

Aalto University  
School of Science  
Master's Programme in Computer, Communication and Information Sciences

Manish Thapa

# Mitigating Threats in IoT Network using Device Isolation

Master's Thesis  
Espoo, February 04, 2018

Supervisor: Professor N. Asokan  
Advisor: Ph.D. Samuel Marchal

<b>Author:</b>	Manish Thapa		
<b>Title:</b>	Mitigating Threats in IoT Network using Device Isolation		
<b>Date:</b>	February 04, 2018	<b>Pages:</b>	ix + 64
<b>Major:</b>	Mobile Computing, Services and Security		
<b>Supervisor:</b>	Professor N. Asokan		
<b>Advisor:</b>	Ph.D. Samuel Marchal		
<p>In recent years, the proliferation of the Internet of Things (IoT) is seen across various sectors. There is a sharp inclination towards using IoT devices in both home and office premises. Many traditional manufacturers are enhancing their traditional appliances into IoT devices. With the myriad of devices in the market, there also exist vulnerable devices which can be exploited by adversaries. Several security solutions are trying to address different areas of security such as network security, privacy, threat detection, etc. IoT Sentinel is one such novel system that can identify device types based on their pattern of communication. IoT Sentinel proposes several isolation levels that can be used to control the traffic of devices identified as vulnerable. IoT Sentinel uses a Software-defined Networking (SDN) component for controlling the traffic flow for devices and isolating them.</p> <p>In this thesis, we develop a solution to extend IoT Sentinel for device isolation, which is not dependent on SDN. The goal is to build a generic and deployable solution for network segmentation and device isolation that is suitable for home networks. The system divides the network into isolated subnets and places new devices into appropriate subnets. Communication between the subnets is controlled using a firewall thereby isolating them. We dynamically configure a DHCP server to place (lease IP address) new IoT devices identified by IoT Sentinel into appropriate subnets based on their level of vulnerability. Using our solution, we can confine vulnerable devices. Thus, the solution minimizes the damage that could be caused by vulnerable devices present in a network.</p> <p>Finally, we evaluate the developed solution for its security requirement of device isolation. We also present the performance evaluation of our solution based on time-delay and throughput analysis. We observe that our solution adds an acceptable delay to the existing IoT Sentinel processes. We also observe that the system throughput is not significantly affected by firewall rules in a home network scenario.</p>			
<b>Keywords:</b>	IoT, IoT Sentinel, Device Isolation, Network Segmentation		
<b>Language:</b>	English		

# Acknowledgements

I want to thank my supervisor Professor N. Asokan and my advisor Samuel Marchal from Aalto University for their support and continuous guidance.

I would like to thank my parents Mr Madhav Thapa and Mrs Meena Kumari K.C. (Thapa), my beloved wife Bimala K.C., my angel son Nirav Thapa, my brother Samujjwal Ram K.C., and my sister Parbati Subedi for the support they gave me in order to complete this thesis work.

I would like to thank my friend Gaurav Bhorkar for his never-ending motivational speeches and extensive proofreading help. I would also like to thank my other friends Sandeep Tamrakar, Kamal Panthi, Shiva Prasad Thagadur Prakash, and Sakib Nizam Khan for their countless discussions and suggestions during the work.

Otaniemi, Espoo, February 04, 2018

Manish Thapa

# Abbreviations and Acronyms

IoT	Internet of Things
AP	Access Point
BLE	Bluetooth Low Energy
WLAN	Wireless Local Area Network
IP	Internet Protocol
SDN	Software-defined Networking
SOHO	Small Office/Home Office
TCP/IP	Transmission Control Protocol / Internet Protocol
DHCP	Dynamic Host Configuration Protocol
BYOD	Bring Your Own Device
BS	Base Station
DMZ	Demilitarized Zone
EAP	Extensible Authentication Protocol
WEP	Wired Equivalent Privacy
WPS	Wi-Fi Protected Setup
PSK	Pre-shared Key
CVE	Common Vulnerabilities and Exposures
NAT	Network Address Translation
GUI	Graphical User Interface
JSON	JavaScript Object Notation
OTS	Off-the-shelf
UI	User Interface
IoTSSP	Internet of Things Security Service Provider
AAA	Authentication, Authorization, and Accounting
VLAN	Virtual Local Area Network

# Contents

Abbreviations and Acronyms	iv
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Contributions . . . . .	2
1.3 Organization . . . . .	3
<b>2 Background</b>	<b>4</b>
2.1 Internet of Things (IoT) Overview . . . . .	4
2.2 Threats in IoT Network . . . . .	7
2.3 IoT Sentinel . . . . .	9
2.3.1 System Design of IoT Sentinel . . . . .	9
2.3.2 Device Fingerprinting . . . . .	11
2.3.3 Device Type Identification . . . . .	11
2.3.4 Device Isolation . . . . .	12
2.3.5 Implementation . . . . .	13
2.4 Technical background . . . . .	14
2.4.1 Python . . . . .	15
2.4.2 Shell Scripting . . . . .	15
2.4.3 Raspberry Pi . . . . .	15
2.4.4 Hostapd . . . . .	16
2.4.5 DHCP . . . . .	16
2.4.6 Subnets . . . . .	18
2.4.7 iptables . . . . .	18
<b>3 Problem Statement</b>	<b>20</b>
3.1 Problem Statement . . . . .	20
3.2 Attacker Model . . . . .	21
3.2.1 Attacker Goals . . . . .	21
3.2.2 Attack Surface . . . . .	22
3.2.3 Attacker Capabilities . . . . .	22

3.3	Requirements . . . . .	23
<b>4</b>	<b>Proposed Solution and Implementation</b>	<b>25</b>
4.1	Proposed Solution . . . . .	25
4.1.1	Capabilities . . . . .	25
4.1.2	System Components . . . . .	26
4.1.3	Design . . . . .	27
4.1.4	Flow Diagram . . . . .	29
4.2	Implementation Choices . . . . .	31
4.2.1	Device Isolation and network segmentation . . . . .	31
4.2.2	Components . . . . .	32
4.3	Implementation . . . . .	32
4.3.1	Subnetting Approach . . . . .	32
4.3.2	DHCP Server Configuration . . . . .	34
4.3.3	iptables . . . . .	35
4.3.3.1	Rules for Isolating Subnets . . . . .	36
4.3.3.2	Rules for Enabling Internet Access . . . . .	36
4.3.3.3	Device Specific Rules . . . . .	37
4.3.3.4	Order of Evaluation of the rules . . . . .	38
4.3.4	Hostapd Configuration . . . . .	38
4.3.5	User Interface . . . . .	39
<b>5</b>	<b>Evaluation</b>	<b>41</b>
5.1	Security . . . . .	41
5.2	Functionality . . . . .	42
5.2.1	Performance . . . . .	43
5.2.1.1	Experimental Setup . . . . .	43
5.2.1.2	Timing Analysis . . . . .	44
5.2.1.3	Throughput Analysis . . . . .	46
5.3	Usability . . . . .	47
5.3.1	Deployment . . . . .	47
5.3.2	Demo User Interface . . . . .	48
<b>6</b>	<b>Related Work</b>	<b>49</b>
<b>7</b>	<b>Conclusions</b>	<b>53</b>
7.1	Summary of Contributions . . . . .	53
7.2	Future Work . . . . .	54
7.3	Conclusion . . . . .	55

<b>A</b>	<b>Configuration Files</b>	<b>62</b>
A.1	DHCP Configuration File . . . . .	62
A.2	Hostapd Configuration File . . . . .	64

# List of Figures

2.1	Smart home IoT Scenario [33]	6
2.2	IoT Sentinel System Design [52]	10
2.3	Sample Isolation Profile received from IoTSSP [53]	12
2.4	Isolation Technique in IoT Sentinel [52]	13
2.5	IoT Sentinel UI showing device identification [53]	14
2.6	DHCP Client/Server Communication	17
4.1	Network Isolation Design	27
4.2	Flow Diagram	30
4.3	Classing of Devices	34
4.4	Example declaration for trusted subnet	35
4.5	Iptables rule to prevent communication across subnets	37
4.6	Rules to enable Internet access for subnets	37
4.7	UI before isolation profile is obtained	40
4.8	UI after isolation profile is obtained	40
5.1	System throughput vs. number of iptables rules	47

# List of Tables

4.1	Network Segments Capabilities . . . . .	28
4.2	Network configuration for main network . . . . .	33
4.3	Subnet configuration . . . . .	33
5.1	Timing breakdown . . . . .	44
5.2	Timing Overhead Comparison . . . . .	45
5.3	Impact of Number of Rules in Throughput . . . . .	46

# Chapter 1

## Introduction

### 1.1 Motivation

The Internet of Things (IoT) is a popular trend and is already a part of our lives. Numerous vendors are currently producing a wide range of IP-connected devices targeted for homes and offices. The number of connected devices is expected to grow up to 50.1 billion by 2020 with a projected compound growth rate of 23.1 % between 2014 and 2020 annually [43].

In a home network, users typically connect their IoT devices to an AP (Access Point) with authentication methods such as Wi-Fi Protected Access (WPA), Wi-Fi Protected Access II (WPA2), Wi-Fi Protected Setup (WPS). Typically, all the devices become part of a single network provided by the access point. IoT devices available in the market have a varying level of security features. Many devices are sold by traditional home appliances vendors, which may not have adequate expertise in the security domain. As a result, several IoT devices are often shipped with flawed security designs and vulnerabilities. Moreover, several vendors do not provide security updates for their vulnerable devices in a timely manner. Therefore, there is a high possibility of vulnerable devices being a part of a user's network. Vulnerable devices in the network provide an opportunity for adversaries to mount attacks by exploiting vulnerabilities and perform malicious activities, for example, gaining unauthorized access to the user's network or associated devices. Therefore, there is a need for a mechanism to identify vulnerable devices and control their communication in a network and mitigate the security risks involved.

IoT Sentinel [52] is a system that protects a user's network from vulnerable devices by identifying and isolating them. The system provides an access point (AP) for IoT devices. IoT Sentinel identifies the type of a device and infers its vulnerability level. The information provided by IoT Sentinel is

called an isolation profile which defines abilities of a device for communication. The communication of the vulnerable devices is controlled to mitigate the risks associated with it by isolating it from legitimate devices. IoT Sentinel uses a Software-defined Networking (SDN) controller to isolate devices. Traditional APs used in home networks usually do not support SDN, which limits the deployment of IoT Sentinel in home networks. In this thesis, we aim to find a generic solution which does not rely on SDN for device isolation.

The goal of this thesis is to develop a solution for wireless and wired device isolation which is easily deployable for home networks. The solution is developed as an extension to IoT Sentinel. We aim to isolate devices in the network into different network segments and use generally available mechanisms for network segmentation and access control. The solution is intended to be simple, easy to use and to configure.

The proposed solution divides the user's network into different segments using a subnetting approach. The devices in different subnets have communication capabilities according to their isolation profile. A DHCP server is used to define the network topology and allocate devices into different subnets. The communication across subnets is controlled using a firewall. The solution is based on customized DHCP server that uses the isolation profile from IoT Sentinel and isolates the devices by allocating them into appropriate subnets.

## 1.2 Contributions

The contributions of this thesis are:

- **A technique for automatically enforcing isolation profiles from IoT Sentinel using commonly available network primitives**

Our solution integrates with IoT Sentinel and automatically enforces the isolation profiles which are used to perform device isolation. IoT Sentinel uses an SDN based approach to utilize isolation profiles to perform device isolation. We proposed and implemented a mechanism to perform device isolation which uses commonly available network primitives and does not rely on SDN. Our technique divides the network into different isolated network segments. Different network segments are constructed by subnetting the network. Communication across the network segments is controlled by `iptables` rules. A DHCP server is customized to provide IP addresses in order to place devices into required network segments. We enforced isolation profiles to control the communication of a device with other devices in the network and the Internet.

- **Evaluation of the implemented device isolation approach.**

We evaluated our device isolation solution to measure the delay overhead incurred on the IoT Sentinel setup for device identification. We observed a timing delay of +42.07% and +35.6% over IoT Sentinel device identification while performing isolation for two test devices. We also evaluated the effects of the number of firewall rules on the system throughput and observed that having 1000 `iptables` rules decreased the throughput by 2.72%. We also discussed the deployability of the proposed solution.

### 1.3 Organization

We now present the organization of this thesis and the information that is contained in subsequent chapters.

**Chapter 2** describes the background information about IoT and the security in IoT networks. We also give an overview of the IoT Sentinel system and explain the related technical background for our work. **Chapter 3** describes the problem statement and the attacker model. We also list and explain the requirements for our solution. **Chapter 4** describes the proposed solution and the implementation details of the system. **Chapter 5** evaluates the implemented system against the requirements and discusses the results. **Chapter 6** presents the related work. **Chapter 7** summarizes the contributions made by thesis, provides some directions for future work and draws conclusions.

## Chapter 2

# Background

### 2.1 Internet of Things (IoT) Overview

Back in 1988, Mark Weiser coined the phrase “ubiquitous computing” which is referred to as a concept where computing is available anywhere and any-time [65]. The term “Internet of Things” was first coined by Kevin Ashton in 1999 [49]. Internet of Things, commonly referred as IoT is an ubiquitous concept where physical objects are connected to the Internet and have an ability to communicate over a network [56]. Such networked objects or things are referred as IoT devices and are deployed universally. The penetration of IoT has been seen across numerous sectors such as building and home automation, smart cities, smart manufacturing, industrial automation, automobiles, wearables, healthcare, farming etc. The power of connectivity offered by IoT has been utilized in various use cases related to such sectors. According to the analysts from the firm Juniper Research, the number of expected IoT connected devices will reach 38.5 billion by 2020 [47]. Therefore, IoT is one of the most current and popular areas of research in which many big companies are investing billions in research and development [60]. Having the extremely large number of interconnected devices, configuration and management of these devices do not seem feasible if we do not have automated approaches.

One of the application areas of IoT is Smart home. A smart home is a home with intelligent IoT household devices, which provides better living conditions to the owner. Smart home IoT devices are bought and installed in the home in order to improve the overall comfort and accessibility which makes regular household jobs automated, easier and better. The applicability of IoT devices in the home can be seen in different scenarios, for example, controlling of home lights efficiently using smart bulbs, automatically main-

taining the temperature and humidity of the house using smart thermostats, securing the entrances etc. People store lots of confidential and personal information across such smart household devices. Such devices are connected to the Internet via the home network using the router equipment. Home is a place where people consider themselves safe and keep their belongings safe. So, maintaining the privacy of the user and securing the home network is an important task. IoT devices communicate using a different set of protocols available for communication. Different wireless standards like Wi-Fi [39], Bluetooth LE [26], ZigBee [29], Z-Wave [31] etc, exist are used to make IoT Smart home automation feasible [61] .

Figure 2.1 shows the typical IoT smart home scenario. A smart home consists of several smart devices. The major components in the figure are Access Point (AP), IoT devices, the Internet, the cloud service, and a control device (smartphone). AP is the central hub that connects different IoT devices to the external world. Usually, an AP is connected to the wired network and provides a point of communication among devices. Several IoT devices, for example, smart switch, smart fridge, smart TV, surveillance camera, thermostat, smoke detector etc are deployed in the home. These different devices have different features and tasks to support home automation. The control device is typically a smartphone which is used to configure and monitor different IoT devices. The Internet is the major component that connects the IoT devices, AP, the control device, and cloud services together. Currently, most of the IoT devices requires vendor specific smartphone application for configuration and management purpose [4, 25] through the control device.

Whenever an IoT device is initially configured, it connects to AP for authentication using vendor-specific smartphone application installed on the control device. This initial configuration requires physical access to the device for resetting purpose which improves the security issues that may arise due to unattended access. Dynamic Host Control Protocol (DHCP) server is a component in the router that provides IP connectivity information to devices. Once authentication is successful, IoT device gets the IP configuration from the DHCP server and becomes part of the network provided by AP. These devices are controllable remotely via the control device. Most IoT devices usually communicate with their vendor-specific cloud service to store and process data. Such data can be user privacy sensitive, for example surveillance cameras may send live capture of the room to their respective cloud service. The control device is used to control and monitor the IoT devices using a registered user account which displays data in a presentable way through the application.

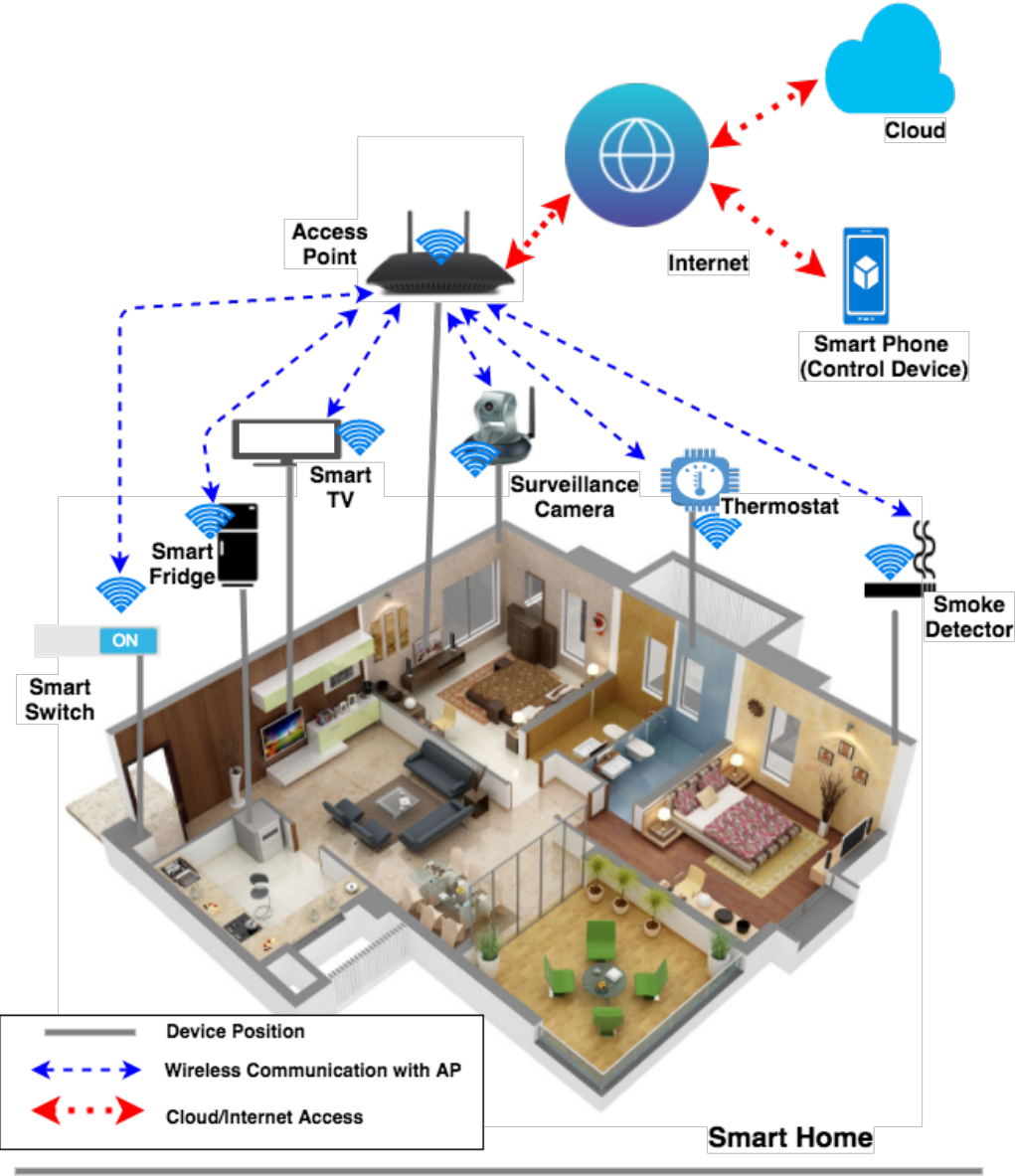


Figure 2.1: Smart home IoT Scenario [33]

## 2.2 Threats in IoT Network

The emergence of IoT has brought enormous possibilities and probable use cases that affect our lives from different aspects. With the upsides comes downsides as well. IoT has been increasingly popular and attractive even to attackers. Security problems in IoT environment arise due to careless program design of IoT devices and heterogeneous interconnected complex protocols [67]. Several attacks targeting connected devices have been reported in the media and include popular brand names like Philips [41, 44]. It has also been reported that even a single flaw can affect a wide range of products from WiFi cameras, camera recorders and cloud storage devices from the same vendor [62]. A single fault affected multiple devices since manufacturers reuse vulnerable code across different device models. Recent news about two hackers remotely controlling the air-conditioning, radio, windshield wipers and even an accelerator of a jeep [45] present the security loophole in IoT systems installed in the jeep. Such kind of failure can be life-threatening for a driver of a jeep. There has been news [46] about WikiLeaks leaked document which reports that some models of Samsung televisions are also vulnerable. Those vulnerable TVs secretly record audio when the TV screen is off and send it to Central Intelligence Agency (CIA) server when TV is turned on and its Internet connection is re-established. In 2014, it was also revealed that more than 100,000 consumer gadgets including home networking routers, televisions, and refrigerators were targeted for a large-scale attack to send 750,000 malicious emails to individuals and business enterprises [12].

The stake of having vulnerable devices in the network is quite high. There is a strong urge to identify and control vulnerable devices before we grant them access to our home network. Securing a home network is difficult to achieve since numerous threats like denial of service, back door, and remote administration programs, malware etc exists [13]. With the enormous potential of IoT, it brings security and privacy concerns. Some IoT devices that are deployed in users' network are vulnerable and can be exploited by attackers [46]. Since IoT devices are capable of connecting to the Internet, vulnerable IoT devices are attractive for attackers to target different kinds of attacks to other devices in the network and breaking into the network [57]. One solution to deal with such vulnerable devices is to patch them with an updated security solution. However, most device manufacturers do not produce patch on time because of associated overhead cost for support. It is also seen that production and support for devices are discontinued after the short period of time of its launch [1, 19].

Ordinary access points provided by Smart Office Home Office (SOHO)

routers which are widely used in home networks lack security measures such as wireless client isolation, guest network allocation, updated encryption modules etc. Naive users do not have adequate knowledge about the need for updating the router firmware. For a normal user, updating a firmware is a daunting task since it requires series of manual task. Thus old routers include a high risk of being exploited due to lack of proper security mechanism and fixes.

Whenever a device joins the AP, it is authenticated by wireless authentication mechanism implemented at the router. After successful authentication of IoT device with AP, IoT device is granted IP connectivity. IP connectivity leases are provided by DHCP server. In this kind of scenario, security provided by the AP is a crucial factor to protect the network. If AP is not protected with the security feature to restrict the capability of vulnerable devices, an attacker can use that vulnerable device to attack other devices in the network. For example, any compromised surveillance camera in explained home setup may upload the live feed to some remote server so that attacker can have 24/7 live view of a home. This kind of situation breaches the overall security of the home which is fully against the purpose of having a camera at home. So, identification of device before granting access to the network is highly important. With proper identification of the device and defining its security boundaries, security of the overall network can be maintained.

A Wi-Fi network is one of the widely used ways to connect to the Internet. When IoT devices want to associate with Wi-Fi network, there is a need of establishing security relationship between IoT device and a Wi-Fi access point. The devices that want to connect to Wi-Fi networks are also referred to as supplicant. Wifi-networks make use of authentication protocol to build the trust relationship and verify the client. Different wireless security protocols, for example, WEP, WPA, WPA2-Personal, WPA2-Enterprise etc exists. Such security protocols are evolving and are enhanced to address the security need of a wireless network. These protocols ensure data confidentiality and integrity. Choosing older security protocols which are prone to dictionary-based brute force attacks questions the overall security of an access point. An attacker can perform an attack to extract the passphrase which can be used to join the network.

The need for identification of IoT devices and defining their capabilities for communication in a network is addressed in a recent work named IoT Sentinel [52]

## 2.3 IoT Sentinel

IoT Sentinel is a solution that can automatically identify the device type whenever a device is introduced in the network. It makes use of network traffic of a device during initial bootstrapping for device type identification. Device identification can help to define if the device is vulnerable or not. Based on its vulnerability assessment, IoT Sentinel provides an isolation profile for a device that can be used to define the capabilities of a device for further communication. After identification of a vulnerable device, proper protection mechanisms are deployed such that it can coexist with the other trusted devices in the network without affecting the security of the network. IoT Sentinel tries to achieve such protection by controlling the traffic flow of vulnerable devices.

### 2.3.1 System Design of IoT Sentinel

The system design goal of IoT Sentinel is based on the requirement of a *brownfield approach*. The term *brownfield approach* refers to scenarios where a solution needs to be integrated into an existing legacy system and the option of developing a new solution from the ground up is not available. The authors of IoT Sentinel [52] have defined a device-type to denote the combination of make, model and software version of a device. IoT Sentinel has two major components. The first component is the Security Gateway and another one is IoT Security Service. The design of IoT Sentinel is shown in Figure 2.2.

The Security Gateway is the core of IoT Sentinel implementation which collaborates with the IoT Security Service hosted in the cloud. Security Gateway acts as an access point for IoT devices to associate with the network. Security Gateway is deployed locally in a device capable of providing a wireless access point. Security gateway fingerprints new devices whenever they are first introduced to the network. Fingerprint capture of a device consists of some distinct communication pattern that can be used in the identification of the device. Such fingerprints are then sent to IoT Security Service for device type identification and vulnerability assessment. Once device fingerprint is received by IoT Security Service Provider (IoTSSP), it uses a machine learning-based device identification technique to compare the received fingerprint with existing reference fingerprints. After a comparison is made, the correct device type is identified. IoTSSP also maintains the database of isolation profile linked with the device type. Based on device type identified, a corresponding profile for device type is retrieved. IoT Se-

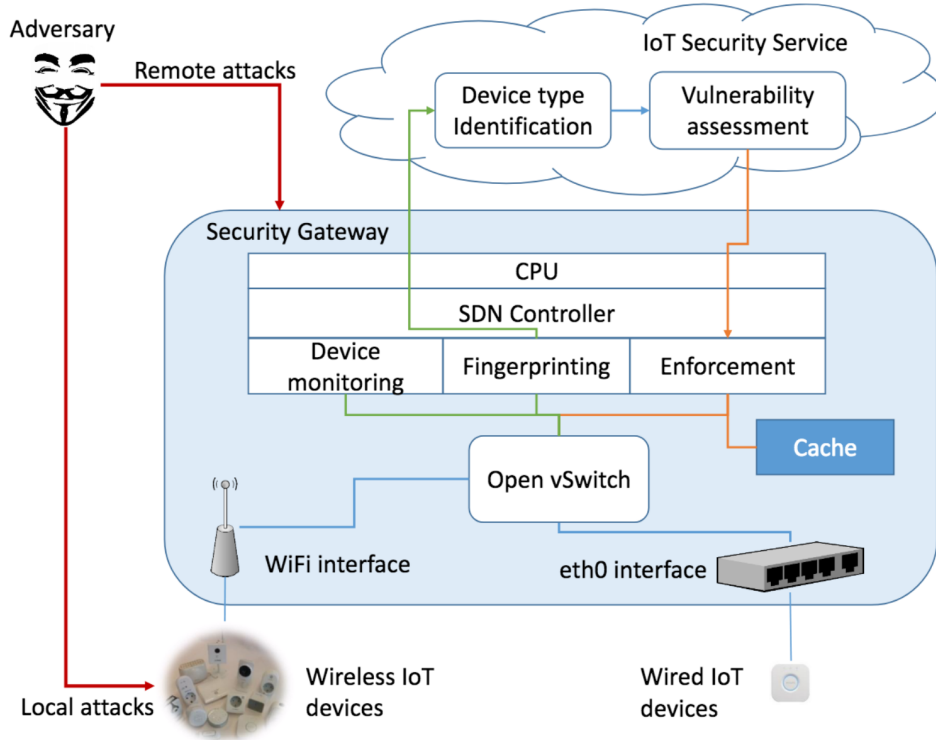


Figure 2.2: IoT Sentinel System Design [52]

curity Service returns back the corresponding isolation profile to the security gateway. Isolation profile contains information about the device name, isolation level, permitted IPs etc. An isolation profile is used by security gateway to grant access to the network for a device. After the device is identified by IoTSSP and isolation profile is received by the security gateway, some device isolation technique must be in place to utilize those isolation profiles and provide network access to the device. The technique adjusts network configuration settings and necessary enforcement rules at security gateway to ensure that the device gets connected to the network with predefined capabilities in isolation profile. The design of a system to utilize the isolation profile and maintain enforcement rules at security gateway ensures the segregation of the devices into different network segments.

### 2.3.2 Device Fingerprinting

Whenever a new IoT device is introduced to the network, we capture the communication involving the Wi-Fi association to build the fingerprint. The fingerprinting process is done using passively observed traffic. IoT devices have a vendor-specific procedure to associate the device to the gateway and induct into the network. So, the aim of fingerprinting procedure is to capture the distinguishable sequence of communication between the device and the gateway. When a newly observed MAC address is identified at the gateway, the fingerprint capturing process collects  $n$  packets  $\{p_1, p_2, p_3, p_4, \dots, p_n\}$ . Since device configuration time varies across different devices, a standard timeout is set to capture all the packets in two minutes. After the fingerprinting process is over, the feature extraction is done. 23 different features are extracted from each packet which generates a  $23 \times n$  matrix  $\mathbf{F}$ , which is a device fingerprint. Packet features include different information about the typical protocols and features used during device association in Wi-Fi network. Authors then compressed the initial fingerprint  $\mathbf{F}$ .  $\mathbf{F}'$  is a compressed fingerprint generated from  $\mathbf{F}$ , which is composed of 12 first unique vector packets  $p$  to produce 276-dimensional feature vector (**12packets**  $\times$  **23features**).  $\mathbf{F}'$  is used as input for device type identification.

### 2.3.3 Device Type Identification

The device type identification is done in two steps. The first step is fingerprint classification, which relies on  $n$  classifiers. The device type identification process refers to the process of analyzing the fingerprint generated to identify the device type. For device type identification, the prerequisite is to have a classifier for each device type with reference fingerprints. They are implemented as Random Forest classifiers [36] which renders a binary decision if the input fingerprint matches the device type. When a new fingerprint  $\mathbf{F}'$  goes through the classification process, a prediction of the limited number of reference fingerprints for the candidate device types is done. In the second step, the edit distance discrimination is computed by comparing the fingerprint with the subset of fingerprints matched after the classification process. The comparison is done by computing Damerau-Levenshtein distances [40]. The result of distance computation is used to get a global dissimilarity score with all the reference fingerprints. The lowest dissimilarity score gives the final predicted device type for the fingerprint.

### 2.3.4 Device Isolation

Device Isolation refers to the separation of a device in different network segments. Once a device type is identified, IoTSSP extracts the corresponding isolation profile and returns it to the security gateway. The isolation profile is a JSON file that consists of a device name, isolation level, allowed IPs, and ID. The isolation level defined in the isolation profile determines which part of the network the device should be placed in. The isolation level values, 0, 1, and 2 correspond to strict, restricted and trusted level of isolation respectively. Figure 2.3 represents a sample isolation profile sent by IoTSSP. The sample isolation profile shows a device named ABCDEF with ID 1 and isolation level of 1 which implies that device should be placed in the restricted part of an untrusted network. The `allowed_ips` field specifies that the device should be able to access IP address 13.20.224.22 but should not have access to the rest of the Internet. The significance of different isolation levels is explained further below.

```
{
  "name": "ABCDEF",
  "allowed_ips": ["13.20.224.22"],
  "isolation": 1,
  "id": 1
}
```

Figure 2.3: Sample Isolation Profile received from IoTSSP [53]

Figure 2.4 depicts the isolation approach of IoT Sentinel. IoT Sentinel divides the user's network into two network overlays: an untrusted (isolation level 0 and 1) and a trusted network (isolation level 2). Devices within these overlays should not communicate with each other. The trusted network should have unrestricted Internet access. An untrusted network is further divided into strict (isolation level 0) and restricted (isolation level 1). Internet connectivity should not be available for the strict part of the untrusted network. The restricted part of the untrusted network should have limited Internet access. Untrusted devices which may need a limited set of vendor-specific cloud service access are placed on the restricted network. All vulnerable devices are isolated in the untrusted network so that they cannot mount attacks on the trusted devices. Enforcement rules are necessary to control the traffic in the network. Prior to our work, the enforcement rules in the network for these overlays are generated by a customized SDN controller according to the isolation profile. Such enforcement rules make network

isolation feasible by intercepting all traffic flows in the network and filtering them.

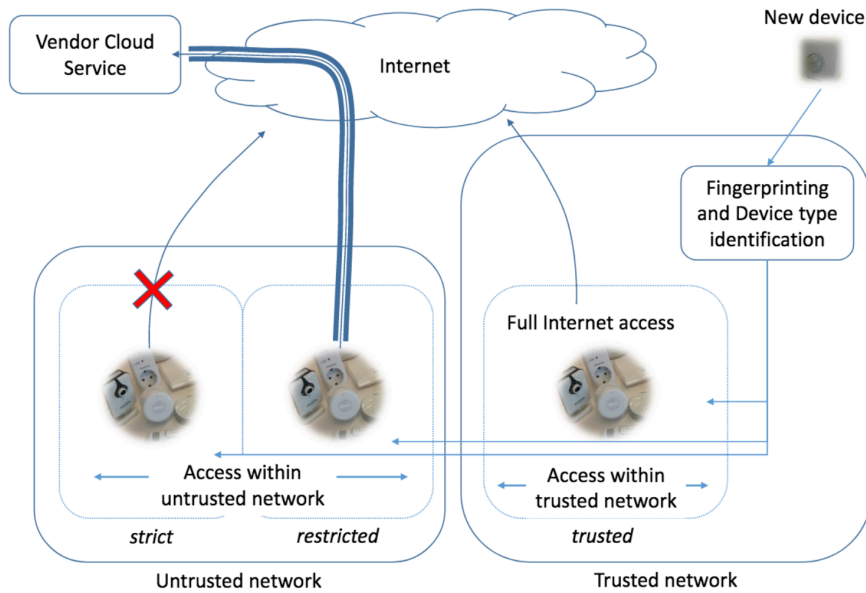


Figure 2.4: Isolation Technique in IoT Sentinel [52]

### 2.3.5 Implementation

The prototype implementation of IoT Sentinel consists of two major components, Security Gateway and a cloud-based IoT Security Service [53].

The Security Gateway is implemented in Raspberry Pi 3 device which sets up a wireless AP in infrastructure mode which ensures that all clients communicate only via the AP. The wireless AP is configured using `hostapd` [50]. `hostapd` consists of a configuration file named `hostapd.conf` which is used to configure different parameters required to set up an AP. The prototype implementation uses open-source Floodlight SDN controller v1.2 [32] which is customized to detect new devices in the network, initiate device fingerprinting and communicate with IoT Security service.

IoT Security Service is the service that is hosted in the cloud. IoT Security Service Provider in cloud hosts the machine learning based approach of IoT

Sentinel to identify the device type and return isolation profile.

Security Gateway and IoT Security Service communicates using restful API over HTTP/HTTPS.

A user Interface is developed for demo purpose and testing. User Interface depicts the real-time display of IoT Sentinel operations when a new device is detected at the gateway. It can provide information about current state of multiple devices being configured. The user interface is developed using Autobahn server [2]. Autobahn server allows tracking of real-time messaging between websockets which can be displayed in the browser. Figure 2.5 shows the user interface of the device going through IoT Sentinel device identification process. When a new device is encountered at the gateway, device features are extracted, compressed and sent to IoTSSP for classification. As seen in Figure 2.5, classification results are obtained which specifies that the device is identified as NetAmo whose isolation level is 2.

```
Start to capture packets: 2017-09-06 11:11:52.680587
Capturing. 79 seconds left.
Capturing. 69 seconds left.
Capturing. 59 seconds left.
Capturing. 49 seconds left.
Capturing. 39 seconds left.
Capturing. 29 seconds left.
Capturing. 19 seconds left.
Capturing. 9 seconds left.
Capturing. 86399 seconds left.
Capturing terminated: 2017-09-06 11:13:22.848548
Feature extraction completed: 2017-09-06 11:13:28.466992
Start to compress features: 2017-09-06 11:13:28.467556
Compression completed: 2017-09-06 11:13:28.763393
Start classification: 2017-09-06 11:13:28.764055
Classification results received from IoTSSP cloud server
{
  "allowed_ips": [],
  "id": 2,
  "isolation": 2,
  "name": "NetAmo"
}
Device classified as NetAmo with isolation level 2
Classification completed: 2017-09-06 11:13:30.093239
End to process the new device: 2017-09-06 11:13:31.231540
```

Figure 2.5: IoT Sentinel UI showing device identification [53]

## 2.4 Technical background

This section is intended to give a general overview of different technologies, tools, and components used in the implementation.

### 2.4.1 Python

Python [28] is an interpreted, object-oriented, high-level programming language. Python is highly readable and can be applied to solve various problems both in product development and research prototypes. Python has been extensively used for scientific computing, web development, scripting, development of games etc. The Python implementation consists of numerous built-in and standard modules. Python distributions are available for multiple platforms and are commonly used across most used operating systems (OS X, Windows, and Ubuntu). During the work, versions 3.4 and 3.5 of Python are used across different components. Several packages like Numpy, SciPy, and Pandas are available for Python which provides the basis for efficient scientific computing. These packages provide high dimensional data structures and tools to work with it.

### 2.4.2 Shell Scripting

Shell Scripting [3] refers to the way of creating scripts that can be used in order to execute the tasks in a run time environment. Scripts help to automate the tasks according to the user goals. Bash shell scripts are used widely during the implementation for automating the experimentation of prototype, invoking other external components, and maintaining the appropriate network configurations.

### 2.4.3 Raspberry Pi

Raspberry Pi is a single board basic computer developed by the Raspberry Pi foundation that is intended to raise the interest in computing for school going children [23]. It is equally popular for different purposes in academic research and prototype development because of its low cost. Several different versions of Raspberry Pi are available in the market. The considered model for implementation is Raspberry Pi 3 Model B which has support for Bluetooth and wireless LAN connectivity. Raspbian [7] is a free Debian based officially supported operating system which is optimized especially for Raspberry Pi hardware. Minibian [18] is the minimal version of Raspbian which does not have a GUI. The goal of using Minibian is to maximize resources utilization of Pi for our solution. Raspberry Pi device can be used as a wireless AP for prototype development.

### 2.4.4 Hostapd

`hostapd` stands for Host Access Point Daemon [50]. It is a software that can be used to turn a network interface into an access point. Using `hostapd`, it is possible to configure any device that supports Wireless Local Area Network (WLAN) in access point mode to a wireless base station. Many of the appliances that act as WLAN router are Linux computers which run `hostapd` software. `hostapd` implements IEEE 802.11 AP and IEEE 802.11/WPA/WPA2/EAP/RADIUS Authenticators [50]. The configuration file for `hostapd` can be modified to set parameters such as listening wireless interface, Service Set Identifier (SSID) of a network, authentication algorithm to use such as WPA or WEP or both, passphrase to connect to network etc.

The 802.11 specification defines two modes of operation for wireless communication, namely, ad-hoc mode and infrastructure mode. In ad-hoc mode, all clients can communicate directly with each other, whereas in infrastructure mode, clients are connected to a central access point through which all communication passes. The infrastructure mode provides increased wireless range and additional security controls [30].

### 2.4.5 DHCP

The Dynamic Host Control Protocol (DHCP) [42] is a network protocol that is widely used to automatically distribute network configuration parameters (eg. IP addresses) for hosts in a network. DHCP simplifies and automates the job of the network administrator to handle multiple devices and systems. There are different DHCP servers available, such as, `isc-dhcp-server` [14], `dnsmasq` [5] etc. The DHCP server can be customized for specifying fixed IP address for hosts and managing groups of devices.

The DHCP communication between the client and server includes four different kinds of messages explained below.

1. **DHCPDISCOVER:** During the initialization state, when a client needs IP connectivity, it broadcasts DHCPDISCOVER packet on the network.
2. **DHCPOFFER:** The DHCP Server responds to the DHCPDISCOVER packet with a DHCPOFFER packet. The DHCPOFFER packet contains the IP address allocated to the device, default gateway, subnet mask, etc.
3. **DHCPREQUEST:** When a client receives the DHCPOFFER packet from the server, it responds back with DHCPREQUEST packet for an IP address offered by the server.

4. **DHCPACK**: The server sends back the DHCPACK packet as an acknowledgement to the DHCPREQUEST packet stating that the client can use the IP address for the amount of time leased.

Figure 2.6 depicts the DHCP client-server communication and the three different states of a DHCP client. In the Initialization state, a DHCP client does not have an existing lease for an IP address. To get a new lease, DHCPDISCOVER, DHCPPOFFER, DHCPREQUEST and DHCPACK packets are exchanged between the client and the server. The lease is limited for a certain amount of time as specified by the server.

Once half of the lease time expires ( $T_1$ ), the DHCP client resends DHCPREQUEST packet to the server asking for the renewal of the existing lease. If DHCP server has availability of address, it sends back DHCPACK back to the client which ensures the renewal of the lease.

If the renewal process is not successful after 87.5% of original lease time ( $T_2$ ) amount of time, the DHCP client goes in the rebinding state. The client broadcasts DHCPREQUEST messages to attempt to contact any available DHCP server. If it receives DHCPACK from any server, the rebinding is successful and client existing lease is renewed.

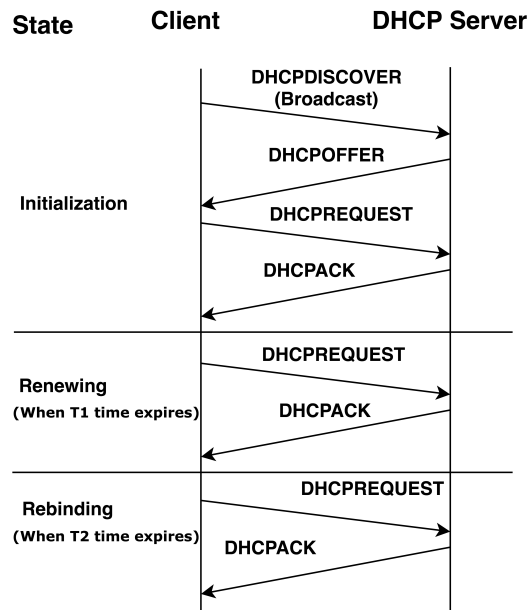


Figure 2.6: DHCP Client/Server Communication

## 2.4.6 Subnets

A subnet is a logical division of an IP network [54]. The process of creating subnets is commonly known as subnetting. Subnetting help to make clear separation within a network. Two different kinds of addressing schemes are available for IP addresses. IPv4 is a 32-bit addressing scheme while IPv6 is a 128-bit addressing scheme. IPv4 address is a 32-bit 4 octant integer [37]. The value of each octant varies from 0 to 255. A single IP address basically consists of two parts: the network part and the host part. IP addresses are represented using Classless Inter-Domain Routing (CIDR) notation. For example, if we consider IP address 192.168.1.0/24, the /24 suffix at the end specifies that the first 24 bits of the IP address represents network part and last 8 bits represent the host part. We can borrow host bits to define subnets depending on the number of hosts we need in our network. If we borrow  $n$  bit from host part we can have  $2^n$  subnets with  $2^{(8-n)}$  hosts in each subnet. The efficient creation of subnets depends upon the requirements of a network.

## 2.4.7 iptables

Efficient filtering rules are essential in order to control the communication in a network. The Linux kernel has an inbuilt network packet processing system known as Netfilter. The configuration of Netfilter is done using `iptables` command. `iptables` is widely used in order to create, maintain, and inspect the tables that are used for IPv4 traffic filtering. Using `iptables`, we can define such tables which consist of a set of rules. Using such rules we can filter/drop the packets. `iptables` operates at the network layer. We can define several tables containing built-in chains or user-defined chains. A chain consists of the set of rules that match the certain set of packets. Different special values like ACCEPT, DROP, QUEUE or RETURN can be specified in the rule to determine if the packet can be transmitted through the interface. Processing of packets is done sequentially across the rules in the chain. One simple example of `iptables` rule is `iptables -A INPUT -i eth0 -p tcp -s 192.168.1.20 -j DROP`. The rule ensures that TCP packets whose source is 192.168.1.20 are dropped at interface eth0. We can write multiple rules based on our requirement which can be defined based on device, network, port, interfaces etc.

`iptables` rules can also be used to enable Internet connectivity for networks that are behind a Network Address Translation (NAT). IP masquerading is a networking function that is used to provide connection to the Internet for devices with the private IP address that are connected to the Linux host with an Internet connection [16]. IP masquerading masks the requests from

nodes with the IP address of the host's external device. For example, the following rule allows requests coming from source network 192.168.0.0/16 to be masked with the IP address of output interface eth0, thereby, enabling Internet connectivity.

```
iptables -t nat -I POSTROUTING -o eth0  
-s 192.168.0.0/16 -j MASQUERADE
```

## Chapter 3

# Problem Statement

### 3.1 Problem Statement

The IoT ecosystem consists of multiple devices from different vendors with varying security designs and implementations. In a typical home environment IoT setup, several IP-connected IoT devices coexist with different gateways, cloud-based systems, and other devices. Attackers are interested in the vulnerable devices in the home network for various malicious reasons such as launching an attack against other devices in the network, exfiltrate personal data etc. Another example is that an eavesdropper can illegitimately gain access to a vulnerable device or sniff communication in the network. Thus, the presence of vulnerable IoT devices in the network introduces privacy and security concerns. Legitimate devices may also collect privacy sensitive data without the consent of the user. It is crucial to address the security of the home network effectively so that the IoT ecosystem runs with minimum security issues.

We present the following scenarios in which IoT systems can bring security and privacy issues. Consider a malicious smart television, which is connected to the Internet that secretly records audio and sends it to some remote server without the knowledge of the owner. This scenario is presented by Hartley-Parkinson [46] in detail. In such a case, the malicious device must be identified and restricted from the home network. Consider another scenario where all the devices in a home are on a single unsegmented network. If some device is vulnerable, an attacker can access other devices or the network via the vulnerable device as a backdoor and perform malicious activities.

There is a strong need for isolating communication of vulnerable devices. A vulnerable device must be isolated such that its communication with other trusted devices in the network is restricted. If a potentially vulnerable device

connects to the network, it should coexist with other devices without affecting the level of security in the network. However, isolation should not affect the communication for secure devices. IoT Sentinel proposes a mechanism to identify vulnerable devices and protect the network against such devices.

IoT Sentinel provides the device identification service at the access point. In addition to device identification, IoT Sentinel implements device isolation in the SDN controller. SDN is more suitable for managing a large network which needs to adapt dynamically. Implementing SDN for home networks adds complexity and needs proper network administration skills. Moreover, most legacy off-the-shelf (OTS) routers do not support SDN. Thus, there is a need for a generic network segmentation solution implemented in the security gateway which can be easily deployed in home networks. The solution should replace the SDN component used in IoT Sentinel for device isolation. The new solution is intended to be a software solution, which removes the complicated aspect of SDN aided approach but provides a similar functionality for device isolation. The solution must control communication of IoT devices within the IoT network and to the Internet. Device isolation in IoT Sentinel is based on the isolation profile of a device. Therefore, enforcement of the isolation profile obtained from IoT Sentinel provides the base of our work.

## 3.2 Attacker Model

To build a secure system, it is important to identify the threats to the system [55]. IoT Sentinel is targeted at a typical home or small office network setup. The AP provides IP connectivity to various wired and wireless devices. The authors of IoT Sentinel have assumed that when an IoT device initially connects to the AP, it might have security vulnerabilities but is initially considered benign [52]. In this section, we present an attacker model for a Wi-Fi network with respect to IoT Sentinel. We explain the attacker goals, attack surface, and attacker capabilities.

### 3.2.1 Attacker Goals

The primary goal of an attacker is to compromise a vulnerable device. An attacker may want to use a compromised device as a stepping stone to launch attacks against other devices in the network. The attacker may communicate with a compromised device remotely from the Internet. If attackers have control over a compromised device in a network, they may spread malware or viruses to other devices in the network. An attacker may want to sniff,

tamper or obstruct the communication between the device and the AP or other devices. The attacker may want to tamper network parameters, exfiltrate data to the Internet, obtain security credentials etc. The attacker may use the compromised device to attack devices or services on the Internet.

### 3.2.2 Attack Surface

An attack surface refers to the entry points in the system where an attacker can get unauthorized access and perform malicious activities. In a home network, vulnerable devices form a major part of the attack surface and can be exploited to compromise different aspects of a network. The compromises may be related to disrupting the communication channel, the AP, or the devices themselves. The communication channel between a device and the AP forms an attack surface since it can be subject to several attacks such as sniffing and tampering. The direct device to device communication can also be considered as an attack surface since it is beyond the control of the gateway. We consider that the attacks on the devices and network are remote rather than physical. Although even physical access to the devices can lead to tampering.

### 3.2.3 Attacker Capabilities

A compromised IoT device serves as the entry point to the network for an attacker. An attacker can have control over a compromised device. If communication of IoT device is unencrypted, an attacker can passively monitor traffic [35] in the communication channel. An attacker can use compromised devices to communicate with other legitimate devices in the network. An attacker can introduce malware through a compromised device and spread the infection to other devices. An infected IoT device can be used to locate other vulnerable devices on the Wi-Fi network. A compromised device can open an attack path for remote attackers (e.g. through an SSH tunnel). An attacker can apply reverse engineering techniques on a compromised IoT device to find and use backdoors (created for testing) in order to exploit the device [67]. An attacker can capture the 4-way handshake between a device and the AP and perform a dictionary attack on the message integrity data to find out the passphrase [51].

### 3.3 Requirements

This section explains the requirements for the solution to address the problem statement explained in Section 3.1. Requirement S1 is a security requirement. Requirements R1, R2, and R3 are functional requirements while requirements U1 and U2 are usability requirements.

#### **S1: Restricting communication of vulnerable devices**

The solution must restrict communication of vulnerable devices and isolate them. This means that a vulnerable device should not be able to communicate with rest of the network and the Internet. This would potentially restrict the capabilities of an attacker that are discussed in Section 3.2.3. With this requirement, we intend to reduce the attack surface area. All device to device and device to Internet communication must be analyzed and controlled at the security gateway. Communication between devices in different network segments should not be allowed. The solution should ensure that all communication is controlled and monitored.

#### **R1: Enforce isolation profile from IoT Sentinel**

IoT Sentinel [52] defines an isolation profile for each device type. The isolation profile for a device consists of information that is used to segregate it. The solution should create an isolation environment which is able to separate devices into different network segments based on the isolation profile provided by IoT Sentinel.

#### **R2: Easy and fast (re)configuration of network**

Whenever a new device is introduced in a network, the reconfiguration process of the network to integrate the device should have a low overhead and no impact on other devices. An IoT device may have a specific profile allowing access to certain IP addresses after initial bootstrapping. Therefore, access control mechanism for a specific device or a group of devices should also be supported dynamically according to the information specified in the isolation profile.

#### **R3: Performance**

The usability of the system is a crucial factor to consider. The system should have a high usability, low-performance overhead, with minimum time over-

head on IoT Sentinel's existing processes. Timing and bandwidth analysis of the introduced solution should be evaluated.

### **U1: Deployment**

The device isolation solution is intended to be easily deployable in Off-the-shelf (OTS) routers or gateways. The choice of tools used for device isolation should be suitable for its deployment in OTS routers.

### **U2: Demo user interface**

The isolation approach should be depicted with a user interface (UI) that is useful for showing the different steps of device isolation. The UI is intended to be helpful for collecting data for evaluation and debugging purposes.

## Chapter 4

# Proposed Solution and Implementation

In this chapter, we describe the proposed solution and its implementation. The implementation work is an enhancement to IoT Sentinel to address the requirements discussed in Section 3.3. We first discuss the capabilities needed for the solution in Section 4.1.1. Sections 4.1.3 and 4.1.4 present the design and flow diagram of the solution. Section 4.2 discusses the implementation choices for the techniques and tools used in the solution. Section 4.3 describes the implementation details of the system.

## 4.1 Proposed Solution

### 4.1.1 Capabilities

The capabilities of the intended solution are as follows.

**Enforcement of isolation profile** As mentioned in requirement R1, the system must use the isolation profile obtained from IoT Sentinel. First, the isolation level and allowed IPs should be parsed from the isolation profile. Thereafter, the device must be placed into an appropriate network segment based on the isolation level which determines if the device is vulnerable or not.

**Construction of Network** The system must define a network where devices can be segregated into untrusted and trusted networks. As described in Section 2.3.4, the network should be divided into Strict, Restricted and Trusted isolated network segments.

**Control of Communication** The system should have the ability to control the communication between devices. The objective is to control the communication of devices according to their isolation level. One such capability is to prevent the communication of vulnerable devices to trusted devices and the Internet.

**Dynamic Management of hosts** The system should make sure that the devices (hosts) get the correct configuration for connectivity during initial device configuration and later device functioning. Initially, the device should have access to the Internet for device configuration. After the device is configured, based on its isolation profile, the network configuration for the device must be updated dynamically according to its network segment determined by isolation level.

## 4.1.2 System Components

The proposed solution consists of the following components. Figure 4.1 depicts the major components of the proposed system. The major components are isolation map, DHCP server, and firewall. User Interface is an additional component which is not responsible for core responsibilities of the solution but is useful for tracking of connected devices and debugging. The components are described below.

**Isolation Map** The isolation map is a list of device MAC addresses mapped to their isolation levels. The isolation map is placed locally at the security gateway to keep track of connected devices and their isolation levels. The isolation map is updated every time a new device is connected or an isolation profile is received from IoTSSP.

**DHCP Server** The DHCP Server provides IP addresses and network configuration settings to the client connected to the wireless AP. The network segmentation is defined in the DHCP server in the form of subnets. The management of different subnets and its components enables us to fulfill the capabilities of construction of network and dynamic management of communication of devices.

**Firewall** The firewall is used to filter communication between the devices or network segments. This enables us to fulfill the capability of the system to control communication. By setting appropriate rule we can allow or deny Internet connectivity to the network segments. Device specific rules for connectivity to specific addresses can also be configured using the firewall.

**User Interface** A user interface is a demo tool that is helpful to keep track of different steps a device is going through during the device identification and device isolation processes. The user interface is not a major component for actual isolation operation, but is beneficial for debugging the solution under development.

### 4.1.3 Design

Figure 4.1 describes the interaction between the components mentioned in Section 4.1.2 as well the network isolation design of the solution.

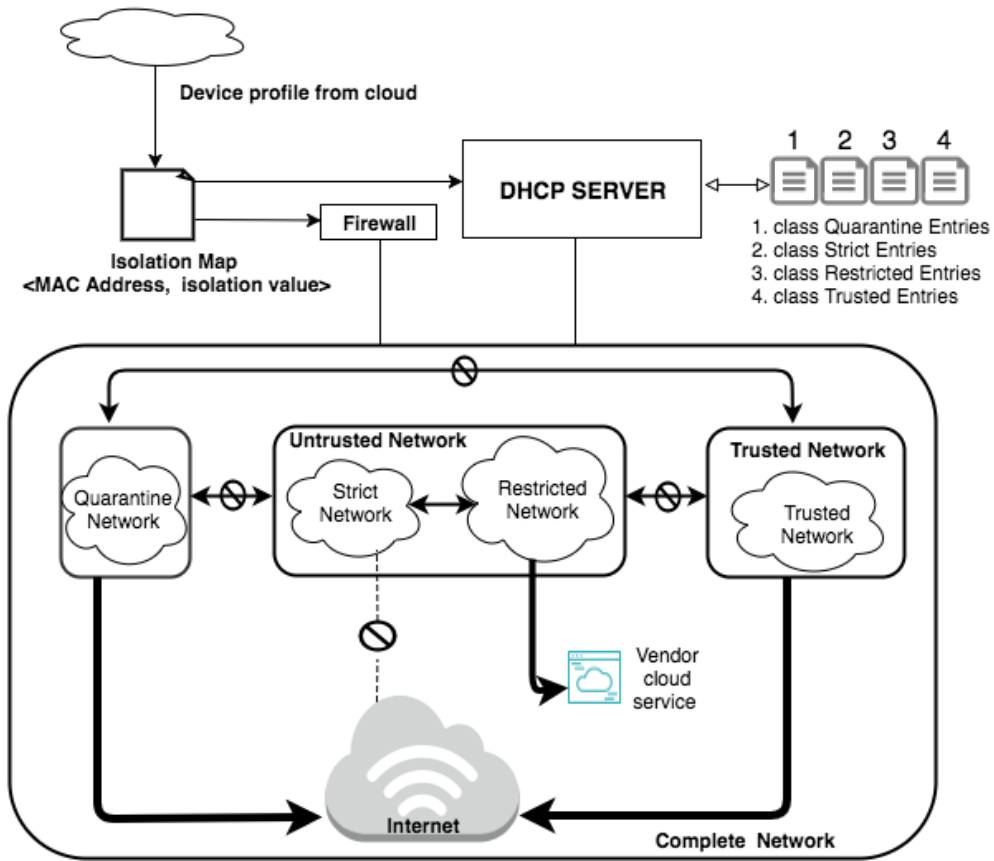


Figure 4.1: Network Isolation Design

The solution depends on IoT Sentinel device identification service in the cloud to provide the isolation profile, based on which the placement of the device in a segmented network is carried out. However, the network must be first divided into several segments which are isolated from each other.

In our solution, the network is divided into four subnets namely quarantine, strict, restricted, and trusted. The quarantine subnet is the network segment where devices are initially placed during device configuration process. The isolation level for devices in quarantine segment is set to -1 until the actual isolation level is received from the cloud. The strict, restricted, and trusted subnets are for the devices with isolation value 0, 1, and 2 respectively. The abilities of network segments are illustrated in the Table 4.1. For example, the restricted subnet can access only specific IP's specified in the isolation profile and cannot communicate with subnet other than strict and itself. The DHCP server is configured to divide the network into the four subnets described above. In order to keep the subnets isolated, a firewall must be configured according to the capabilities described in Table 4.1.

Table 4.1: Network Segments Capabilities

<b>Network Segment</b>	<b>Isolation Value</b>	<b>Internet Communication</b>	<b>Communication with other segments</b>
Quarantine	-1	Full	No communication with other segments
Strict	0	Blocked	Allowed with strict and restricted
Restricted	1	Specific IPs	Allowed with strict and restricted
Trusted	2	Full	Allowed with trusted

At the security gateway, whenever a new device is associated with the AP, the solution should place the device in the quarantine network until its isolation profile is obtained from IoTSSP. We need to put the device in quarantine network because IoT Sentinel takes time to fingerprint the device and obtain the isolation profile. When the isolation profile is obtained, the device is moved to the appropriate network segment according to its isolation level.

The solution utilizes the isolation map to keep track of the isolation level of the devices. When the device is in the quarantine network, the isolation map has the default isolation level as -1 for the device. When the actual isolation level is obtained from IoTSSP, the corresponding entry in the isolation map is updated. The solution keeps track of any changes in the isolation map and configures the DHCP server accordingly. The DHCP server is used

to grant IP address and lease time to a device based on its isolation level in the isolation map.

The firewall filters the communication across the different network segments defined. It also ensures the Internet connectivity required for certain network segments. The system also updates the firewall in order to apply any device-specific requirements in the isolation profile such as allowing communication with specific IPs.

#### 4.1.4 Flow Diagram

The flow diagram in Figure 4.2 presents the flow of control in the system.

Firstly, IoT Sentinel is started which also starts the wireless AP to allow IoT devices to connect to the home network. Thereafter, the solution starts the DHCP server with the configuration to create four required subnets namely quarantine, strict, restricted, and trusted. Then the default firewall rules are loaded which ensures the isolation of subnets as mentioned in Section 4.1.3.

Our solution then waits for detection of new MAC address. When a new MAC address is detected, we add an entry in the isolation map with isolation level -1 and then update the DHCP server to place the device in quarantine subnet for initial configuration. Meanwhile, IoT Sentinel extracts the device fingerprint and sends it to IoTSSP to obtain the isolation profile. Any errors during the fingerprinting process are reported in the dashboard on the UI. Note that, if the device has previously been configured successfully, it will already have an entry in the isolation map. In such a case, the DHCP server configuration file should ideally contain the appropriate entry for the device and it should automatically get placed in its previously assigned subnet.

The isolation level is then checked from isolation profile and the configuration files for DHCP server are updated and reloaded. DHCP configuration files are updated such that if the isolation level is 0, 1, or 2, the device is moved to strict, restricted, or trusted subnet respectively. Effectively, this means that when the lease for the device in quarantine subnet expires, the DHCP server will assign an IP address to a device in the new subnet based on the updated configuration. For example, if an isolation level for a device is 0 in isolation profile, after DHCP server update, it will get an IP address from strict subnet when its initial lease from quarantine subnet expires.

If the isolation level for a device is 1, it also consists of a set of allowed IPs that device can communicate. Thus, firewall rules are generated for the device to access certain allowed specific IPs.

In case of an error situation where the isolation profile is not received from IoTSSP, the DHCP configuration files are updated to remove the device from

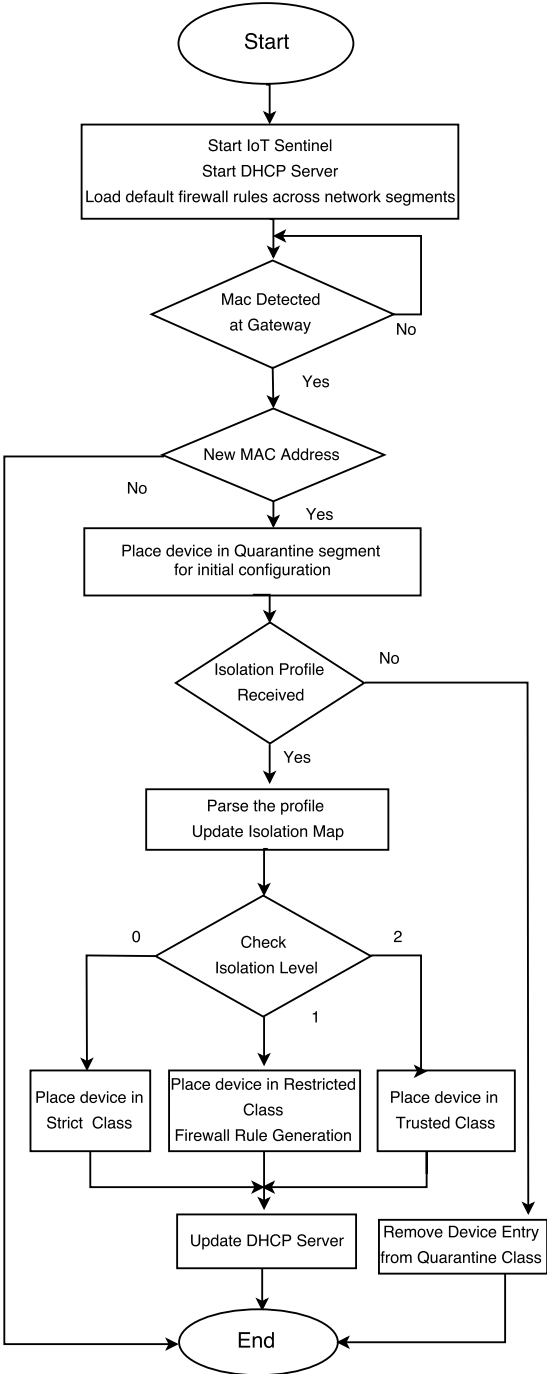


Figure 4.2: Flow Diagram

the quarantine subnet and the dashboard is updated with an error message on the UI. This will effectively disconnect the device from the network when its initial lease expires. This prevents any devices from persistently being in the quarantine subnet if their isolation profiles cannot be obtained in case of any error. This ensures if isolation profile is not received for a device, it will be removed out completely from the network.

## 4.2 Implementation Choices

In this section, we justify the implementation choices for techniques and tools made for the solution.

### 4.2.1 Device Isolation and network segmentation

The objective of device isolation is to build isolated network segments within the internal private network and between the external Internet world. We studied various techniques that are used to facilitate the network segmentation. We studied two different approaches to perform device isolation. The first choice is setting up different isolated subnets and customization of a DHCP server to provide IPs dynamically in different subnets. Subnetting a network is done at layer 3 (network layer) of the TCP/IP model. The subnets are isolated from each other by applying relevant firewall rules. The second choice for the solution is to use a dynamic Virtual Local Area Network (VLAN) [8]. Using dynamic VLAN, we can partition the network to create isolated segment at layer 2 (data link layer) of the TCP/IP model. The subdivision of the network into multiple VLANs is done by configuring the network equipment such as a switch. A wireless user is placed in a specific VLAN based on credentials supplied by the user based on the user group set in the RADIUS server.

We decided to use the approach of subnetting to implement the device isolation and network segmentation since it does not require any additional network equipment as in the case of VLAN. Subnetting is a standard practice of dividing a network into smaller segments. By creating subnets we can fulfill the solution's capability of construction and division of the network into different segments. The subnetting approach for device isolation with a DHCP server and a firewall is not dependent on any external Authentication, Authorization, and Accounting (AAA) server like RADIUS. All the configuration can be done is one host thus making administration easier. Moreover, free and open source implementation of the DHCP server and firewall are easily available to implement the subnetting based approach.

### 4.2.2 Components

Another important decision is to choose the DHCP server. DHCP server to use should be able to run on the Linux platform. We have different isolation levels for a device which correspond to network segments. DHCP server should be customizable in order to address the need for different network segments and device isolation levels. `isc-dhcp-server` [14] is open source software that can be configured to manage different classes of devices and define the topology of network using subnets. The isolation level of a device can be mapped to a class of device in the `isc-dhcp-server` which is easy to manage and configure. Therefore we use `isc-dhcp-server` as our choice of the DHCP server. This helps in fulfilling our solution’s capability of construction of the network and dynamic management of hosts.

We use `iptables` [17] for firewall in order to address the control of communication capability of our solution. It is also easy to obtain information about how to construct effective `iptables` rules. `iptables` is included by default in Linux distributions and is a well-known utility, thereby making it an obvious choice.

For scripting purposes we use Python [28] in order to integrate and configure different components (firewall, DHCP server, IoT Sentinel, etc.) to fulfill the capability of enforcing the isolation profile. Python is a simple general purpose programming language. Moreover, Python is already used in the IoT Sentinel project.

## 4.3 Implementation

For implementing the solution we first describe the actual IP configuration of the subnets which includes IP addresses and ranges. Then, we configure these subnets in the `isc-dhcp-server` DHCP server. Thereafter, we implement the `iptables` rules and extended the IoT Sentinel user interface. We also described the configuration of `hostapd` to support device-specific Pre-Shared Key (PSK).

### 4.3.1 Subnetting Approach

The DHCP server is configured such that the network uses the address 172.16.0.0/16. We can also use other private space depending on the size of the home IoT network. The main network configuration is presented in Table 4.2. We can accommodate a maximum total of  $2^{16} - 2$  (65534) hosts in the whole network.

Table 4.2: Network configuration for main network

<b>Network Parameter</b>	<b>Value</b>
<b>Network</b>	172.16.0.0/16
<b>Netmask</b>	255.255.0.0
<b>Broadcast</b>	172.16.255.255
<b>HostMin</b>	172.16.0.1
<b>HostMax</b>	172.16.255.254
<b>Hosts/Net</b>	65534

The main network is divided into four different subnets namely quarantine, strict, restricted and trusted. The subnet configuration is presented in Table 4.3. Since we have the total of four subnets, we borrowed two bits from the host part for subnetting. Thus, each subnet can hold up to 16382 devices. However, borrowing more bits can give more flexibility and leave space for the creation of additional subnets in the future, but less number of hosts in each subnet. Considering the number of hosts in a typical home network, the approach we used leads to a waste of host addresses in the quarantine subnet since the number of devices that will use quarantine subnet is lower compared to other subnets. For an optimized network setup, it is recommended to have a smaller quarantine subnet and bigger other subnets. Nevertheless, the current configuration is easy to understand and setup.

Table 4.3: Subnet configuration

<b>Network / Parameters</b>	<b>Subnet1 (Quarantine)</b>	<b>Subnet2 (Strict)</b>	<b>Subnet3 (Restricted)</b>	<b>Subnet4 (Trusted)</b>
<b>Network</b>	172.16.0.0/18	172.16.64.0/18	172.16.128.0/18	172.16.192.0/18
<b>Netmask</b>	255.255.192.0	255.255.192.0	255.255.192.0	255.255.192.0
<b>Broadcast</b>	172.16.63.255	172.16.127.255	172.16.191.255	172.16.255.255
<b>HostMin</b>	172.16.0.1	172.16.64.1	172.16.128.1	172.16.192.1
<b>HostMax</b>	172.16.63.254	172.16.127.254	172.16.191.254	172.16.255.254
<b>Hosts/Net</b>	16382	16382	16382	16382

### 4.3.2 DHCP Server Configuration

The main configuration file for `isc-dhcp-server` is `dhcpd.conf`. In the default configuration, the DHCP server listens on a network interface and provides IP address from a range defined in the configuration file. We can configure the `dhcpd.conf` file according to the network topology with different subnets declarations. Each subnet declaration is specified with its address, subnet mask, lease time, etc. Four other configuration files representing the class for subnets are also used by DHCP server.

A class is a grouping of client devices. The devices can be grouped in a class by matching their MAC hardware addresses. Figure 4.3 shows a sample configuration file for a *Trusted* class. Each subclass entry in the example configuration is a MAC address of a device that is placed in the *Trusted* class.

```
class "Trusted" {
    # match hardware address of the devices
    match hardware;
}

# Devices (MAC addresses) in the class
subclass "Trusted" 1:98:01:a7:ae:57:23;
subclass "Trusted" 1:c0:ee:fb:35:73:e2;
subclass "Trusted" 1:d0:27:00:02:eb:2e;
```

Figure 4.3: Classing of Devices

We define four classes *Quarantine*, *Strict*, *Restricted* and *Trusted*. Each class is defined for the corresponding subnet. Each class is defined in a separate files (e.g., `dhcpd.conf.classQ` for *Quarantine* class). These files are then included in the main `dhcpd.conf` using the `include` statement. This way the configuration is modular and we only need to modify the class files in order to move the devices between different subnets. For example, if a device needs to be moved from quarantine to strict subnet we need to delete the entry from *Quarantine* class file and add it to the *Strict* class file. The full DHCP configuration file is described in Appendix A.1

The subnet declaration also contains a `pool` section which defines the IP address range for the subnet and the `class` of devices which are allowed to be a part of the subnet. Figure 4.4 shows the configuration for a trusted subnet to allow devices from the *Trusted* class. Thus to place a device in a certain subnet, its entry must be present in the corresponding class. The

lease time for quarantine subnet is set for 40 seconds since we do not want the device to stay unnecessarily long in quarantine subnet after receiving the isolation profile. However, lowering the lease time further also increases the DHCP communication overhead in DHCP server.

```

subnet 172.16.192.0 netmask 255.255.192.0 {
    option broadcast-address 172.16.255.255;
    option routers 172.16.0.1;
    default-lease-time 600;
    max-lease-time 600;
    option domain-name "local";
    option domain-name-servers 8.8.8.8, 8.8.4.4;

    #Range of address for Trusted Subnet
    pool {
        allow members of "Trusted";
        range 172.16.192.1 172.16.255.254;
    }
}

```

Figure 4.4: Example declaration for trusted subnet

Consider an example scenario from the point of view of the device. For the first time when the device is connected to the network, the solution will put the device's MAC address in the *Quarantine* class file. This will make the DHCP server to give an IP address to the device in the quarantine subnet with a lease time of 40 seconds. The lease can get renewed until its entry exists in the *Quarantine* class file. When the isolation level is obtained (e.g. 2), the solution will remove the device's entry from the *Quarantine* class and create an entry in the *Trusted* class file (for isolation level 2). Thus when the lease for the device in quarantine subnet expires, the device will ask for a new lease. However, the DHCP server will not grant a new lease for the IP address in the quarantine subnet, since the device will not match the *Quarantine* class but the *Trusted* class. Instead, the DHCP server will offer an IP address in the trusted subnet with a new lease. Thus, the device is effectively moved from the quarantine subnet to the trusted subnet.

### 4.3.3 iptables

Firewalls are configured with a set of rules for a network interface to allow, deny or filter traffic. `iptables` can be set to filter the communication based

on different variables such as IP address, protocol, port, and interfaces. In the default mode `iptables` allows all traffic and does not have any filtering rules. On top of the default rule, we add rules to filter the traffic based on our requirements.

We define three different sets of rules in `iptables` for network isolation, enabling internet access, and device specific rules as follows.

#### 4.3.3.1 Rules for Isolating Subnets

The first set of rules ensure that subnets are not allowed to communicate and are isolated from each other according to the capabilities defined in Table 4.1. This set of rules are presented in Figure 4.5 and is applied initially when the wireless AP starts and a network is setup. This set of rules is always active and does not change.

The rules prevent forwarding of packets from one subnet to the other on the security gateway. For example, the following rules ensure that any device placed in the strict subnet (172.16.64.0/18) cannot communicate with a device in the trusted subnet (172.16.192.0/18) as well as the quarantine subnet (172.16.0.0/18).

```
iptables -I FORWARD -i br0
-s 172.16.64.0/18 -d 172.16.0.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.64.0/18 -d 172.16.192.0/18 -j DROP
```

Since this rules are applied at the security gateway where all the communication from the subnet passes through, the rules essentially drop the packets and prevent them from getting forwarded from one subnet (source) to the other (destination). Thus, the subnets are isolated.

#### 4.3.3.2 Rules for Enabling Internet Access

The second set of rules is constructed in order to give Internet access to the subnets according to Table 4.1. Since all the subnets are behind a NAT, we need to enable IP masquerading [16] to enable Internet access. Refer Section 2.4.7 for more details on IP masquerading. Figure 4.6 shows the list of rules that ensure that there is Internet access for quarantine, restricted and trusted subnet by allowing IP masquerading for the said subnets. We need to enable Internet access for the restricted subnets in order to allow the devices to access specific hosts on the Internet. Since there is no IP masquerading enabled for the strict subnet, it will not Internet access. These rules are also applied as soon as the wireless AP is set up.

```

# No communication from quarantine to other subnets
iptables -I FORWARD -i br0
-s 172.16.0.0/18 -d 172.16.64.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.0.0/18 -d 172.16.128.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.0.0/18 -d 172.16.192.0/18 -j DROP
# No communication from strict to other subnets than restricted
iptables -I FORWARD -i br0
-s 172.16.64.0/18 -d 172.16.0.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.64.0/18 -d 172.16.192.0/18 -j DROP
# No communication from restricted to other subnets than strict
iptables -I FORWARD -i br0
-s 172.16.128.0/18 -d 172.16.0.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.128.0/18 -d 172.16.192.0/18 -j DROP
# No communication from trusted to all other subnets
iptables -I FORWARD -i br0
-s 172.16.192.0/18 -d 172.16.0.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.192.0/18 -d 172.16.64.0/18 -j DROP
iptables -I FORWARD -i br0
-s 172.16.192.0/18 -d 172.16.128.0/18 -j DROP

```

Figure 4.5: Iptables rule to prevent communication across subnets

```

iptables -t nat -I POSTROUTING -o eth0
-s 172.16.0.0/18 -j MASQUERADE
iptables -t nat -I POSTROUTING -o eth0
-s 172.16.128.0/18 -j MASQUERADE
iptables -t nat -I POSTROUTING -o eth0
-s 172.16.192.0/18 -j MASQUERADE

```

Figure 4.6: Rules to enable Internet access for subnets

#### 4.3.3.3 Device Specific Rules

The third set of rules is applied dynamically for each device in the restricted subnet since devices in restricted subnet should be able to access only certain

IP addresses mentioned in the isolation profile. As soon as the isolation profile is received at the gateway with restricted isolation level (1), we parse the `allowed_ips` field and generate this set of rules and applied. By default, the restricted subnet should deny all outgoing traffic except to strict, while each dynamic rule generated after receiving isolation profile at security gateway allows traffic to a specific destination. For example, the below rule enables access to a specific host (`along.umeng.com`), strict subnet, restricted subnet, and denies access to all other addresses. We used MAC address of the device in the filtering rules rather than the IP address because the MAC address is constant whereas an IP address may change when the device reconnects to the network.

```
# Allow host
iptables -I FORWARD -i br0 -d along.umeng.com
-m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
# Allow strict subnet
iptables -I FORWARD -i br0 -d 172.16.64.0/18
-m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
# Allow restricted subnet
iptables -I FORWARD -i br0 -d 172.16.128.0/18
-m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
# Deny rest of the network
iptables -I FORWARD -i br0 -d *
-m mac --mac-source d0:27:00:02:eb:2e -j DROP
```

#### 4.3.3.4 Order of Evaluation of the rules

The order of evaluation of rules for any packet flowing through the AP is as follows.

1. Device Specific Rules
2. Rules for Isolating Subnets
3. Rules of enabling Internet for subnets

Note that if a packet satisfies any rule condition, the consecutive rule evaluation is skipped.

#### 4.3.4 Hostapd Configuration

IoT Sentinel uses `hostapd` to set up a WiFi access point. The wireless communication is set up in infrastructure mode which ensures that all device

communication passes through an access point. Thus, the infrastructure mode setup prevents direct device to device communication.

In IoT Sentinel, `hostapd` is configured to use a single pre-shared key for all devices. If a single pre-shared key is compromised, then any device can get access to the network. Attackers can use dictionary attacks on sniffed message integrity data from the 4-way handshake to find the passphrase. Therefore, we configured `hostapd` to use device-specific pre-shared keys for authentication. In this case, each device uses a different passphrase to connect to the AP. Thus, in case a key is compromised for a certain device it cannot be used by other devices to connect to the AP because each key is tied to a MAC address. This reduces the attack surface.

`hostapd` uses a configuration file `hostapd.conf` to set the parameters required for setting up an AP in infrastructure mode. A file containing a list of MAC address and corresponding passphrase pairs is created and specified in the `wpa_psk_file` parameter of `hostapd.conf` file. A sample `hostapd.conf` file is available in Appendix A.2.

### 4.3.5 User Interface

There are multiple processes involved in device identification and isolation. In the deployed solution, a user does not care about the processes involved. However, the processes are tracked for testing and debugging of the system as well as the collection of the data for evaluation purposes. We extended the IoT Sentinel user interface to show the additional steps done for device isolation after receiving the isolation profile. Figure 4.7 presents the sequence of events during the device identification process. As seen in the figure, a new device is encountered at the gateway, IoT Sentinel device identification process starts and the device is placed in the quarantine subnet. The red rectangular boxes highlight the events related to device isolation.

Figure 4.8 shows the sequence of events after the isolation profile is obtained. The red rectangular boxes highlight the applied dynamic `iptables` rules and the movement of the device from quarantine subnet to the restricted subnet. The IP address which is shown at top red rectangular box of 4.7 is IP address from restricted subnet updated after device isolation is performed (initially it is from quarantine subnet 172.16.0.10).

## Dashboard

Connected to Securebox

New Device 0 Actual state: Device with mac address d0:27:00:02:eb:2e

is allocated new ip address: 172.16.128.2

Device with MAC: d0:27:00:02:eb:2e placed in Quarantine Network (172.16.0.0/18) for configuration

Start to capture packets: 2017-09-06 11:24:47.231572

Capturing. 79 seconds left.

Capturing. 69 seconds left.

Capturing. 59 seconds left.

Capturing. 49 seconds left.

Capturing. 39 seconds left.

Capturing. 29 seconds left.

Capturing. 19 seconds left.

Capturing. 9 seconds left.

Capturing. 86399 seconds left.

IP being used from Quarantine subnetwork(172.16.0.0/18) for configuration:172.16.0.10

Capturing terminated: 2017-09-06 11:26:17.401031

Feature extraction completed: 2017-09-06 11:26:26.915700

Start to compress features: 2017-09-06 11:26:26.916278

Compression completed: 2017-09-06 11:26:27.407256

Start classification: 2017-09-06 11:26:27.407935

Figure 4.7: UI before isolation profile is obtained

```
Classification results received from IoTSSP cloud server
{
  "allowed_ips": [
    "8.8.8.8",
    "8.8.4.4",
    "eu-disp.coolkit.cc",
    "along.umeng.com"
  ],
  "id": 1,
  "isolation": 1,
  "name": "SmartSocket"
}
Device classified as SmartSocket with isolation level 1
Dynamic Rule from profile set to iptables
iptables -I FORWARD -d 8.8.8.8 -m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
Dynamic Rule from profile set to iptables
iptables -I FORWARD -d 8.8.4.4 -m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
Dynamic Rule from profile set to iptables
iptables -I FORWARD -d eu-disp.coolkit.cc -m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
Dynamic Rule from profile set to iptables
iptables -I FORWARD -d along.umeng.com -m mac --mac-source d0:27:00:02:eb:2e -j ACCEPT
Classification completed: 2017-09-06 11:26:29.112805
End to process the new device: 2017-09-06 11:26:30.202005
DHCP server updated for a device
Device moved from Quarantine (172.16.0.0/18) to Restricted Subnetwork (172.16.128.0/18)
Device with mac address d0:27:00:02:eb:2e is allocated new ip address: 172.16.128.2
```

Figure 4.8: UI after isolation profile is obtained

## Chapter 5

# Evaluation

This chapter presents the evaluation of the system. The system is evaluated for the requirements listed in section 3.3. Section 5.1 explain the evaluation of security requirement S1. Section 5.2 describe the functional requirements R1-R3 while the usability requirements are evaluated in Section 5.3. Refer to Section 3.3 for a detailed description of all the requirements.

### 5.1 Security

The security requirement states that the communication of a vulnerable device must be restricted in order to thwart the capabilities of an attacker. The requirement also states that device to device and device to Internet communication must be controlled by the AP. The wireless AP is set up in infrastructure mode which ensures that device to device communication happens only through the AP. Since all device communication passes through AP, it is controllable by `iptables` in the security gateway.

After implementing the solution, we observe that the network is isolated into different subnets as described in Table 4.1. Thus, any vulnerable device connected to the system will be placed into either the strict or the restricted subnet by our solution. The restriction is enforced with `iptables` as described in Section 4.3.3. In case of strict isolation level, the vulnerable device will not be able to connect to the Internet while in case of restricted it can communicate with only selected IPs or hosts. Moreover, a vulnerable device in the untrusted segment (strict or restricted) cannot communicate with the trusted segment. Considering the successful placement of a vulnerable device in the strict or restricted subnet, following attacker capabilities described in Section 3.2.3 have been mitigated.

- The attacker cannot gain control over a compromised device remotely

since the device has no external communication capabilities in the strict or restricted subnet. Moreover, the devices are behind a NAT which will restrict all inbound connections from the Internet.

- The attacker cannot use a compromised device to locate or mount attacks on trusted devices since there is no communication allowed between vulnerable and trusted devices as they are in isolated subnets and the gateway will not allow packets from being forwarded from untrusted segments to the trusted segment.
- A compromised device cannot open an attack path for attackers (e.g. via SSH tunnel or connection to cloud service) because communication to the Internet is controlled for the untrusted segment in the security gateway.
- A device cannot discover other vulnerable devices on the network or vice-versa because of the isolation of subnets and the gateway preventing the communication.

When a device initially gets connected to the network, it is placed in the quarantine subnet until it gets configured. During this phase, the device has access to the Internet for its configuration. However, the quarantine subnet is still isolated from the other subnets. The device will be moved to an appropriate subnet when its isolation profile is obtained. The device is disconnected if its isolation profile cannot be obtained to prevent malicious devices to stay in the network. Thus communication in the quarantine subnet is also controlled and isolated from trusted devices.

This isolation of devices in quarantine, strict, restricted, and trusted subnets along with a tight control of communication between the devices, subnets, and Internet, mitigates the attacker capabilities and fulfill the security requirements of our solution.

Although not a security requirement, the solution also has an added security benefit if device specific keys are used as explained in Section 4.3.4. The usage of device specific keys ensures that unauthorized access to the network is not feasible even if a PSK is compromised for a specific device.

## 5.2 Functionality

Requirement R1 is fulfilled when the isolation profile from IoT Sentinel is enforced. We constructed the network environment comprised of isolated subnets for different isolation levels. Our solution segregates the devices into

strict, restricted and trusted subnets. The isolation profile obtained from IoT Sentinel for a device is enforced by configuring the DHCP server to place the device into a correct subnet based on its isolation level. The implementation of the control flow explained in Section 4.1.4 fulfills the requirement of enforcing the isolation profile. We evaluated this requirement by connecting a devices with different isolation profiles and observed that the devices were first placed in the quarantine subnet and eventually moved into appropriate subnets.

Requirement R2 states that an easy and a fast reconfiguration of the network is required as well as the successful implementation of device specific rules derived from the isolation profile. The network is configured according to the subnetting approach by the DHCP server configuration described in Section 4.3.2. The device specific rules based on the isolation profile are implemented dynamically by adding additional `iptables` rules as described in Section 4.3.3.3. We observed that the solution does not impact the already connected devices when a new device is introduced into the network since every new connection starts a new instance of the configuration scripts. The whole process does not require manual intervention since the solution configures the DHCP server (eg. network configuration, IP addresses) and `iptables` automatically, thus making the process fast and easy. This fulfills the requirement R2.

The speed of reconfiguration of network to move device from quarantine to other subnets is dependent on initial lease time of quarantine subnet. The experimental analysis to measure speed of reconfiguration is explained below.

## 5.2.1 Performance

Requirement R3 states that the system should have a high usability, low performance overhead and a minimum time overhead on IoT Sentinel's existing processes. For the performance evaluation, we set up a test environment and measured the performance of the system in terms of time-delay and throughput analysis. The experiment setup and the subsequent evaluation is presented below.

### 5.2.1.1 Experimental Setup

We used an IoT Sentinel setup running on a Raspberry Pi [23]. We used two IoT devices namely a SmartSocket [24] device and a Netatmo [21] weather station. First, the devices were set up with their corresponding mobile applications [9, 20] using an Android phone to collect their reference fingerprints

for building the classifier. After the isolation profiles and the device classifiers were set up in IoTSSP, our solution was tested and evaluated.

The device configuration and subsequent placement in the isolated subnet were tracked in the UI. The timings were also recorded for the whole configuration process, IoT Sentinel operations, application of rules, subnet reallocation, etc. IoT Sentinel operations refers to total time taken for fingerprint capture, feature extraction, feature compression and obtaining classification results from IoTSSP. The time for applying rules refers to the total time taken for dynamic generation and application of rules based on information in the `allowed_ips` field in isolation profile. The subnet reallocation delay refers to the time taken to move device from quarantine subnet to the intended subnet after classification result is obtained. The experiment was repeated 15 times to determine the average  $\pm$  standard deviation.

For throughput analysis, we set up the security gateway with varying number of rules in each experiment iteration and recorded the throughput in Kbits/sec for communication from a test device to a remote server. We started with base number of default rules and took measurements for 1000, 2000, 5000 and 10000 rules at security gateway. The objective of this experiment is to observe the effect of number of `iptables` rules in the throughput.

### 5.2.1.2 Timing Analysis

Table 5.1 shows the breakdown of average timing data collected for each of the tested devices for different processes.

Table 5.1: Timing breakdown

<b>Process/Device</b>	<b>SmartSocket Mean <math>\pm</math> StDev</b>	<b>Netatmo Mean <math>\pm</math> StDev</b>
IoT Sentinel Operations	131.78 s $\pm$ 5.61 s	130.26 s $\pm$ 1.07 s
Apply Rules	1.37 s $\pm$ 2.3 s	0.71 s $\pm$ 0.61 s
Subnet Reallocation Delay	55.44 s $\pm$ 19.61 s	46.29 s $\pm$ 40.47 s
Total time for configuration	187.22 s $\pm$ 16.60 s	176.63 s $\pm$ 41.36 s

For the SmartSocket device, the IoT Sentinel operations took 131.78 seconds. The processing of isolation profile to generate and apply the dynamic rules took 1.37 seconds. Moving the device from the quarantine subnet to the intended subnet took 55.44 seconds. The total time taken is 187.22 seconds

on average. Similarly, for Netatmo weather station, the IoT Sentinel operations took 130.26 seconds on average. It took 0.71 seconds to apply rules. Moving the device from quarantine subnet to the intended subnet took 46.29 seconds. The total time taken was 176.63 seconds on average.

Based on the above data, we calculated the time overhead incurred with our approach in Table 5.2. It is seen that the time overhead is +42.07 % and +35.6 % over IoT Sentinel device identification for SmartSocket and Netatmo devices respectively.

Table 5.2: Timing Overhead Comparison

	(A)	(B)	Time Dif- ference	% Overhead
SmartSocket	131.78 s	187.22 s	+55.44 s	+42.07%
Netatmo	130.26 s	176.63 s	+46.37 s	+35.60%

(A) IoT Sentinel Device Identification

(B) IoT Sentinel Device Identification + Isolation

A significant factor that contributes to the time overhead is the movement of the device from quarantine subnet to the intended subnet because of the lease duration of 40 seconds. The device will continue to stay in the quarantine subnet until its lease expires. The DHCP server has no control over the lease until the lease duration is over, which means that the device does not get moved to a new subnet unless the device asks for a new lease. In some cases, in order to save power, devices do not ask for a new lease even after it expires and stays dormant for some time or until a manual interaction occurs. This adds to the time delay.

The lease duration for the quarantine subnet can be lowered to minimize the delay so that existing leases expire sooner. However, if lease duration is set for a short time, it increases the DHCP communication which might add a high load on the DHCP server in case of more devices. During a single device configuration process, it is likely that there will be only one device in quarantine subnet. So, choosing shorter lease time seems more appropriate to reduce the delay. However, the shorter lease will create frequent renewals of a lease, which add noise to the fingerprint of an IoT device which might reduce the accuracy of device identification. Therefore, the lease time for quarantine subnet is set neither too long nor too short. Moreover, it should also be noted that this delay is encountered only during the first instance the device is connected to the network. Once the device is configured, the

Table 5.3: Impact of Number of Rules in Throughput

Number of Rules	Throughput in Kbits/sec
0	16978.509
1000	16516.036
2000	14060.869
5000	9480.578
10000	5852.48

subsequent connections will not be tracked for performing IoT Sentinel device identification and our isolation solution. DHCP server places the device into its preconfigured intended subnet instead of the quarantine subnet.

### 5.2.1.3 Throughput Analysis

The impact of a high number of `iptables` rules in our solution is another metric that we evaluated. We added dummy rules to `iptables` for random source MAC addresses similar to the device-specific rules mentioned in Section 4.3.3.3. We used a Linux host allocated to the trusted subnet which transferred data to a remote server with varying number of rules in the security gateway. Table 5.3 presents the throughput collected for varying number of rules.

Figure 5.1 depicts the effect of the number of rules to the throughput of the system. With the increase in the number of rules, we observe that the throughput of the system decreases linearly. We see that the throughput decreases by 2.72 percent when there are 1000 rules as compared to the default number of rules. The rate of decrease in throughput is lower until 1000 rules compared to other higher intervals of the number of rules. In a typical home setup with limited devices, the number of rules is not expected to rise drastically. The throughput decreases significantly with the higher number of rules because each packet goes through all the rules sequentially in `iptables` chain.

Nevertheless, in a typical home network scenario, the system throughput will not get affected significantly. Let us consider a home network scenario with 20 IoT devices in the restricted subnet, each with 10 device-specific rules to allow communication with specific IP addresses. The default setup of our solution has 3 rules for IP masquerading and 10 rules for isolating subnets as explained in Section 4.3.3. Moreover, the devices in trusted and

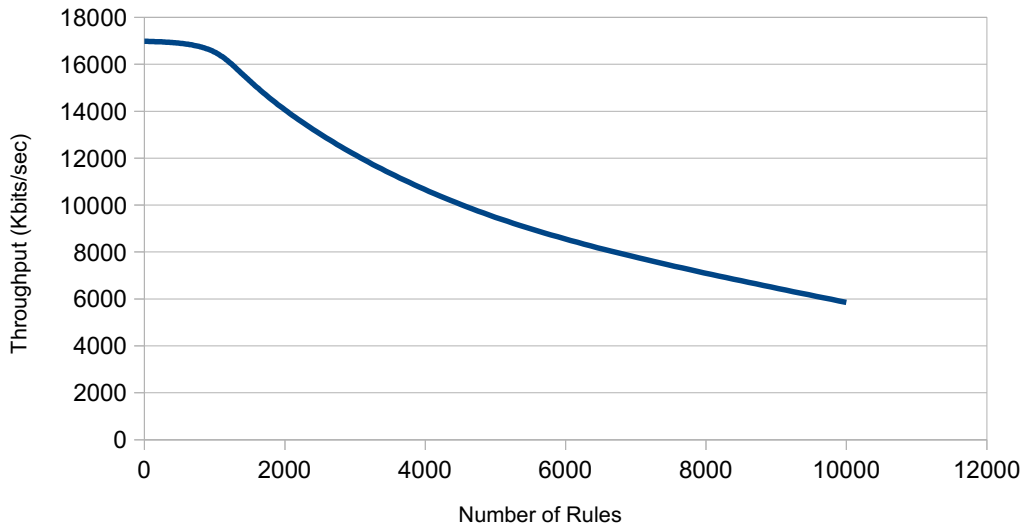


Figure 5.1: System throughput vs. number of `iptables` rules

strict subnets do not require any device specific rules ideally. Thus the total number of rules in `iptables` will be 213 ( $3 + 10 + 20 * 10$ ). As seen in our experiments, this number of rules does not affect the system throughput significantly.

## 5.3 Usability

There are two usability requirements U1 and U2 related to deployment and demo user interface respectively which are described in subsequent sections.

### 5.3.1 Deployment

Requirement U1 states that the solution should be easily deployable in OTS routers. The major components used in our solution are, a DHCP server and a firewall. OTS routers typically have a DHCP server and a firewall already present. The router firmware can be modified to include the functionality of our solution and will not require significant changes. Moreover, the primary motivation for this work is to create an SDN free approach for device isolation. Our solution does not use any SDN components and hence it is possible to deploy on traditional routers.

Our solution can be easily deployed on any Linux based system because we use `isc-dhcp-server` and `iptables` which are open source and well

supported on Linux. `iptables` is typically the default firewall in most of the Linux distributions. This makes the deployment easy.

### 5.3.2 Demo User Interface

The usability requirement U2 states that the isolation approach should be aided with a demo UI that is useful for showing the different steps of device isolation approach. We modified the existing IoT Sentinel demo UI and presented in Section 4.3.5. The extended demo UI correctly shows the different steps the device goes through during the device identification and isolation process. Figures 4.7 and 4.8 show a sample UI. This addresses the usability requirement of having a demo user interface.

## Chapter 6

# Related Work

IoT is a new and an active area of research trying to address problems across different domains such as security, networking, applications, etc. Security research in IoT is still in the infant stage [64]. Most of the prior works have been targeted to identify potential threats and adapt the existing security infrastructure to the IoT environment. Our work considers securing trusted IoT devices in the network from other vulnerable devices by profiling IoT devices and defining their capabilities at the network level. The network is managed in a way that there is a clear line of separation between the trusted and untrusted devices. We found a limited number of close research prototypes ([38, 59]) aiming at controlling the vulnerable IoT devices. In this section, we discuss related works carried out in the domain of IoT security.

In [58], Noor et al. have given a clear picture of the threats to a wireless network and discussed the effects of latest developments such as BYOD and device tethering on the security of Wi-Fi networks. The authors also present a case study of cybercrime incidents and level of Wi-Fi security in Malaysia. The results of the case study show that there is a steady increase in a number of significant threats to the wireless networks. The authors have also presented countermeasures and mitigation strategies that help to protect wireless APs and networks such as choosing strong authentication protocol, disabling SSID broadcast, enabling the firewall, intrusion prevention etc. The paper was helpful especially in getting an overview of the state of Wi-Fi network security.

In [34], Atamli et al. have discussed a threat-based security analysis of IoT. The authors have discussed relevant use cases of IoT such as power management, smart car, and smart health care system. The authors have identified the sources of threats such as malicious user, bad manufacturer and external adversary. They also identified different classes of attacks vectors in the use cases such as device tampering, information disclosure, spoofing,

elevation of privilege, signal injection etc. Thereafter they have discussed impacts of such attacks and deduced the security and privacy properties of the IoT systems. Such properties include the need for access control mechanism in IoT device, protecting data exchanged between a user and an IoT device etc. The authors have also conducted a preliminary risk assessment for those systems with respect to different attack vectors.

In [59], the authors have proposed a network level protection to block communication of vulnerable devices that utilizes a database of firewall rules maintained in the cloud. Three different IoT devices namely Nest Smoke Alarm, Hue Light-Bulbs, and WeMo Motion Switch were studied to identify security issues related to them. For example, Nest Smoke Alarm transfers 20Kb of data to its log server daily, which is suspicious. The solution is then tested by extracting firewall rules from the cloud and applying them to a device. The test was successful to block the device's communication to the log server without affecting other functions of the device. The approach used in the paper to control the communication of vulnerable device using the firewall is similar to our work.

Paper [38] describes an approach for limiting malware propagation in IoT network with the number of unpatched resource constrained IoT devices. Instead of directly patching individual malware propagating IoT device, their approach uses patching intermediate nodes such as AP, Base Station (BS) or gateway with which IoT devices communicate. The approach is traffic-aware which means that the intermediate node which can contact a high number of IoT nodes is patched first to avoid the catastrophic spread of malware. The authors conclude that the approach can assist detection of new attacks and patching strategies.

In [63], Matt Smith describes tackling security and privacy issues in a smart home environment from the point of view of network topology and infrastructure. The author has also discussed technologies used in IoT environments such as common wireless standards including IEEE 802.15.4, 6LoWPAN and IEEE 802.11. The paper focuses on the improvement of security in the home network to protect the network by incorporating security techniques practiced in enterprise level systems. This includes commonly used approaches such as device authentication using RADIUS [11] or Kerberos [15] and setting up DMZs. These approaches can be imported into the smart home scenario to aid network segmentation. The author has concluded that using enterprise level techniques may be too complex for simple home network maintenance. This work suggests a theoretical approach for securing the home network by borrowing techniques used in enterprise level networks which is hard and expensive to deploy. Though the motivation of work is similar to our's, our solution is practically implemented, more lightweight

and easily deployable.

F-Secure SENSE [10] aims to provide security in home networks and protect connected devices from online threats. F-Secure SENSE consists of a securely configured router, mobile application, and a cloud service. F-Secure SENSE monitors the traffic of all connected devices to the router in real time and provides different protection features such as blocking traffic from a compromised device, blocking malicious connection etc. The mobile application allows different security configurations for the router. F-Secure SENSE analyzes the network traffic of any device connected to the router with automatic protection against phishing, intrusive tracking and attacks. Upcoming features include parental control, guest wifi, IoT device reputation-based protection, etc. F-Secure provides automatic updates for SENSE router. Though the technical implementation details of the product are unavailable, this work is similar to our work in a way that both try to control the traffic of vulnerable devices. However, F-Secure SENSE is a commercial product already available in the market that addresses different areas of security. Dojo [6] is also one another product already in the market that is aimed at securing home against vulnerable devices by providing control over the devices in the network. It analyzes the traffic of home network and enforces the security.

Lancope's Stealthwatch System [22] uses a software-defined approach for network segmentation. The system makes use of the dynamic policies based on contextual information around a node's role. Such approach is referred to as active segmentation. The solution is feasible for enterprise network's security. The approach requires a network administration skills for designing and modeling of segmentation policies which is suitable for nodes based on their role.

In their master's thesis [27], Stelma investigated the network security of the home networks. Similar to our work, the author has proposed a solution to enhance the security of the home network using subnetting and a DHCP server for isolating devices at the designed gateway. They created isolated subnets dedicated to unregistered devices within a single IP subnet defined for registered devices. The discussions that we made about the effect of setting up a longer/shorter DHCP lease time in our solution are similar to this work. In [66], authors have presented the significance and usage of VLAN technology in their library network. The logical segregation of internal subnet of a library is achieved from other distrusted network segments. Cisco switches are configured to divide main library network into four VLANs with different authority levels to ensure data security.

In [48], Kevin et al. have presented the Improved-Cross Layer Scheduling Model to ensure an optimized resource management in home networks at a gateway. In their experiment, they used a subnetting technique to divide the

intelligent home network across six subnets for Bluetooth, Zigbee, Ultra Wide Band, Smart Grid, Body Area and WiFi. Subnets and devices in the network are classified and prioritized. Authors have simulated a testbed to compare Cross Layer Switching (CLS) and Dynamic Load Balancing (DLB) and evaluate the network. The simulation results show a good network throughout and minimal delay. The paper is more focused on evaluation network performance among prioritized subnets.

## Chapter 7

# Conclusions

In this chapter we summarize our contributions, describe areas of improvements for future work and draw conclusions.

### 7.1 Summary of Contributions

The contributions of this thesis are summarized below:

- **A technique for automatically enforcing isolation profiles from IoT Sentinel using commonly available network primitives**  
Our solution integrates with IoT Sentinel and automatically enforces the isolation profiles which are used to perform device isolation. IoT Sentinel uses SDN based approach to utilize isolation profiles to perform device isolation. We proposed and implemented device isolation using standard network functionality available on off-the-shelf APs. We present a device isolation technique, which addresses the security requirement of restricting communication of vulnerable devices. The isolation technique addresses the communication capabilities for devices with different isolation profiles. The isolation technique is based on dividing the network provided by the AP into isolated network subnets using subnetting approach and controlling communication of subnets using a firewall. A DHCP server is customized to place the devices dynamically into required network segments. We make use of generally available tools namely `isc-dhcp-server` for DHCP customization and `iptables` for a firewall.
- **Evaluation of the implemented device isolation approach.**  
The device isolation solution is further evaluated to measure the delay overhead incurred on the IoT Sentinel setup for device identification.

The timing delay of +42.07% and +35.6% over IoT Sentinel device identification was observed for two test IoT devices (Smartsocket and Netatmo) while performing isolation. Based on the evaluation, the overhead incurred is found to be acceptable since the effect of overhead is only during the first time a device is introduced. We also evaluated the effects of the number of firewall rules on the system throughput and found that it does not significantly affect the throughput in a home network scenario. We observed that having 1000 `iptables` rules decreased the throughput by 2.72%. We also discussed the deployability of our solution.

## 7.2 Future Work

The solution can be enhanced further. The subnetting process used in the solution provides equal numbers of hosts in all subnets. The placement of devices in the quarantine subnet is temporary and devices will not be present in the quarantine subnet forever. Therefore, minimizing the size of the quarantine subnet can lead to a more flexible design and have room for more devices in other subnets.

In the performance evaluation, we observed that a longer lease time for devices in quarantine subnet impact adversely on the total time taken for device configuration. The overhead is lower if the lease time is reduced so that devices are moved to their intended subnets quickly. However, this comes at the cost of high load on the DHCP server. Finding the optimal lease time is an area for improving the performance of the solution.

In our solution, the DHCP server reads the updated configuration files from the disk whenever it needs to move the devices between subnets. However, loading files from the disk is a performance bottleneck. Finding techniques for using a memory mapped configuration or an alternative DHCP server implementation which uses better techniques is a candidate for future work.

Our evaluation work is done with a limited set of devices. Since different IoT devices have different ways of connecting to network the amount of overhead differs from device to device. Thus, evaluating our system with more types of devices will be useful to find out the reliability of the system and identify areas for improvement.

The solution is built and tested on Raspberry Pi 3 device. A direction for future work is to integrate the solution in OTS routers to improve the deployability of the solution.

## 7.3 Conclusion

IoT Smart home networks pose a range of security and privacy challenges. Since the number of IoT devices deployed is growing, maintaining the security of IoT networks is crucial. Protecting our network from the vulnerable devices is important. It is always recommended to install proper software updates and patches for all devices in the network. But a lack of updates of patches leave some devices vulnerable in the network. Properly identifying vulnerable devices and restricting them from accessing other valuable and trusted device is necessary. Our work proposes to segregate such vulnerable devices into different isolated network segments such that secure devices stay protected. Essentially our solution protects the trusted devices in the network from vulnerable devices.

We successfully designed and implemented our solution and solved the problem of device isolation with an acceptable performance overhead. Isolated network segments help to define boundaries for devices with varying security and communication needs. It helps to enforce security in IoT environments when a device is identified correctly by IoT Sentinel. Our solution is thus suitable to address the need for identifying the vulnerable devices and isolate them from the trusted devices.

# Bibliography

- [1] All your IoT devices are doomed — ZDNet. <http://www.zdnet.com/article/all-your-iot-devices-are-doomed/>. Accessed: 12.12.2017.
- [2] autobahn 17.10.1 : Python Package Index. <https://pypi.python.org/pypi/autobahn>. Accessed: 17.07.2017.
- [3] Bash Shell Scripting - Wikibooks. [https://en.wikibooks.org/wiki/Bash\\_Shell\\_Scripting#What\\_is\\_Bash.3](https://en.wikibooks.org/wiki/Bash_Shell_Scripting#What_is_Bash.3). Accessed: 12.05.2017.
- [4] BroadLink Official Website - e-control. <http://www.ibroadlink.com/app/>. Accessed: 14.05.2017.
- [5] Dnsmasq - network services for small networks. <http://www.thekelleys.org.uk/dnsmasq/doc.html>. Accessed: 14.05.2017.
- [6] Dojo FAQ. <https://dojo.bullguard.com/faq/>. Accessed: 02.01.2018.
- [7] Download Raspbian for Raspberry Pi. <https://www.raspberrypi.org/downloads/raspbian/>. Accessed: : 02.12.2017.
- [8] Dynamic VLAN Assignment with RADIUS Server and Wireless LAN Controller Configuration Example - Cisco. <https://www.cisco.com/c/en/us/support/docs/wireless-mobility/wireless-vlan/71683-dynamicvlan-config.html#switch>. Accessed: 12.12.2017.
- [9] eWeLink - Android Apps on Google Play. <https://play.google.com/store/apps/details?id=com.coolkit&hl=en>. Accessed: 12.16.2017.
- [10] F-Secure SENSE — Secure router and app. : [https://www.f-secure.com/en\\_US/web/home\\_us/sense](https://www.f-secure.com/en_US/web/home_us/sense). Accessed: 18.02.2017.
- [11] Freeradius. <https://freeradius.org/>. Accessed: 01.05.2017.

- [12] Hackers Use A Refrigerator To Attack Businesses - Business Insider. <http://www.businessinsider.com/hackers-use-a-refridgerator-to-attack-businesses-2014-1?r=US&IR=T&IR=T>. Accessed: 14.06.2017.
- [13] Home Network Security — US-CERT. <https://www.us-cert.gov/Home-Network-Security#III-B>. Accessed: 01.06.2017.
- [14] ISCs open source DHCP software system Internet Systems Consortium. <https://www.isc.org/downloads/dhcp/>. Accessed: 14.05.2017.
- [15] Kerberos: The network authentication protocol. <https://web.mit.edu/kerberos/>. Accessed: 01.05.2017.
- [16] Linux IP Masquerade HOWTO. <http://www.tldp.org/HOWTO/IP-Masquerade-HOWTO/ipmasq-background2.1.html>. Accessed: 21.12.2017.
- [17] Man Page of IPTABLES. <http://ipset.netfilter.org/iptables.man.html>. Accessed: 12.08.2017.
- [18] MINIBIAN raspberry pi MINIBIAN: MINImal raspBIAN image for Raspberry Pi. <https://minibianpi.wordpress.com/>. Accessed: 02.12.2017.
- [19] Nest's Hub Shutdown Proves You're Crazy to Buy Into the Internet of Things. <https://www.wired.com/2016/04/nests-hub-shutdown-proves-youre-crazy-buy-internet-things/>. Accessed: 12.12.2017.
- [20] Netatmo Weather - Android Apps on Google Play. <https://play.google.com/store/apps/details?id=com.netatmo.netatmo&hl=en>. Accessed: 12.16.2017.
- [21] Netatmo Weather — Weather Station - Rain and Wind Gauge. <https://www.netatmo.com/en-GB/product/weather/weatherstation>. Accessed: 12.16.2017.
- [22] Network Segmentation Solution Brief. <https://www.lancope.com/resources/solution-briefs/network-segmentation-solution-brief>. Accessed: 25.5.2017.
- [23] Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. <https://www.raspberrypi.org/>. Accessed: 12.05.2017.
- [24] S20 Smart Socket - ITEAD Wiki. [https://www.itead.cc/wiki/S20\\_Smart\\_Socket](https://www.itead.cc/wiki/S20_Smart_Socket). Accessed: 12.16.2017.

- [25] Set up your Weather Station. <https://www.netatmo.com/en-GB/start>. Accessed: 14.05.2017.
- [26] Specifications — Bluetooth Technology Website. <https://www.bluetooth.com/specifications>. Accessed: 1.12.2017.
- [27] Stelma Jaap, Master's Thesis, Securing the Home Network . <https://pure.tue.nl/ws/files/47037062/799535-1.pdf>. Accessed: 12.12.2017.
- [28] Welcome to Python.org. <https://www.python.org/>. Accessed: 14.05.2017.
- [29] What is Zigbee? — Zigbee Alliance. <http://www.zigbee.org/what-is-zigbee/>. Accessed: 01.12.2017.
- [30] Wireless LAN Networking (White Paper). <http://support.usr.com/download/whitepapers/wireless-wp.pdf>. Accessed: 21.01.2018.
- [31] Z-Wave — Z-Wave Smart Home Products. <http://www.z-wave.com/about>. Accessed: 01.12.2017.
- [32] Floodlight OpenFlow Controller -Project Floodlight, 2017. [www.projectfloodlight.org/floodlight/](http://www.projectfloodlight.org/floodlight/). Accessed: 12.12.2017.
- [33] Pinterest, 2017. <https://fi.pinterest.com/pin/332422016217852229/>. Accessed: 25.06.2017.
- [34] ATAMLI, A. W., AND MARTIN, A. Threat-Based Security Analysis for the Internet of Things. In *2014 International Workshop on Secure Internet of Things* (Sept 2014), pp. 35–43.
- [35] BARCENA, M. B., AND WUEEST, C. Insecurity in the Internet of Things, 2015. [https://www.symantec.com/content/en/us/enterprise/iot/b-insecurity-in-the-internet-of-things\\_21349619.pdf](https://www.symantec.com/content/en/us/enterprise/iot/b-insecurity-in-the-internet-of-things_21349619.pdf). Accessed: 12.12.2017.
- [36] BREIMAN, L. Random forests. *Machine Learning* 45, 1 (Oct 2001), 5–32.
- [37] CHALLINOR, S. An Introduction to IP Networks. *BT Technology Journal* 18, 3 (Jul 2000), 15–22.
- [38] CHENG, S. M., CHEN, P. Y., LIN, C. C., AND HSIAO, H. C. Traffic-Aware Patching for Cyber Security in Mobile IoT. *IEEE Communications Magazine* 55, 7 (2017), 29–35.

- [39] CROW, B. P., WIDJAJA, I., KIM, J. G., AND SAKAI, P. T. IEEE 802.11 wireless local area networks. *IEEE Communications magazine* 35, 9 (1997), 116–126.
- [40] DAMERAU, F. J. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7, 3 (Mar. 1964), 171–176.
- [41] DARLENE, S. Of 10 IoT-connected home security systems tested, 100% are full of security FAIL — Computerworld. <http://www.computerworld.com/article/2881942/cybercrime-hacking/of-10-iot-connected-home-security-systems-tested-100-are-full-of-security-fail.html>. Accessed: 27.04.2017.
- [42] DROMS, R. Dynamic Host Control Protocol RFC. <https://www.ietf.org/rfc/rfc2131.txt>, 1997. Accessed: 14.05.2017.
- [43] GIL, P. Internet of things by the numbers: What new surveys found. <https://www.forbes.com/sites/gilpress/2016/09/02/internet-of-things-by-the-numbers-what-new-surveys-found/#2abf4f1316a0>. Accessed: : 06.07.2017.
- [44] GOODIN, D. 9 baby monitors wide open to hacks that expose users most private moments. <https://arstechnica.com/security/2015/09/9-baby-monitors-wide-open-to-hacks-that-expose-users-most-private-moments/>. Accessed: 27.04.2017.
- [45] GREENBERG, A. Hackers remotely kill a jeep on the highway with me in it. <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway>. Accessed: 27.04.2017.
- [46] HARTLEY-PARKINSON, R. Weeping Angels from MI5 and CIA have been spying on us through our Samsung Smart TVs — Metro News. <http://metro.co.uk/2017/03/09/mi5-and-cia-have-been-spying-on-us-through-our-tvs-6497867/>. Accessed: 27.04.2017.
- [47] JUNIPERRESEARCH. The number of IoT (Internet of Things) connected devices will number 38.5 billion in 2020. <http://www.businessinsider.de/how-the-internet-of-things-market-will-grow-2014-10>. Accessed: 27.04.2017.
- [48] KEVIN, K. M., KOGEDA, O. P., AND LALL, M. An Improved-Cross Layer Scheduling model for intelligent home networks. In *2016 IST-Africa Week Conference* (May 2016), pp. 1–9.

- [49] KEVIN ASHTON. That ‘Internet of Things’ Thing, 2009. <http://www.rfidjournal.com/articles/view?4986>. Accessed: 18.04.2017.
- [50] MALINEN, J. hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator, 2004. <https://w1.fi/hostapd/>. Accessed: 16.06.2017.
- [51] MAVRIDIS, I., ANDROULAKIS, A.-I., HALKIAS, A., AND MYLONAS, P. Real-life paradigms of wireless network security attacks. In *Informatics (PCI), 2011 15th Panhellenic Conference on* (2011), IEEE, pp. 112–116.
- [52] MIETTINEN, M., MARCHAL, S., HAFEEZ, I., ASOKAN, N., SADEGHI, A. R., AND TARKOMA, S. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (June 2017), pp. 2177–2184.
- [53] MIETTINEN, M., MARCHAL, S., HAFEEZ, I., FRASSETTO, T., ASOKAN, N., SADEGHI, A. R., AND TARKOMA, S. IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (June 2017), pp. 2511–2514.
- [54] MOGUL, J., AND POSTEL, J. RFC 950 Internet Standard Subnetting Procedure. <https://tools.ietf.org/rfc/rfc950.txt>, 1985. Accessed: 12.05.2017.
- [55] MYAGMAR, S., LEE, A. J., AND YURCIK, W. Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (SREIS)* (2005), vol. 2005, pp. 1–8.
- [56] NDIBANJE, B., LEE, H.-J., AND LEE, S.-G. Security analysis and improvements of authentication and access control in the Internet of Things. *Sensors* 14, 8 (2014), 14786–14805.
- [57] NICKY WOOLF. DDoS attack that disrupted internet was largest of its kind in history, experts say — Technology — The Guardian. <https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>. Accessed: 13.05.2017.
- [58] NOOR, M. M., AND HASSAN, W. H. Wireless networks: developments, threats and countermeasures. *International Journal of Digital*

- Information and Wireless Communications (IJDIWC)* 3, 1 (2013), 125–140.
- [59] NOTRA, S., SIDDIQI, M., GHARAKHEILI, H. H., SIVARAMAN, V., AND BORELI, R. An experimental study of security and privacy risks with emerging household appliances. In *2014 IEEE Conference on Communications and Network Security* (Oct 2014), pp. 79–84.
- [60] PETER, W. *Learning Internet of Things*. Packet Publishing Ltd, 2015.
- [61] SAMUEL, S. S. I. A review of connectivity challenges in IoT-smart home. In *Big Data and Smart City (ICBDSC), 2016 3rd MEC International Conference on* (2016), IEEE, pp. 1–4.
- [62] SENRIO. 400,000 Publicly Available IoT Devices Vulnerable to Single Flaw - Senrio. <http://blog.senr.io/blog/400000-publicly-available-iot-devices-vulnerable-to-single-flaw>. Accessed: 27.04.2017.
- [63] SMITH, M. A Network View of the Internet of Things for the Smart Home Environment, 2015. : <https://ora.ox.ac.uk/objects/uuid:8a310633-5b8a-4bc6-a841-b3006180b02e>. Accessed: 01.06.2017.
- [64] SUO, H., WAN, J., ZOU, C., AND LIU, J. Security in the Internet of Things: A Review. In *2012 International Conference on Computer Science and Electronics Engineering* (March 2012), vol. 3, pp. 648–651.
- [65] WEISER, M. The computer for the 21st century. *Scientific american* 265, 3 (1991), 94–104.
- [66] ZHANG, Y., LIU, H., AND REN, F. Applied Study of Layer 3 Switching Configuration Based on VLAN among Colleges' Library Network Systems. In *2011 International Conference on Control, Automation and Systems Engineering (CASE)* (July 2011), pp. 1–4.
- [67] ZHANG, Z. K., CHO, M. C. Y., WANG, C. W., HSU, C. W., CHEN, C. K., AND SHIEH, S. IoT Security: Ongoing Challenges and Research Opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications* (Nov 2014), pp. 230–234.

# Appendix A

## Configuration Files

### A.1 DHCP Configuration File

```
# Include configuration files consisting of list of devices
```

```
include "/etc/dhcp/dhcpd.conf.classS";  
include "/etc/dhcp/dhcpd.conf.classR";  
include "/etc/dhcp/dhcpd.conf.classT";  
include "/etc/dhcp/dhcpd.conf.classQ";
```

```
ddns-update-style none;  
default-lease-time 600;  
max-lease-time 7200;  
authoritative;  
log-facility local7;
```

```
# Network Declaration
```

```
shared-network my_network{
```

```
#Quarantine Subnet Declaration
```

```
subnet 172.16.0.0 netmask 255.255.192.0 {  
option broadcast-address 172.16.63.255;  
option routers 172.16.0.1;  
default-lease-time 40;  
max-lease-time 40;  
option domain-name "local";  
option domain-name-servers 8.8.8.8, 8.8.4.4;
```

```
#Specifying range of address for Quarantine Subnet
pool {
allow members of "Quarantine";
range 172.16.0.10 172.16.63.254;
}
} #End of Quarantine subnet declaration
```

```
#Strict Subnet Declaration
subnet 172.16.64.0 netmask 255.255.192.0 {
option broadcast-address 172.16.127.255;
option routers 172.16.0.1;
default-lease-time 600;
max-lease-time 600;
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
#Range of address for Strict Subnet
pool {
allow members of "Strict";
range 172.16.64.2 172.16.127.254;
}
} #End of Strict subnet declaration
```

```
#Restricted Subnet Declaration
subnet 172.16.128.0 netmask 255.255.192.0 {
option broadcast-address 172.16.191.255;
option routers 172.16.0.1;
default-lease-time 600;
max-lease-time 600;
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
#Range of address for Restricted Subnet
pool {
allow members of "Restricted";
range 172.16.128.2 172.16.191.254;
}
} #End of Restricted subnet declaration
```

```
#Trusted subnet declaration
subnet 172.16.192.0 netmask 255.255.192.0 {
option broadcast-address 172.16.255.255;
option routers 172.16.0.1;
```

```
default-lease-time 3600;
max-lease-time 3600;
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
#Range of address for Trusted Subnet
pool {
allow members of "Trusted";
range 172.16.192.2 172.16.255.254;
}
} #End of Trusted subnet declaration

}# End of main network declaration
```

## A.2 Hostapd Configuration File

```
interface=wlan0
driver=nl80211
hw_mode=g
bssid=c8:3a:35:c1:e1:30
ssid=IoTSentinel
channel=1
auth_algs=1          # 1=wpa, 2=wep, 3=both
wpa=2                # WPA2 only
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_psk_file=/etc/hostapd.wpa_psk
```