pady=30).pack()
bu1=Button(root, font=('Times',15), text='Show ARP Table',bg='peach puff', bd=12,padx=17, pady=10).pack()
root.configure(background='NavajoWhite4')
p6=PhotoImage(file='cf.png')
display1=Label(fr2, image=p6, bg='gray22',bd=90)
display1.grid(row=0,column=0)
bu2=Button(root, font=('Times',15), text='Enter a new device', bg='peach puff', bd=12,padx=10,pady=10, command=new).pack()
bu3=Button(root, font=('Times',15), text='Remove a device', bg='peach puff', bd=12, padx=17, pady=10).pack()

else:
    message = "Incorrect Username or Password"
    self.text.delete(0.0, END)
    self.text.insert(0.0, message)

app = Application(root)
root.mainloop()

```


APPENDIX-H

Code for Node MCU for controlling Appliances

```
WIFI_IOT_DEVICES | Arduino 1.8.5
File Edit Sketch Tools Help
WIFI_IOT_DEVICES
1 #include <ESP8266WiFi.h> // Including the WiFi Module Header file
2
3 String i; // Declaring the String var
4
5 const int bulb = 5; // Defining the Pins for Devices
6 const int fan = 4;
7 const int charger = 0;
8 const int table_lamp = 2;
9
10
11 WiFiServer server(80); // Making Object of Web Server HTTP at port 80
12
13 void setup()
14 {
15     pinMode(bulb , OUTPUT); // Setting the PINS to be as OUTPUT Pins
16     pinMode(fan , OUTPUT);
17     pinMode(charger , OUTPUT);
18     pinMode(table_lamp , OUTPUT); //
19
20     i = ""; // Initializing String as RAW String
21
22     Serial.begin(9600); // Starting Serial Monitor with Buad Rate of 9600
23     WiFi.disconnect(); // Disconnecting the Module at the Start
24
25     digitalWrite(bulb , HIGH);
26     digitalWrite(fan , HIGH);
27     digitalWrite(charger , HIGH);
28     digitalWrite(table_lamp , HIGH);
29
30     delay(3000);
31
32     Serial.print("STARTED.."); // Printing a Message at the Start
33     Serial.println("");

```

```
13 WiFi.disconnect();
14 delay(3000);
15
16 WiFi.begin("Hawk_Eye_Security","QyOuuRrt");
17 Serial.println("");
18 Serial.println("STARTED..!!");
19 Serial.println("Connecting to Hawk_Eye_Security");b
20
21 while (!(WiFi.status() == WL_CONNECTED)){
22     delay(300);
23     Serial.print(".");
24
25 }
26
27 Serial.println(WiFi.localIP().toString());
28 server.begin();
29 delay(1000);
30
31 }
32
```

```
94
95   if (i == "FanOn")
96   {
97     digitalWrite(4,LOW);
98     client.println("HTTP/1.1 200 OK");
99     client.println("Content-Type: text/html");
100    client.println("");
101    client.println("<!DOCTYPE HTML>");
102    client.println("<html>");
103    client.println("Fan is On");
104    client.println("</html>");
105    client.stop();
106    delay(1);
107  }
108
109
110  else if (i == "FanOff") {
111    digitalWrite(4,HIGH);
112    client.println("HTTP/1.1 200 OK");
113    client.println("Content-Type: text/html");
114    client.println("");
115    client.println("<!DOCTYPE HTML>");
116    client.println("<html>");
117    client.println("Fan is Off");
118    client.println("</html>");
119    client.stop();
120    delay(1);
121  }
122
```

Bibliography

- [1] IDC, Intel, United Nations
- [2] Strategy Analytics M2M Strategies advisory service, McKinsey Global Institute, NY Times
- [3] Krebs, Brian (September 21, 2016). "KrebsOnSecurity Hit With Record DDOS". Brian Krebs. Retrieved 17 November 2016
- [4] "Linux," [Online]. Available: <https://w1.fi/hostapd/>. [Accessed 20 11 2017].
- [5] Tasneem Yousuf, Rwan Mahmoud, Fadi Aloul and Imran Zulkernan, "Internet of Things (IoT) Security: Current Status, Challenges and Countermeasures", International Journal for Information Security Research (IJISR), Volume 5, Issue 4, December 2015

References

- 1) Naman Gupta, Srishti Sengupta and Vinayak Naik. "A FIREWALL FOR INTERNET OF THINGS". 9th International Conference on Communication Systems and Networks, 2017.
- 2) Tasneem Yousuf, Rwan Mahmoud, Fadi Aloul and Imran Zualkernan. "INTERNET OF THINGS (IoT) SECURITY: CURRENT STATUS, CHALLENGES AND COUNTERMEASURES". International Journal for Information Security Research (IJISR), Volume 5, Issue 4, December 2015
- 3) Guanglei Zhao, Xianping Si, Jingcheng Wang, Xiao Long and Ting Hu. "A NOVEL MUTUAL AUTHENTICATION SCHEME FOR INTERNET OF THINGS". Proceedings of 2011 International Conference on Modelling, Identification and Control, Shanghai, China, June 26-29, 2011
- 4) Hui Suo, Jiafu Wan, Caifeng Zou and Jianqi Liu "SECURITY IN THE INTERNET OF THINGS: A REVIEW". International Conference on Computer Science and Electronics Engineering, 2012.
- 5) Zaeniah and Bambang Eka Purnama "AN ANALYSIS OF ENCRYPTION AND DECRYPTION APPLICATION BY USING ONE TIME PAD ALGORITHM". International Journal of Advanced Computer Science and Applications (IJACSA), 2015.
- 6) M. Abomhara and G. M. Koien, "Security and privacy in the Internet of Things: Current status and open issues," in Int'l Conference on Privacy and Security in Mobile Systems (PRISMS), 1-8, 2014.
- 7) K. Zhao and L. Ge, "A survey on the internet of things security," in Int'l Conf. on Computational Intelligence and Security (CIS), 663-667, 2013.
- 8) L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (sIoT)—when social networks meet the internet of things: Concept, architecture and network characterization," Computer Networks, vol. 56, 3594-3608, 2012.
- 9) M. Leo, F. Battisti, M. Carli, and A. Neri, "A federated architecture approach for Internet of Things security," in Euro Med Telco Conference (EMTC), 1-5, 2014.
- 10) R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," Computer, vol. 44, 51-58, 2011.

- 11) R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, 2266-2279, 2013.
- 12) Q. Wen, X. Dong, and R. Zhang, "Application of dynamic variable cipher security certificate in internet of things," in *Int'l Conference on Cloud Computing and Intelligent Systems (CCIS)*, 1062-1066, 2012.
- 13) G. Zhao, X. Si, J. Wang, X. Long, and T. Hu, "A novel mutual authentication scheme for Internet of Things," in *Int'l Conference on Modelling, Identification and Control (ICMIC)*, 563-566, 2011.
- 14) N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, 203-209, 1987.
- 15) J.-Y. Lee, W.-C. Lin, and Y.-H. Huang, "A lightweight authentication protocol for internet of things," in *Int'l Symposium on Next-Generation Electronics (ISNE)*, 1-2, 2014.
- 16) Y. Xie and D. Wang, "An Item-Level Access Control Framework for Inter-System Security in the Internet of Things," in *Applied Mechanics and Materials*, 1430-1432, 2014.
- 17) B. Anggorojati, P. N. Mahalle, N. R. Prasad, and R. Prasad, "Capability based access control delegation model on the federated IoT network," in *Int'l Symposium on Wireless Personal Multimedia Communications (WPMC)*, 604-608, 2012.
- 18) M. Castrucci, A. Neri, F. Caldeira, J. Aubert, D. Khadraoui, M. Aubigny, et al., "Design and implementation of a mediation system enabling secure communication among Critical Infrastructures," *Int'l Journal of Critical Infrastructure Protection*, vol. 5, 86-97, 2012.
- 19) R. Neisse, G. Steri, and G. Baldini, "Enforcement of security policy rules for the internet of things," in *Int'l Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 165-172, 2014.
- 20) M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, "Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT)," in *Joint Intelligence and Security Informatics Conference (JISIC)*, 232-235, 2014.
- 21) I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal, "Choices for Interaction with Things on Internet and Underlying Issues," *Ad Hoc Networks*, 2015.

- 22) S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, 146-164, 2015.
- 23) W. H. Chin, Z. Fan, and R. Haines, "Emerging technologies and research challenges for 5G wireless networks," *Wireless Communications*, vol. 21,106-112, 2014.
- 24) X. Duan and X. Wang, "Authentication handover and privacy protection in 5G hetnets using software-defined networking," *Communications Magazine*, vol. 53, 28-35, 2015.
- 25) M. Brachmann, S. L. Keoh, O. G. Morchon, and S. S. Kumar, "End-to end transport security in the ip-based internet of things," in 2012 21st International Conference on Computer Communications and Networks (ICCCN). IEEE, 2012, pp. 1–5.
- 26) D. Altolini, V. Lakkundi, N. Bui, C. Tapparello, and M. Rossi, "Low power link layer security for IOT: Implementation and performance analysis," in 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, 2013, pp. 919–925.
- 27) S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the internet of things (IOT)," in International Conference on Network Security and Applications. Springer,2010, pp. 420–429.
- 28) S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad, "Proposed embedded security framework for internet of things (IOT)," in Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronics Systems Technology (Wireless VITAE), 2011 2nd International Conference on. IEEE, 2011, pp. 1–5.
- 29) P. Martin, "Using your new raspberry pi 3 as a Wi-Fi access point with hostapd," <https://frillip.com/using-your-raspberry-pi-3-as-aWi-Fi-access-point-with-hostapd/>, 2016, accessed: 2016-11-17.