

Policy Creation and Bootstrapping System For Customer Edge Switching

Fofana Ibrahima Kalil

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of
Science in Technology.

Espoo 30.11.2017

Thesis supervisor:

Prof. Raimo Kantola

Thesis advisor:

M.Sc. Hammad Kabir

Author: Fofana Ibrahima Kalil		
Title: Policy Creation and Bootstrapping System For Customer Edge Switching		
Date: 30.11.2017	Language: English	Number of pages: 8+63
Department of Communications and Networking		
Professorship: Networking Technology		
Supervisor: Prof. Raimo Kantola		
Advisor: M.Sc. Hammad Kabir		
<p>The number of Internet connected devices is increasing and have caused the shortage of the IPv4 addresses. The adoption of Network Address Translation (NAT) has solved the IPv4 address depletion problem but it has introduced “reachability problem”. This problem prevents a host on the Internet from being able to reach another host behind a NAT. While several NAT traversal methods have been developed to solve the reachability problem, no ideal solution for mobile devices has been suggested.</p> <p>Customer Edge Switching (CES) is a new experimental architecture for Internet communications that prior to allowing communication between hosts and establishes a generalized chain of trust between the hosts. CES aims to replace NAT and removes the problems known in NAT traversal methods. In addition, CES has security features which are more comprehensive in nature and can protect the interest of the served hosts over the Internet. CES proposes policy tools such as Policy Creation and Bootstrapping System (PCBS) and Policy Management System (PMS) to allow the end user to control flows emanating from the Internet to her device.</p> <p>In this thesis, PCBS was developed to provide the end user the ability to create her own policies. The PCBS has a utility tool running on the end user device called Policy App that aims to glean as much information as possible from the device and store that to the User Policy Database (UPS) for further processing, validation and modification by end user using Policy Interface. The policies that are created are then pushed to the Policy Management System (PMS). The PMS on the other hand provides the end user policies to the CES nodes that act as firewalls.</p>		
Keywords: CES, UPS, UPD, UPA, NAT, PCBS, PMS, Policy, Security		

Acknowledgements

First of all, thanks to Almighty Allah who grants me ease to successfully achieve this thesis. I thank my family and all friends for their support during my studies in Finland. I especially thank my uncle Aly Djiguine for his support.

I want to thank my supervisor Professor Raimo Kantola, for welcoming me to work in his research group, granted his trust and supported me for more than 8 months, and his advice was a great help to me.

Thanks to my instructor Hammad Kabir for supporting me all the many times for his generous and stimulating readings, and then for his confidence and his remarks.

Finally, I thank Aalto University for giving me a place to study and providing me resources during my studies in Finland.

Otaniemi, 30.11.2017

Fofana Ibrahima Kalil

Contents

Abstract	ii
Acknowledgements	iii
Contents	iv
List of Acronyms	vii
1 Introduction	1
1.1 Research Problem	2
1.2 Objectives	2
1.3 Scope	2
1.4 Structure	3
1.5 Contribution	3
2 Basics of Computer Security	4
2.1 Device security	4
2.2 Principle of security	4
2.2.1 Confidentiality	4
2.2.2 Integrity	4
2.2.3 Availability	4
2.2.4 Authenticity	5
2.2.5 Non-repudiation	5
2.2.6 Access Control	5
2.3 Principle of security	5
2.3.1 Linux architecture	6
2.3.2 Android architecture	9
2.4 End system vulnerabilities and threats	11
2.4.1 OS vulnerabilities	11
2.4.2 Hardware vulnerabilities	12
2.4.3 Application/ Software vulnerabilities	13
2.5 Attack types	13
2.6 Prevention mechanisms	17
3 State of the art in policy management	19
3.1 What is a policy?	19
3.2 Policy classification	20
3.3 Policy properties	20
3.4 Policy-Based Model	21
3.4.1 IETF policy-based management architecture	22

4	TCP/ IP model	24
4.1	Network and Protocol	24
4.2	Internet protocol	25
4.2.1	Internet Protocol version 4 (IPv4)	25
4.2.2	Internet Protocol version 6 (IPv6)	26
4.3	Transport protocols	27
4.3.1	Transmission Control Protocol (TCP)	27
4.3.2	User Datagram Protocol (UDP)	29
4.4	Domain Name System (DNS)	30
4.4.1	Domain Name Space	30
4.4.2	Name Server	30
4.4.3	Resolver	30
4.5	Network Address Translation (NAT)	31
4.5.1	How NAT works	31
4.5.2	Problem with NAT	31
5	Customer Edge Switching	33
5.1	Motivation	33
5.2	Architecture	33
5.3	Communication in CES	34
5.3.1	Inter-CES communication	34
5.3.2	Intra-CES communication	35
5.3.3	Packet forwarding in PRGW	36
5.4	Customer Edge Traversal Protocol (CETP)	36
5.5	Conclusion and research questions	37
6	Implementation and Evaluation	39
6.1	User Policy Agent (UPA)	40
6.1.1	Linux Policy App	41
6.1.2	Android Policy App	43
6.1.3	Policy Interface	47
6.2	User Policy Database (UPD)	47
6.3	User Policy Server (UPS)	48
6.4	Policy Management System (PMS) client	48
6.5	Implementation and Usage of Policy Interface	48
6.6	Usage of the PCBS	51
6.6.1	Usage for Android user	51
6.6.2	Usage for Linux user	51
7	Performance testing	53
7.1	Test tools	53
7.2	Test environment	53
7.3	Test data	54
7.4	Performance Test of User Policy Server	54

7.4.1	Test1: Average number of replies between concurrent Policy Apps and UPS without processing	55
7.4.2	Test2: Response time between UPS and UPD	56
7.4.3	Test3: Average number of replies between concurrent Policy Apps and UPS with processing	57
7.5	Test4: Policy App performance	59
7.6	Conclusion and Future works	60
7.7	Discussion	60

List of Acronyms

ALG	Application Layer Gateway
APK	Android Application Package
ARP	Address Resolution Protocol
BSD	Berkeley Software Distribution
CES	Customer Edge Switching
CETP	Customer Edge Traversal Protocol
CIDR	Classless Inter-Domain Routing
CN	Customer Network
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DMTF	Distributed Management Task Force
DNS	Domain Name System
DoS	Denial-of-Service
DRDoS	Distributed Reflection Denial of Service
DVM	Dalvik Virtual Machine
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HIDS	Host Intrusion Detection Systems
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
iCES	inbound CES
IDS	Intrusion Detection Systems
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ITF	Internet Trust Framework
LDAP	Lightweight Directory Access Protocol
MITM	Man in the Middle
NAPTR	Naming Authority Pointer
NAT	Network Address Translation
NIDS	Network Intrusion Detection Systems
NSP	Network Security Policy
oCES	outbound CES
OS	Operating System
OSI	Open Systems Interconnection
PCBS	Policy Creation and Bootstrapping System
PDP	Policy Decision Point
PEP	Policy Enforcement Point

PMS	Policy Management System
PMT	Policy Management Tool
PR	Policy Repository
PRGW	Private Realm Gateway
RAM	Random Access Memory
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SPN	Service Provider Network
SQL	Structured Query Language
SSH	Secure Shell
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TURN	Traversal Using Relay around NAT
UDP	User Datagram Protocol
UPA	User Policy App
UPD	User Policy Database
UPS	User Policy Server

1 Introduction

In the modern world of technology, Internet connectivity has become fundamental for daily and work-related activities. Individuals and organizations access the Internet using various smart devices such as tablets, phones, and computers. These devices run Applications or Apps as a means of performing particular functions and tasks on a computer or mobile device. Google's Android and Apple's iOS are the two leading mobile operating system platforms, with their respective apps markets: Google Play store and Appstore. Each of these stores has above 2 million apps [1]. While smart devices and their apps, together with an increase in Internet connectivity penetration, have brought convenience to people's lives, there is a growing set of security vulnerabilities and malicious programs targeting end users and their devices.

The mobile OS vendors provide their application stores as safety platforms from where users can download trusted and reputed application software. However, malware can still find its way to the end user devices because of the magnitude of apps published daily, which limits the ability of these platforms to control and detect all the published malicious applications. Thus, there exists the possibility of rogue applications within these platforms that an unsuspecting end user can download to her device, and they can thus pose security threats to the end user device.

There have been many high profile cybersecurity breaches in recent years. For example, Mirai is a malicious program that attacks the Internet of Things (IoT) devices and makes them controllable by a remote host forming, e.g. a botnet. These devices then can be used for large-scale coordinated attacks on networks. For example, a recent large-scale attack on Dyn DNS services affected hundreds of popular websites through such compromised devices [2]. A cyber-security firm McAfee has shown that 2.5 million Internet gadgets were infected by "Mirai" malware in 2016 [3] and five IoT gadgets are added to the Mirai botnet each minute [4]. In view of the threat landscape, organizations and individuals are increasing their spending on cybersecurity. For example, in 2017, Gartner expects the cybersecurity market to be worth 90 billion USD [5]. Despite the increase in cyber security expenditures, there has been a 24% increase in malware detected in 2016. Furthermore, in 2016, Smartphones and Tablets malware increased by a massive 99% with 179 new cyber threats per minute [3][4].

The conventional security typically employs network-based solution tools such as firewalls, intrusion detection, virus detection, and DDoS mitigation to protect the hosts. However, a common theme in information and network security is sparing co-operation in security measures. The prevalent attitude is that an organization or its security administrators care about the security of hosts and networks within their purview, without considering the fact that networks are interlinked and malicious traffic and software can and do spread. The amount of co-operation is usually limited to sharing vulnerability information and co-operating with the country-specific Computer Emergency Response Team (CERT). This cooperation, although welcome, is too little and more cooperative network security is not only possible but needed [6].

1.1 Research Problem

The above mentioned security solutions generally provide network-based security and lack the detailed knowledge of traffic expected at an individual host or its service. Thus, there is a challenge to design mechanisms that can make network-based security methods aware of interests of each host or its services, so that these interests can be protected and executed over the Internet.

An aspect to the problem is enriching network edge devices. To this end, we propose Customer Edge Switching (CES) to overcome the shortcomings of Network Address Translation (NAT) [7]. It has security features which are more comprehensive in nature and can protect the interests of the served entities over the Internet. CES establishes a connection between end users and public networks through implicit signaling. It allows the end user to control flows emanating from the Internet to his host. Within CES, every flow is admitted by policy and the source of the flow can always be immediately traced back to its source network; this makes simple flooding DDoS attacks infeasible. Additionally, network-based security at CES with full knowledge of Apps running on end hosts will admit exactly the desired traffic and drop all flows that are not expected.

1.2 Objectives

The goals of the thesis is to:

1. Design and implement utility tools that can be used to bootstrap a policy management system that needs full knowledge of the Apps that are installed on end-user devices.
2. Store applications information into a database for further validation.
3. Provide a web tool for end users to set security or reachability policies for their Apps. This policy is then stored in the policy database.

Since hosts cannot be fully trusted as they are vulnerable to malware and any layers within the communication stack can be infected, this implies a concern on how to prevent malicious packages or payloads from infecting the network-based policy management system. In this thesis, an initial system of bootstrapping the policy management system was created. Another aspect of the policy bootstrapping problem is that the end users might want to modify the policies created by any automatic tool or installation. We designed and implemented rudimentary tools that can be used for this purpose by the end user. The user modified policies are stored in the policy database for further processing in network-based tools. Our work take into consideration the possibility of hackers trying to infect the policies but this thesis leaves comprehensive analysis and testing of this question for future work.

1.3 Scope

In this thesis, a utility tool is developed, that runs on Linux and Android and will seek to gather all the required information about active apps installed in the system

such as local IP address, local port, remote IP address, remote port, the protocol used, status of the connection, FQDN and signature of the app. This utility tool together with the web tool will allow the user to set the policies for her devices. The utility tool designed is meant for enhanced mobile broadband devices rather than IoT devices.

1.4 Structure

The thesis is divided into following chapters.

The second chapter covers device security and the general principle that guide cyber and gadget security. Security vulnerabilities of Internet devices along with prevention techniques are also covered in this chapter. Chapters 3 is concerned with the fundamental principles of the Internet protocols, how devices manage and transfer information in the web

Chapter 4 introduces the concept of customer edge switching. Chapter 5 presents the concept of policy and policy-based model. This chapter highlights the IEFT policy management architecture.

Chapter 6 presents the architecture of the Policy Creation and Bootstrapping System (PCBS) and describes the components of the PCBS. Chapter 7 discusses the result of performance test of the PCBS. Finally Chapter 8 concludes the thesis.

1.5 Contribution

The developed software was published under GPLv3 in Github (https://gitlab.cloud.mobiledn.org/CES/policy_tools/).

2 Basics of Computer Security

This chapter is concerned with attacks and security of computer systems. The chapter discusses the device vulnerabilities to threats posed from the Internet on Android, IOS, and Windows devices. Finally, the prevention technique used against attacks will be introduced.

2.1 Device security

Broadly, device security is defined as the measures or steps to protect the computer system and the data stored in it [8]. In the recent past, some malwares encrypted the victim's data, thereby denying access to it and demanded a payment in order for the victims to re-access their data. Victims in this scenario varied from private firms to critical government departments, such as health organizations. The attack was subsequently blamed on ransomware [3].

In the context of computer science, computer security is “the ability of a system to protect information and system resources with respect to confidentiality, privacy, and integrity” [9].

2.2 Principle of security

Device security can be summarized but not restricted to three main aspects: CIA. That is confidentiality, integrity, and availability. In addition, authenticity, non-repudiation and access control are some of the principles of security [9]. We discuss these principles below.

2.2.1 Confidentiality

The word confidentiality and privacy are often used interchangeably. In terms of device security, confidentiality means keeping data private and ensuring restricting access by only the authorized individuals. A cyber security system will seek to ensure that clients data is not vulnerable to unauthorized third-parties.

2.2.2 Integrity

Integrity is one of the computer security principle involving to keep the information accurate, consistent and trustworthy. It refers to the state of data which, during processing, storage or transmission, shall not be subjected to any unauthorized changes. This implies that only authorized persons have an access or ability to modify the information. The integrity is violated when the information is corrupted, disrupted or damaged.

2.2.3 Availability

The principle of availability ensures guaranteed access to the data at all times to the authorized users. It deals with measures against delayed access of the data and

protection from unauthorized retention of the data. It guarantees that clients have control over their own information and can use it in the best way possible.

2.2.4 Authenticity

The principle of authenticity means that only legitimate users access the data or the system resources. In other words, It ensures that exchange information or transaction is from the source it claims to be from. This principles involves proof of identity.

2.2.5 Non-repudiation

This principle ensures that a person or process engaged in a communication can not deny having received or sent a message.

2.2.6 Access Control

This principle defines what information or resources an entity can access and what she can do with the information.

2.3 Principle of security

Operating system (OS) is the software interface that acts as an intermediary between the system hardware and the system user. It manages system resources, controls input and output, memory allocation and monitors peripheral devices. Figure 1 presents the common structure of computer system. Today, there are many types of OS such as Android Mobile OS, Linux OS, Apple mobile OS, Windows phone etc. This thesis discusses the architecture of Android Mobile OS and Linux OS, as well as their vulnerabilities and threats.

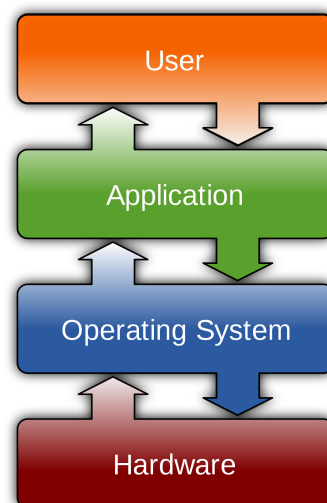


Figure 1: Common structure of a computer system [10]

2.3.1 Linux architecture

Linux computer system comprises of different layers which have their own features and mechanisms to enable communication. At runtime, Linux can be classified into: User Mode and Kernel Mode [11]. Figure 2 presents the general architecture of a Linux system.

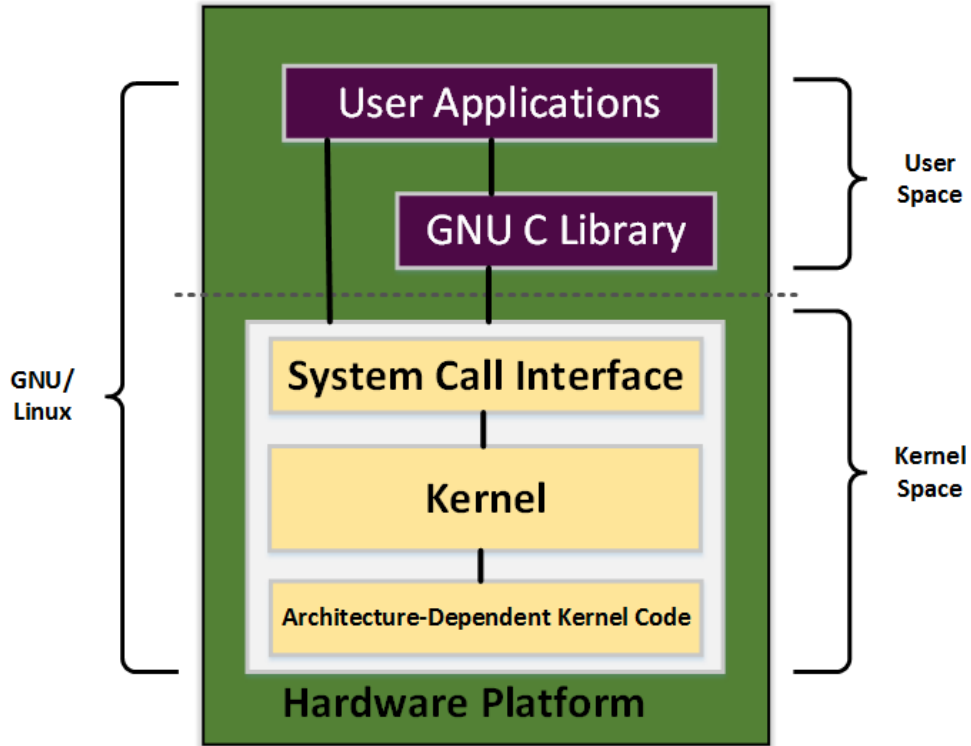


Figure 2: Architecture of Linux system

As shown in Figure 2, the user applications run in the user space. The GNU C Library in the user space serves as an interface between the user application and the kernel. The Kernel space is below the user space and above the hardware platform. It allows Linux to work on multiple hardware platforms. Moreover, the system call interface and the kernel itself are located inside the kernel space. The hardware platform which includes physical devices of the end system i.e. keyboard, mouse, a network card are located at the lowest level in the architecture.

The kernel internal architecture is layered into many different subsystems. Each subsystem provides specific functions and services to other subsystems. Figure 3 shows the internal architecture of Linux kernel.

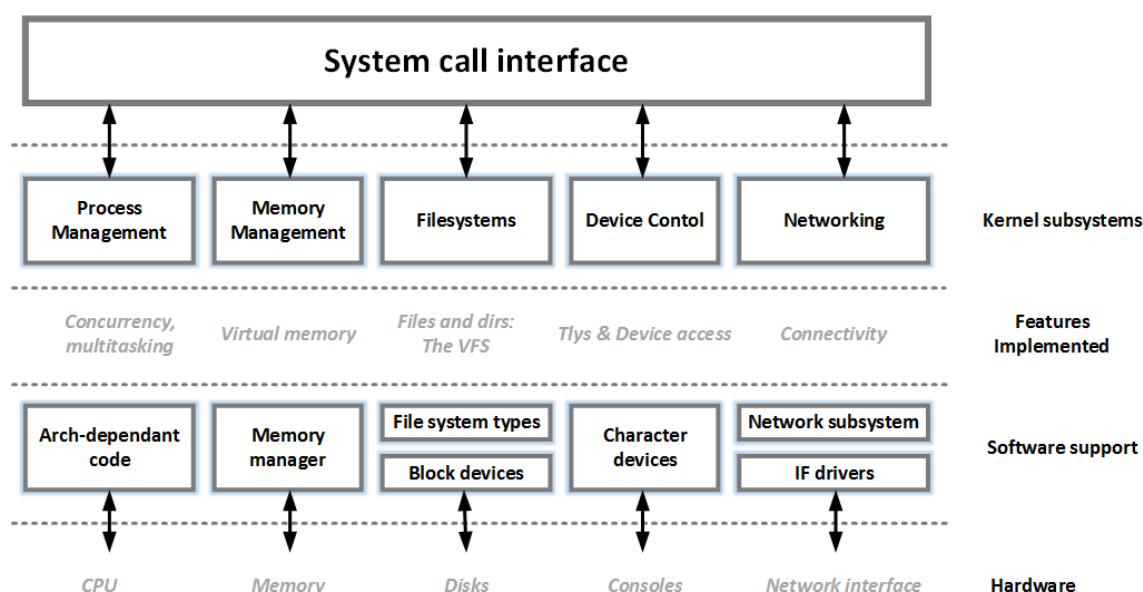


Figure 3: Architecture of Linux kernel

Properties of Linux kernel

Linux kernel properties component are defined below:

Process management

The process management is the most critical subsystem of the OS. Its functions can be classified into three parts:

1. creates and destroys processes and also handles their connection with I/O peripherals.
2. the process manager is used to perform communication between different processes.
3. process scheduling controls how the processes will be sharing the CPU. A process is defined as a set of running programs in memory. The process is assigned a numerical value called process identification (PID) to uniquely identify a single process in OS.

Memory management

Memory management is another important subsystem that manages the memory in the kernel. It monitors the allocated space in the system, which memory space is allocated for a process and decides the amount of memory that should be given to a process. This subsystem also manages physical and virtual memory and gives larger virtual address space, reasonable virtual physical memory allocation and protection.

Virtual memory allows the system to seemingly have more memory than it actually has in Random Access Memory (RAM).

File system

Linux heavily depends on the file-system. In Linux, everything is treated as a file. The concept of the file-system is not only applicable to the text files, but also covers images, compiled programs, partitions, directories, or hardware device drivers. A file is defined as a data container that is structured as an ordered string of bytes. Moreover, a file is located following a tree-structured space where the name uniquely identifies the file in the directory where it is located. In Linux, the root is the source for all resources files and it is noted as (/). Anything addition to the root is referred to as sub-directory and each sub-directory can have further sub-directories. Figure 4 presents a directory tree in Linux system.

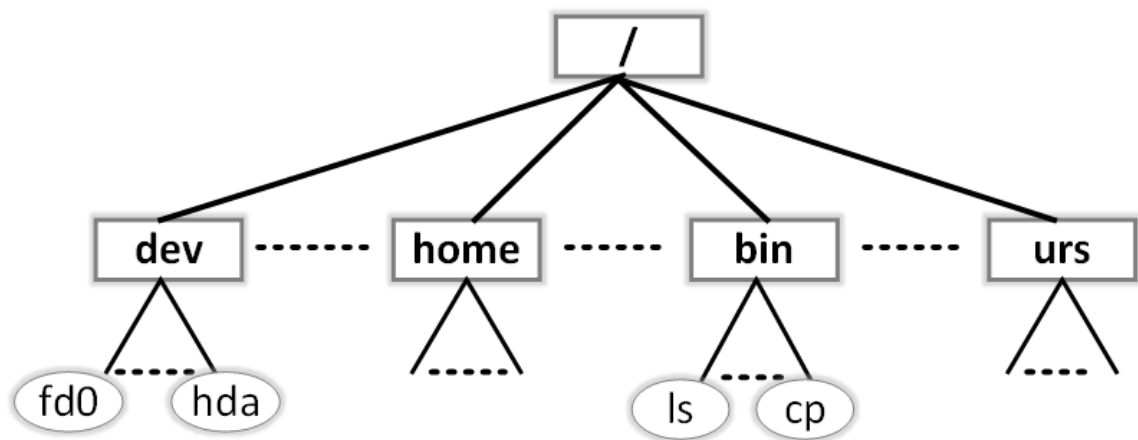


Figure 4: Directory tree in Linux system

Some of the sub-directories are:

- /bin, command binaries for all users
- /boot, boot loader files such as the kernel
- /home, users home directories
- /mnt, for mounting disk storage
- /root, home directory for the root user
- /sbin, executables used only by the root user
- /usr, where most application programs get installed

In the kernel, each file is identified by a unique number rather than its name. This unique number is called inode. Each inode has an entry in the array of inodes that have the information required to access the file [12]. The list includes:

- The inode number which is related with a particular file
- The owner of the file which is a user or a group of users
- The type of the file (device, directory, pipe, ...)
- The file creation, access and modification times
- The size of the file
- The pointer to data blocks that store the file's content

Device driver

Most of the source code in the Linux kernel exists in device drivers. Each driver allows a single hardware device to be used. There are many types of devices connected to computer systems: CPU the brain of the system, network devices, storage devices and also human interface devices like keyboard, mouse, and screen. The device driver is an abstract layer between the concept of software and hardware (that gives a defined interface to a higher level application so that the device operation details are hidden).

Networking

The networking subsystem is an abstract layer that is responsible for providing the network connectivity between the Linux system and other devices. The subsystem follows a layered architecture design that consists of three layers. The sockets layer is the highest layer in the architecture that provides the standard API to the networking subsystem and gives a user interface for a variety of networking protocols. The protocol layer is below the socket layer that consists of transport and network layer protocol. The Internet Protocol (IP) is the core network layer protocol that is located below the transport protocol. The low level is the lowest layer in the architecture that gives access to the physical devices.

2.3.2 Android architecture

Android OS is based on Linux kernel and created by Google. Android uses Linux kernel version 2.6 for the core system services such as memory management, security, process management, network stack and driver model. In addition, Android OS has four layers and each layer has its own features [13]. Figure 5 presents the components of the Android system.

At the bottom layer of the Android architecture is the Linux kernel which provides the basic functions such as device management, inter-process communication (IPC) and memory management etc.

Above the Linux kernel, there is a set of libraries written in C or C++ i.e., an SQLite database which is used to store the information of the applications. In the same layer, there is a section called runtime which has two boxes including Java

libraries and Dalvik Virtual Machine (DVM) sandbox which is used to support Android application sandbox.

The application framework is the third layer from the bottom. This layer provides many higher level services to the user application. The services of application framework include:

Activity Manager: This service monitors all aspects of the application lifecycle and activity stack.

Content Providers: It is a service that enables applications to publish and share information with other applications.

Resource Manager: This service gives access to non-code embedded resources such as strings, user interface layouts, and color settings.

Notifications Manager: It is a service that enables applications to show alerts and notifications to the user.

View System: It is an extensible list of views used to create user interfaces of applications.

Package Manager: This service provides various type of information related to the application packages that are currently installed on the android device.

The Applications are located at the top layer of the architecture. This is the layer where the device users can install applications. Moreover, these applications can come from the official applications stores or third-parties.

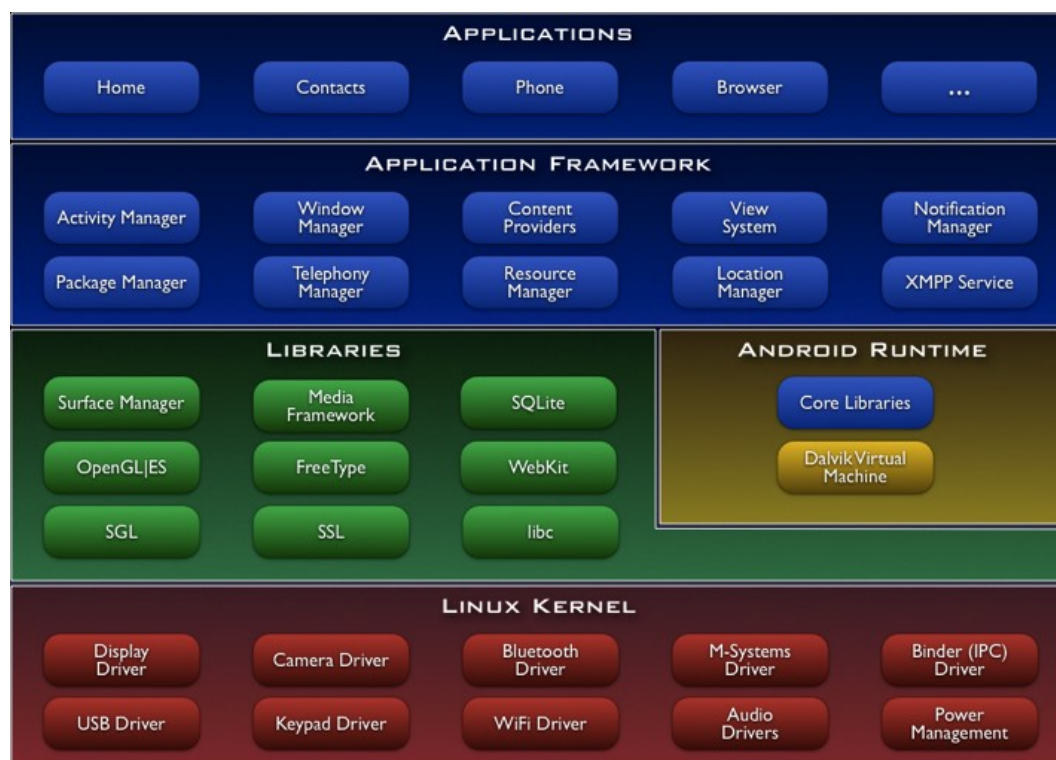


Figure 5: Internet architecture of Android [13]

2.4 End system vulnerabilities and threats

This section will discuss the type of vulnerabilities in the end user device and different attacks used against the device.

2.4.1 OS vulnerabilities

Today, mobile devices are often connected to Internet making them vulnerable to a number of threats that target the device's operating system. At the same time the malware industry is also growing both in terms of technology and structure. The threats can be broadly categorized into three categories: Malware, Vulnerabilities, and Attacks.

Malware

A malware is a malicious software aimed at gaining access to private information so that it can be destroyed, duplicated or altered. Malware can also make the information unusable by the device users or cause device breakdown [14]. Malware is usually not installed by the user but gain access to the system through Internet connection taking advantage of the vulnerabilities in the operating system. Most common forms of malware are the viruses, Worms, Trojans and Spyware. The first case of malware for smartphones was reported back in 2004 where a malware known as Cabir was created for the Symbian operating system[15]. The most affected smartphone by this malware was the Nokia 60 series. This worm used Bluetooth connection to spread itself. An infected phone can easily be known as it creates the word "Cabire" on the screen. Apple Inc. phones are more protected from malware owing to their closed system. Android OS is specifically targeted by malware because of the possibility of installing applications from many insecure third-party sources and because of its large market share.

Trojans

A Trojan horse is a malware program that poses as a legitimate program. It is disguised as a legitimate software and end users are easily tricked into running it on their systems. After users install the dummy app or save the file, the Trojan assumes control of the device, and the device security is compromised.

The main use of the Trojan software is to seize the device management and control information rather than to spread itself. It is in this sense that they differ from worms and viruses [15]. They are transmitted under the cover of other files and often unintentionally activated by the user. Once activated, they get the full control of the device in the background making them hard to detect. To minimize the risk of Trojan software, it is of utmost importance that downloaded application is checked and verified for its correctness. Android users should limit their source for applications to Google App Store. Even though there can be buggy applications on playstore.

Virus

Mobile viruses are malicious software that infect devices when executed by replicating themselves through modifying existing programs and subsequently inserting their own code. There are however fewer viruses that can attack mobile devices as compared to computer devices, but this will change especially with the rise in mobile device usage and the complexity of mobile operating systems.

A virus has the ability to transfer itself between the existing documents and changing their locations [15]. It can also distort the information in the documents making them unusable to the user. Virus may cause the phone applications to be unresponsive or slow. Virus is however easy to detect, unlike the Trojan malware.

Worm

A worm is a self-replicating computer program that attacks an operating system by sending copies of itself to other devices within the network. Worms mainly consume bandwidth in the mobile devices and might even be capable of compromising the integrity of stored files.

When a worm infects the device, it will replicate itself and over spreading to other devices within the network. A malicious code in the worm called the payload might compromise the integrity of the data or even launch a ransomware attack.

Worms do not require the users' interaction to cause the damages to the mobile device as they are transmitted using text messages or picture messages. Worms are in fact a kind of virus, only that they do not require users' intervention to harm the mobile devices.

2.4.2 Hardware vulnerabilities

In regard to hardware vulnerabilities, the first consideration which causes the problem is age of the mobile device. This is because, the device manufacturers do not support the devices after certain date. The older devices therefore, do not receive the latest security updates from the manufacturer. The second issue is the inability of the device in assuring, the safety of the ports used to connect to a network. Due to the lack of navigation limit used in the Internet's environment and the lack of firewall to control the navigation creates a vulnerability to the mobile devices. A hacker can therefore easily gain access to the mobile device using the insecure port [9]. The use of a firewall software reduces the risk of remote access of the device as it requires the user to have permission to connect to the specific Internet device. Another challenge comes from Jailbreaking which is the process used to download an application on Apple iOS that does not belong to the Apple iTunes store. This method causes unauthorized changes on mobile devices that do not have a firewall. Jailbreaking makes the mobile device vulnerable to threats because the jailbroken device cannot receive security updates from the manufacturer.

2.4.3 Application/ Software vulnerabilities

The out-datedness of the operating system poses a security threat to mobile devices [16], since it contains many unpatched security vulnerabilities from old software [17]. Another challenge comes from applications installed. Android OS supports the installation of applications from Google Play or any another file system. The downloads from Google Play are all secure because the packages Android Package Kit (APKs) are from a protected area. However, APK files downloaded from third-party app stores and local storage units such as SD card are usually unprotected, hence pose a security threat to the mobile device. Security firms try to meet such vulnerabilities through new versions or patches. Another vulnerability of mobile devices is brought about by shared open source software (OSS). OSS is software that has its source code freely available to anyone. The design of OSS system usually contains some common open source components such as Linux and Web Kit. Such components have a reusable structure so as to minimize the costs, a common practice by the manufacturers of large open source systems like the Android. When a vulnerability is discovered in the Android OS due to the use of Web Kit or Linux, a patch was created in order to solve this problem in the mobile devices. But this problem also affected Apple's iPhone-like Web Kit and BSD kernel derivative [18]. At this point, the problem is not the reuse of the component but rather where it is applied.

2.5 Attack types

The attacks can be classified into active and passive attacks as shown in Figure 6 [19]. A passive attack involves interception of data without changing the collected data and the various techniques used for the interception include sniffing, keyloggers, traffic analysis and release of the message. A active attack can be classified into three main categories which are interruption, fabrication, and modification. The techniques used in the interruption are Denial of Service (DoS), Structured Query Language (SQL) injection attack, Distributed Denial of Service (DDoS) and Distributed Reflection Denial of Service (DRDoS). In fabrication, the techniques used include replay attack and masquerading. Lastly, man in the middle attack technique is used in modification.

Passive attack

A passive attack is an attack where an attacker listens to the communication between two communication parties and records the information. However, the attacker does not alter the recorded data and that makes it difficult to identify this type of attack. Examples of passive attacks include sniffing, keyloggers, traffic analysis and release of message [19].

Sniffing

Sniffing is a type of passive attack where an attacker captures the information that is being sent from one device in the network to another device in the network without

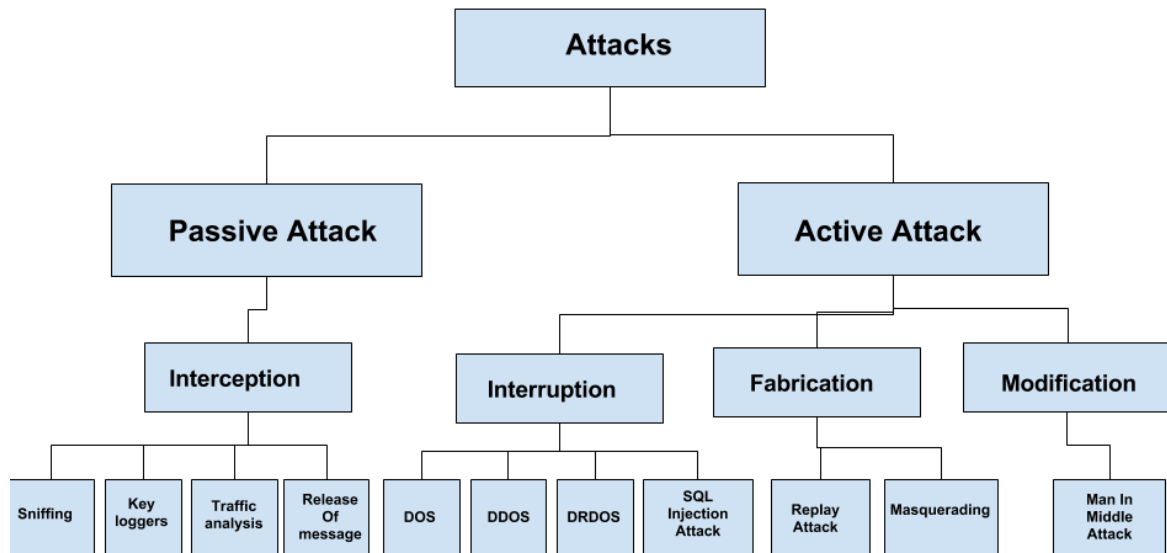


Figure 6: Classification of Network Security Attacks [19]

the consent of the sender or the receiver. The data gathered by this malicious user may include usernames and passwords which will allow the attacker to have access to a network or computer.

Keyloggers

Keyloggers illegally install a software on the victim's machine that will record keystrokes entered by the victim. A keylogger is often seen as a professional tool but it can be used for a criminal purpose such as capturing sensitive information (e.g., password, bank details).

Traffic analysis

By traffic analysis, the attacker captures encrypted information on a communication channel and extracts the data using traffic analysis tools. Figure 7 depicts an example of traffic analysis attack.

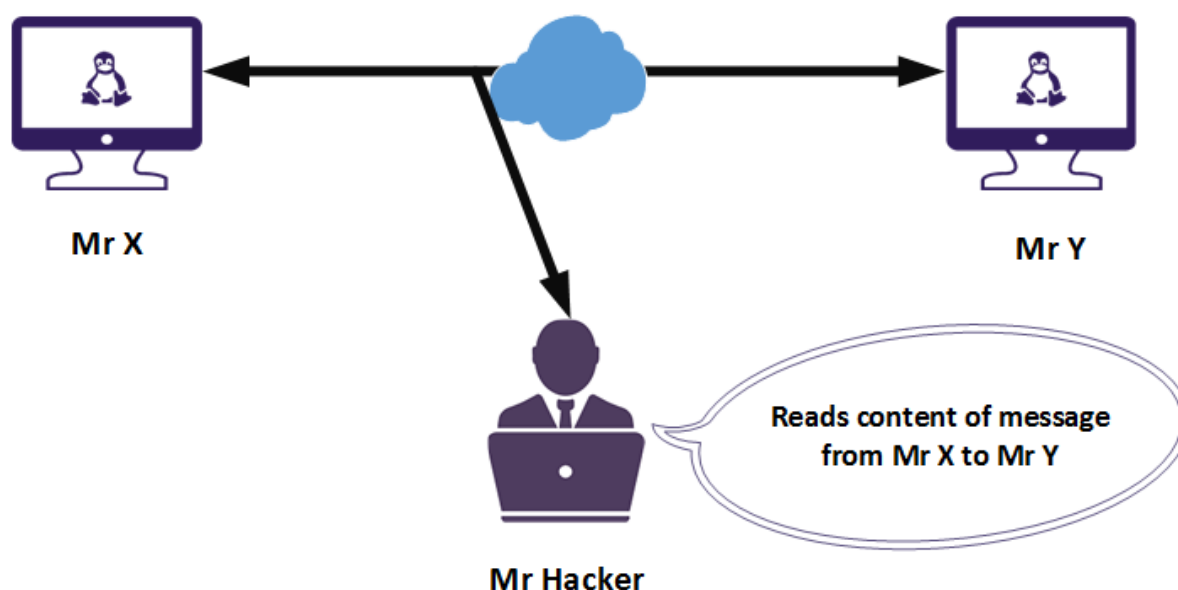


Figure 7: Traffic analysis attack

Active attack

An active attack can use the data recorded during a passive attack, such as usernames and passwords. It may alter the information. An active attack poses various threats such as DoS, SQL injection attack, DDoS, DRDoS, replay attack, masquerading and man in the middle attack.

Denial of service attacks (DoS)

DoS attack is one the most common attacks on the Internet. The goal of the attacker is to make the services or resources unavailable to legitimate users for an indeterminate period of time. This type of active attack can occur in several forms. The malicious user can flood the victim computer with many connection requests that the victim cannot handle. This will result in blocking all the incoming connection requests to legitimate users since the server has several connection requests from the attacker that have not been completed. Additionally, DoS is a costly type of attack since it interrupts the normal course of business transactions; the organization may lose money. DoS attacks can be combined with other types of attacks such as Synchronization attack (SYN) and Internet Protocol (IP) spoofing.

Distributed denial of service attacks (DDoS)

DDoS is a type of an active attack, in which the victim host is targeted by several machines. This attack is considered as the most devastating form of attack. DDoS consists of three participants; which are the Master, the Slave and the Victim. The Master is the initial source of the attack such as the attacker or machine responsible.

The Slaves are the vulnerable machines or the network that the Master gains control of and to which it installs attack software. The Victim is the target host or server that is being attacked. The Master commands the Slave(s) to launch an attack on the Victim's machine, since the attack comes from multiple hosts simultaneously, it is very difficult to trace back or stop the attacker. In other word, the actual attacker cannot be identified, even if the victim can trace back the intermediate sources [19]. Figure 8 depicts a scenario how the victim is being attacked by the malicious user.

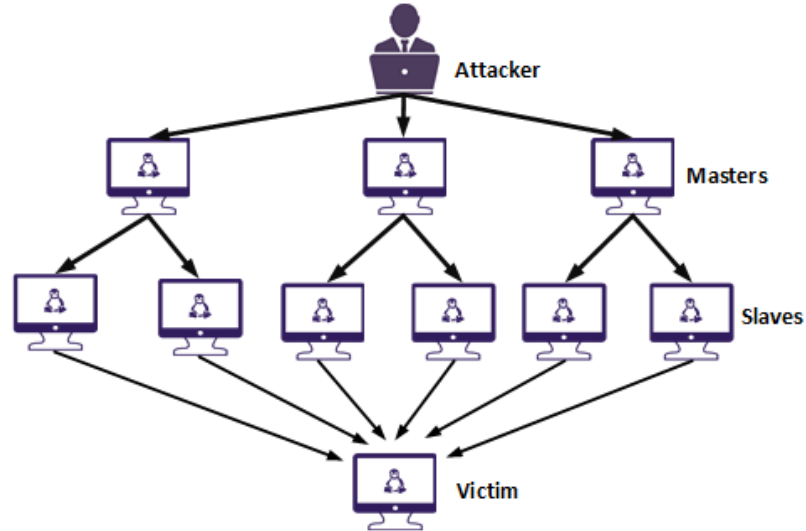


Figure 8: Distributed Denial of service attacks

Distributed reflection denial of service (DRDoS)

This type of attack involves using DDoS technique by sending several spoofed request to legitimate user machines. Moreover, in DRDoS attack, the attacker cannot be detected due to spoofing [15]. An example of reflection attack is performed by sending spoofed DNS queries with the victim's IP address given as the source address. DNS will then send responses to the victim.

SQL injection attack

SQL injection is a well-known method of attacking Web applications. This type of attack uses a technique of inserting SQL code into a program or query or to inject malware into a computer for remote commands that can read or modify a database as well as to alter the data on the website [20].

Replay attacks

This attack allows attackers to duplicate a valid data between sender and receiver and re-send a stream of data again to the receiver to prove his identity [15]. An example of such an attack may occur between two communicating parties *Bob* and

Eve, where *Bob* is sharing his key with *Eve* to authenticate himself, but during that communication process an attacker *Ibra* is capturing the communication and duplicates the data so that he can authenticate as *Bob* to *Eve* later.

Masquerading

Masquerading is a type of active attack, where the attacker impersonates the identity. This allows the attacker to gain access to the resources as a legitimate user.

Man in the middle attack

Man in the middle (MITM) is a type of attack in which a hacker listens to a communication between two entities and impersonates both entities in order to get access to the communication stream. This type of attack enables the attacker to alter the information by pretending to be one of the parties [21]. Most "man in the middle" attacks consist of listening to the network using tools such as a sniffer, ARP spoofing or DNS cache poisoning, these tools will allow the attacker to control the exchange traffic. Figure 9 presents MITM attack where the intruder intercepts the stream of data between two communicating parties.

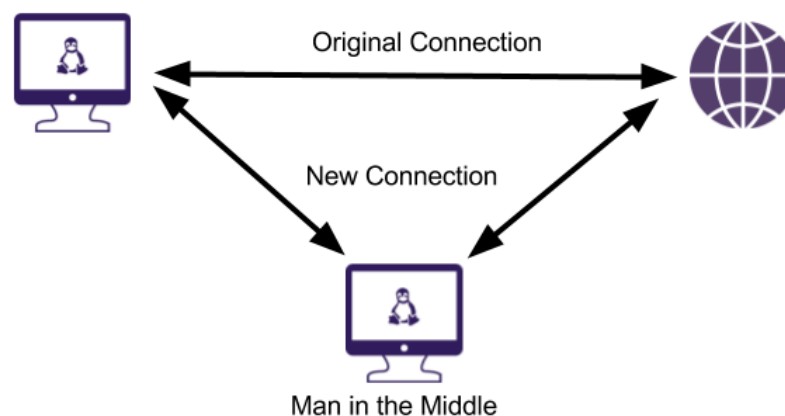


Figure 9: Man in the middle attack

2.6 Prevention mechanisms

These are protection measures implemented in a computer or device network system to ensure the integrity of the system. Security solutions comprise of three basic elements: prevention, detection, and response. In this section, we will delve into the measures that can be used to ensure the security of our systems.

Signature

Since the hash value calculated over a message only ensures the integrity of the message sent, when transmitting the message, the computed hash value is encrypted with the sender's private key, and in this way a digital signature of the message is generated. The use of digital signature validates that the message has indeed come from the expected source since only the claimed sender could have access to the private key which generated the signature, the receiver decrypts the hash value using the sender's public key and the hash value is compared with recomputed hash value.

Firewalls

A firewall is literally a wall that isolates a private network from the malicious actors. This isolation mechanism helps protect the private network from malicious attacks from the public network [22]. The firewall has a set of rules and protocols that filter and allow selective access to the private network. A firewall can be defined as a combination of both hardware and software systems that are put in place in order to protect from malicious intrusion from hackers in the external networks, where they are located. Firewalls, can be classified into: network-based firewalls and host-based firewalls. Network-based firewalls are deployed within the network and monitor traffic in and out of the network while host-based firewalls monitor traffic at the host level [22].

Intrusion detection systems

Due to the limitations of the prevention mechanisms, new intrusions continually emerge which creates the need for an intrusion detection system (IDS) [23]. The IDS system detects any possible violations of a security policy by monitoring the activities and responding to the ones it considers intrusive. Once an attack is detected in the network, the response activity is initiated to prevent or minimize the damage the intrusion caused to the system. An IDS also serves the purpose of providing information about the intrusion techniques which is important in enhancing the understanding of the attacks. Such information is also important in informing the administrator for the prevention and mitigation procedures. This device or system detects any unauthorized activity in the network traffic and reports it to the network administrator. This is because all legitimate users have a limit to the resources which they can access while a malicious user will show discrepancies and abnormal behavior [23]. Intrusion Detection Systems can still be categorized into: Host Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS). HIDS monitors any attack that targets the host while NIDS handles an attack against the network [23].

3 State of the art in policy management

This chapter will present the concept of the policy and its properties. Next, an overview of Policy Based System will be briefly explained. Finally, the chapter will conclude by presenting IETF policy-based management system.

3.1 What is a policy?

Policies are developed in various technologies to control their behavior, and many of them use a different types of definition. Therefore, there is no standard definition for the policy e.g, Computer networking , each area has their own way of defining policy.

The Internet Engineering Task Force (IETF) tries to produce highly recommended engineering documents that influence the design and usage of the Internet. Therefore, they have published many RFC's that covered policies and policy based system. Some of the definition of the "policy" given by IETF are:

- A policy is a definite goal, course or method of action to guide and determine present and future decisions [24].
- A policy is a set of rules to administer, manage, and control access to network resources [24].

In policy-based network, a policy is a set of rules that governs choices and behavior of a system. A rule defines the action(s) that must take place when specific condition(s) exist.

Generally, a policy comprises of the principles and guidelines that enable adequate management of a system. In cyber security, policies are often used to guide the participating entity that operates within the realm of internet security. In a large scale, these entities could be departments within the state working alongside private institutions to make the global internet a more secure place to surf. Most nations have an overarching umbrella cyber policy that influences how the different entities cooperate, referred to as a Network security policy (NSP) [25]. NSP harmonizes and outlines guidelines of policies along with how to enforce. It streamlines the whole area and sets out the basic tenets and premise upon which the whole cyber security infrastructure of a network or company is built.

There are many definition of policy in literature. For the purposes of this thesis we define policy as a set of rules either on a rather abstract level of principles or on a concrete level such that the rules can be directly executed by a computer system. In some policy systems particular types of abstract rules can be translated to executable rules using an off-line computer tool.

In this thesis, we are interested in creating policies that can actually be executed, so more abstract rules that are meant for people to follow rather than computers to execute are out of scope.

3.2 Policy classification

Policies can be classified into following based on their function and intent [26].

Configuration policy

Configuration policy describes the setup of a managed host. For instance, the per-hop default sending behavior of a router can be indicated as a configuration policy.

Installation policy

Installation policy indicates what applications or software are allowed to be installed on a device, and the setup of the components that perform the installation. This policy commonly represents a particular administrative authorizations.

Security policy

This policy deals with verification that the end user is actually who he purports to be by choosing and applying suitable rules such as authentication mechanisms, allowing or denying resources. For instance, access list of network policy is a security policy that defines the employees that are allowed or denied to enter a network.

Service policy

Service policy are used to indicate network or other services. This policy specifies the services that are available in the network. For instance, a service policy may indicate to allow only outgoing connection.

Usage policy

This policy defines a specific binding of application flow to the available services that are set in the network. For example, a usage policy may specify to drop any VoIP traffic that is consuming more than 10Mbps in the network.

Error and Event policies

Error and Event policies indicate the type of action that a system must perform when an event occurs. For instance, a policy can be set in the event when the bandwidth consumption of a host, alternatively the number of streams coming from a host, is over the limit threshold level, shutdown the port to which the host is associated. such policies will prevent the DoS attacks.

3.3 Policy properties

The most challenging aspect of managing modern information technology systems is ensuring data security. Alongside other challenges, policy properties typically revolve around issues concerning accessibility, preferences and security

Accessibility

Policy accessibility is the case where the system has set rules governing who or what can have access to the private network. The duration of time such as rights are available and when they are waived off.

Preferences

Preferences can be defined as user-specified settings of parameters in interactive computer software. Preferences allow the end user to set many aspects of a session of application behavior. For example, end users may want to set logging and notification preferences, under this policy property the user will have the leverage to determine what sort of notification he or she gets.

Security

In the context of cyber security, there can be network security policies and end user security policies.

A network security policy basically helps in protecting a network device from network security threats – both internal and external – from the organization or network [27]. It is generally a document and varies based on the underlying environment, organization and/or legal requirements. A network security is needed to streamline security matters within the organization and prevent it from being ineffective. A network security policy needs to meet certain criteria for it to be effective. It has to be implementable through existing network technologies in the market that are also appropriate. The network security policy must be viable with security tools where it's also feasible for the firm both in term of finance and technology. The policy has to define demarcations of responsibility for administrators and end users alike.

End user security policies have to inspire a sense of security for consumers. Full disclosure of vulnerabilities to the end users is key in boosting end user confidence. Storage of personal data should be in open formats that are compatible and accessible to the end users. Any form of encryption has to be performed at both ends of the communication network. End user security policies should also be privacy friendly that allows for trustworthy systems to be developed. There should be a deliberate attempt made to protect the clients' private data.

3.4 Policy-Based Model

Policy-based management is a promising model in the management and security of enterprise network systems. This management model provides for the development of complex systems which are able to deal with the ever-changing information technology world. Policy-based management systems have been successfully specified by organizations such as the Internet Engineering Task Force (IETF) and Distributed Management Task Force (DMTF) [24].

In the policy-based model we define policies as event triggered condition-action rules that can be adaptable management actions. These management actions in

a system may include server backups, user registration or software installations. Authorization policies are used to define what resources a user can access in the system, in addition, security management policies are required to define what happens when a violation of security protocols is detected for instance a brute force login attempt.

An important element of a policy-based management system is that it separates the policies governing system from its basic functionality leading to lowered maintenance/upgrade costs and more flexibility. Enterprise systems may have millions of users and resources. It is cumbersome to specify policies that govern individual entities, therefore it must be possible to specify policies that govern a group of resources or users. Policies are summarized and extracted from business goals and agreement within relation.

A Policy-based model provides many benefits such as being easy to upgrade and it's extremely flexible and dynamic. This has motivated the deployment of policy-based techniques for quality of service as well as for Service Level Agreements (SLAs) .

Merits of policy-based management are enormous, for instance, in case of an upgrade of a system we don't need to recode but we simply add or modify new policies, furthermore it is extensible to wishes of the users.

3.4.1 IETF policy-based management architecture

The IETF policy framework is a policy-based management system for providing admission controls decisions in integrated management systems. The IETF policy framework has four functional blocks: Policy Management Tool (PMT), Policy Repository (PR), Policy Decision Point (PDP) and Policy Enforcement Point (PEP). Figure 10 describes IETF policy framework architecture.

- Policy Repository (PR) is a database used for the storage of all policies. The stores policies are access by the PDPs by means of repository access protocol. IETF recommends the use of Lightweight Directory Access Protocol (LDAP) [28]
- Policy Decision Point (PDP) retrieves policies and interprets them, it also receives policy requests from Policy Enforcement Points and returns policy decisions to them.
- Policy Enforcement Point (PEP) is a network devices such as a router, firewall or host that is responsible for enforcing policy decisions when a criterion is satisfied from the PDP. The PEP collects the information about the state of the network and describes its network attribute or characteristics to the PDP.
- Policy Management Tool is a user interface for creating, viewing, updating, enforcing and validating policies.

In accordance with the IETF policy framework, one PDP is responsible for more than one PEP. The PEP tells the PDP what actions it is able to enforce and the format

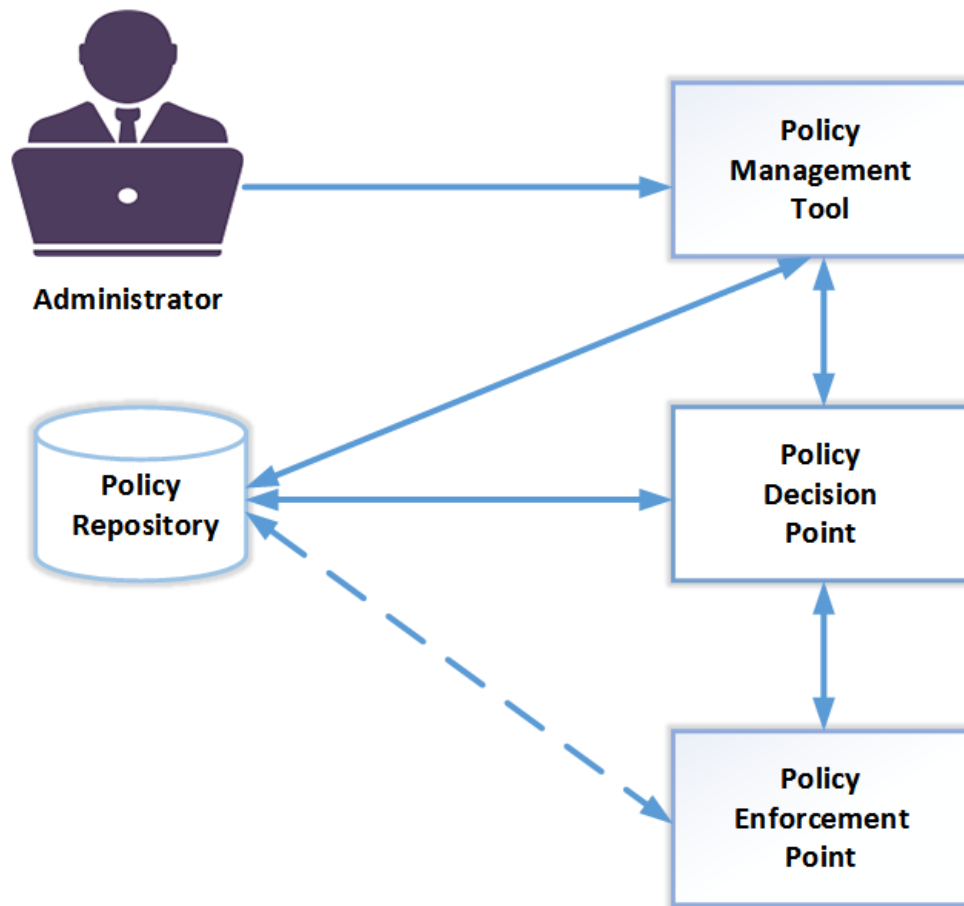


Figure 10: IETF policy framework architecture

of how it wants the actions represented in the policies. The PDP is responsible for making the high level decision based on the policies stored in the Policy Repository. Additionally, the PDP is responsible for translating the policies into a language that a network device can understand. The PDP and PEP are logical isolated [24]. IETF framework defines many protocols such as DIAMETER and Simple Network Management Protocol (SNMP) as the protocol for policy exchange between the PEP and PDP, however other protocol like HTTP may still be used [24][29].

The upsurge of the Internet and increased use of complex systems has motivated a paradigm shift from the traditional centralized model to a decentralized paradigm. The IETF defines an architecture of a decentralized system with distributed managers who can also act in a manager role with all privileges as well as an agent role when they are remotely controlled or observed

4 TCP/ IP model

This chapter will mainly focus on explaining the TCP/IP model . Next an overview of NAT is presented. Finally this chapter will conclude with the discussion on the Domain Name System (DNS).

4.1 Network and Protocol

A network connects two or more electronic devices physically (wired) or logically (wireless) for the purpose of exchanging information. Examples of such devices are computers or printers which are referred to as *host* in the network. Since the primary goal of any communication network is to exchange information and share assets between hosts, the network must follow specific rules. The set of rules is referred to as network protocol, which allow to discover the resources that network shares between communication hosts in order to ensure reliability and security in the network.

In 1978, the International Organization for Standardization(ISO) developed a set of recommendations for connecting heterogeneous devices to a network which is referred as Open Systems Interconnection model (OSI).

The Internet relies on TCP/IP model which was developed by the US department of Defense Advanced Research Projects Agency (DARPA) . This model has 4 main layers: the Application layer, the Transport layer, the Internet layer and the Network layer [30]. Figure 11 shows the architecture of the OSI model and TCP/IP model.

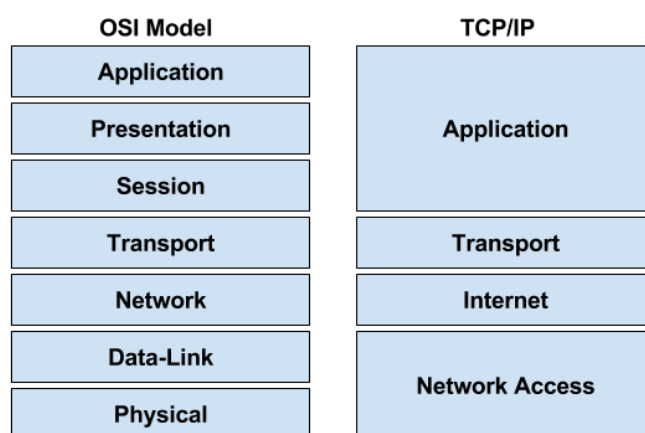


Figure 11: Internet model

Network access layer

The lowest layer in the TCP/ IP model is the network access layer, as the name suggests this protocol is used to connect to the hosts. It is also known as the host to the network layer and defines the details of how data is physically sent in the

network including the details of how bits are signaled and the hardware devices that interact directly with a network medium.

Internet layer

The second layer in the TCP/ IP model is the Internet layer. This layer is in between the network access layer and the transport layer. The primary function of the Internet layer is to pack data into packets known as IP datagrams [30]. These datagrams have an IP address that is used to transfer them between hosts and networks. The Internet protocol is used in this second layer. It is important to note that this layer essentially holds together the entire TCP/ IP model.

Transport layer

The third layer of the TCP/ IP model is the transport layer. The transport layer makes a decision whether the data should be transmitted in a parallel path or single one [8]. Also, it is in charge of functions such as multiplexing, segmenting and splitting, it is important to note that in this layer an application is able to read and write the data. The layer also arranges the data in tiny packets that are sent sequentially. Also, header information is added in this layer.

Application layer

The fourth and last layer of the TCP/ IP model is the Application layer. This layer has protocols which define how host programs interact with transport layer services to use the network. Protocols in the application layer include DNS (Domain Naming System), HTTP (Hypertext Transfer Protocol), Telnet, SSH and many more common protocols [30].

4.2 Internet protocol

Basically, the Internet protocol is the defined set of rules set to govern all Internet activity. The main purpose of the Internet protocol is to facilitate the delivery of datagrams from the sender to the receiver. This is achieved through a process called encapsulation which roughly is defined as putting tags in datagrams, the tags, in this case, are address information. The Internet Protocol specifies the addressing of the datagrams, however, most networks combine Internet protocol with a protocol known as Transmission Control Protocol (TCP). This protocol establishes a virtual connection between two hosts. The current version of IP is IPv4. A new version, called IPv6 is still under development and it is expected to solve flaws associated with IPv4.

4.2.1 Internet Protocol version 4 (IPv4)

IPv4 was the first widely adopted version of the Internet Protocol by IETF. IPv4 has shaped the establishment of the Internet. Under IPv4, Internet connected devices

are identified by IPv4 addresses. IPv4 addressing scheme identifies Internet device interfaces uniquely and this facilitates the sending and receiving of information without conflicts. Today, the most critical problems with this IPv4 is the depletion of the address space, issues in efficient routing and lack of security. As shown in the Figure 12, IPv4 has 32 bit addressing and the number of connected devices is raising [31]. According to Gartner Inc [32], 6.4 billion devices were connected in 2016 and forecast shows that the numbers of the connected devices will reach 20.8 billion by 2020. Several solutions have been proposed to tackle the IPv4 address depletion problem. These solutions include Classless Inter-Domain Routing (CIDR), Network Address Translation (NAT) which was developed as a short term solution for the IPv4 address depletion problem and the Internet Protocol version 6 (IPv6) which is the long term solution.

Bits			
0	4	8	161931
Version	Length	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live	Protocol	Header Checksum	
Source Address			
Destination Address			
Options			
Data			

Figure 12: IPv4 packet header

In the classical Internet without NATs, routing is based on the destination address and each packet is routed independently.

4.2.2 Internet Protocol version 6 (IPv6)

To solve the problems inherited in IPv4, the IETF began to develop IPv6 during the first part of 1990s. IPv6 provides additional functionalities that include enhanced addressing capabilities, built-in security and the integrity of communication. IPv6 is the long term solution to replace IPv4. Figure 13 shows IPv6 packet header.

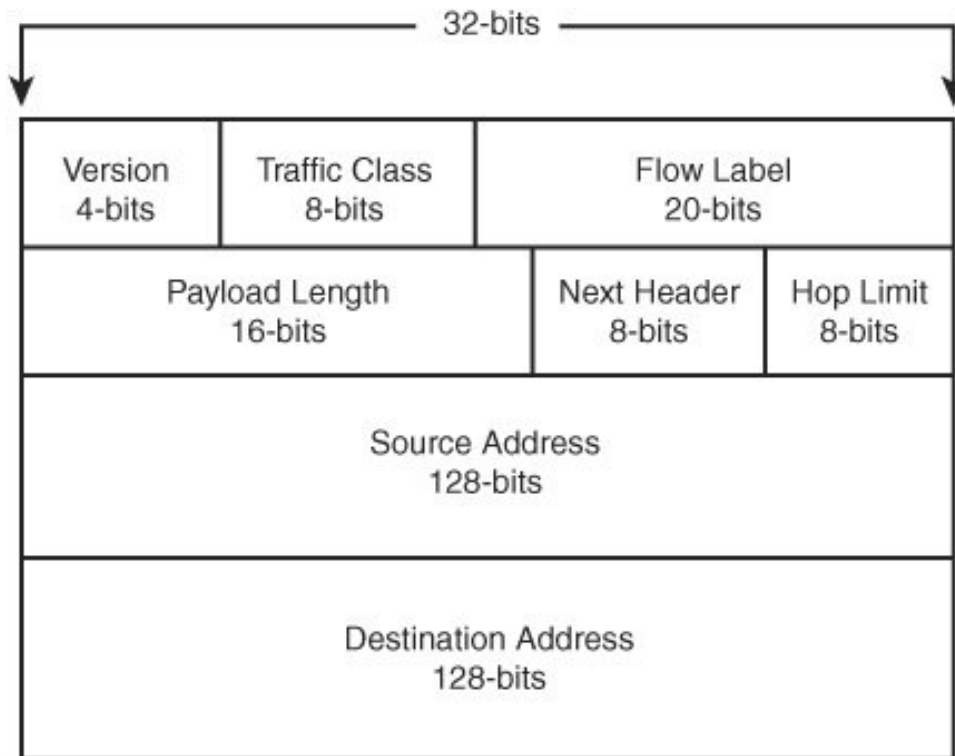


Figure 13: IPv6 packet header

4.3 Transport protocols

The transport layer has two protocols to facilitate the transfer of data packets: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP and UDP protocols provide a mechanism that can differentiate applications running on the same host through the use of port numbers. In normal cases, there are numerous applications running concurrently on a host and the TCP protocol helps decide which application uses which port.

4.3.1 Transmission Control Protocol (TCP)

The Transmission Control Protocol is connection oriented meaning that it establishes a connection before any transmission is carried. This protocol provides functionalities such as reliability, congestion and flow control. TCP supports several fields for error detection (the function that identifies receiving packets), acknowledgement (ACK number that sends information to a sender that the packet was successfully delivered), timers and the other header fields. Figure 14 presents the TCP header.

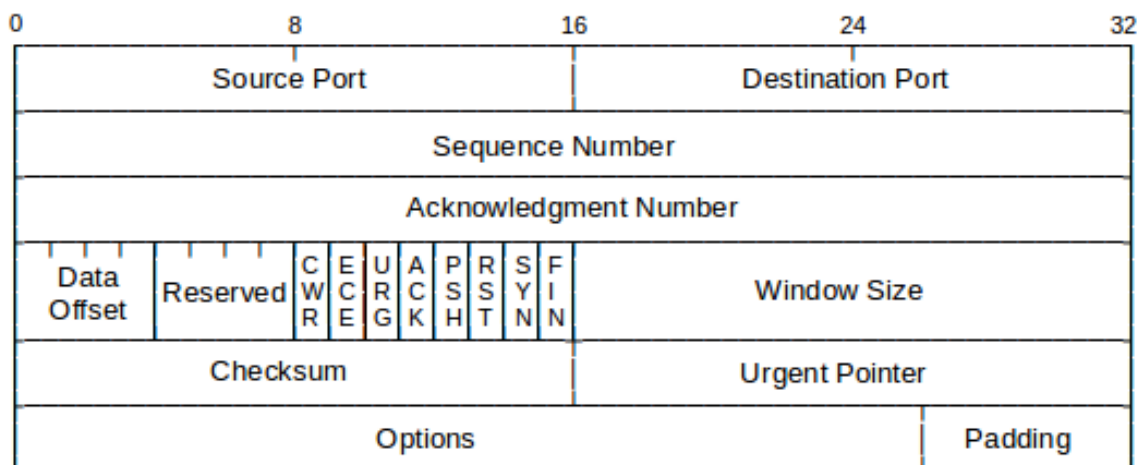


Figure 14: TCP Header

Source and Destination Port: is a numerical value that indicates the source and destination port of a service or application.

Sequence Number and Acknowledge Number: ensure the reliability of data transmission. The Sequence Number field is a 32 bit number used to keep track of number of the bytes sent. The value of the sequence number indicates the location of a single packet in the packet stream which has arrived. While the acknowledgment number is also a 32 bit number indicates the next sequence number that the sending device is expecting from the receiving device. For example, if the value in the acknowledgment field is 1201, this means that all datagrams that were received are less than or equal to 1200.

Data Offset: field sets the number of 32-bit words in the TCP header. This field indicates where the data begins.

Reserved field: is 6 bits field reserved for future purposes. This field must be set to zero.

Window field: is 16-bit, which corresponds to the number of bytes indicated in the acknowledgment field which the source of this segment is willing to accept.

Checksum field: is a mathematical number generated by the protocol sender to help the receiver to identify messages that are corrupted or tampered.

Urgent Pointer: is a 16 bits field which communicates the position of an urgent data by giving its offset in relation to the sequence number. The pointer should point to the next byte following the urgent data. This field is only interpreted when the URG Flag is set to one and as soon as this byte is received, the TCP stack must send the data to the application.

Options : fields are generally a multiple of 8 bits in length and can occupy space at the end of the TCP header. All options are taken into account by the Checksum. An option parameter always starts on a new byte. Two standard formats are defined for the options:

- Case 1 - One-byte option.

- Case 2 - Option-kind octet, option-length octet, option-data octet.

The option length takes into account the option-kind, the option-length itself and all option-data octets.

4.3.2 User Datagram Protocol (UDP)

Unlike TCP, UDP is connectionless protocol i.e., there is no pre-connection procedure for sending data, and there is no guarantee of delivery of a datagram to its destination. The arrival order of the datagrams may differ from the sending order. Figure 15 shows that UDP has simple header compared to TCP. It is not a reliable protocol because there is no acknowledgement flag. But, UDP is faster than TCP since it does not need to set up a connection before data can be transferred. However, checksum, source port and destination port fields work same as in TCP and there is no need to acknowledge the receipt of data [33].

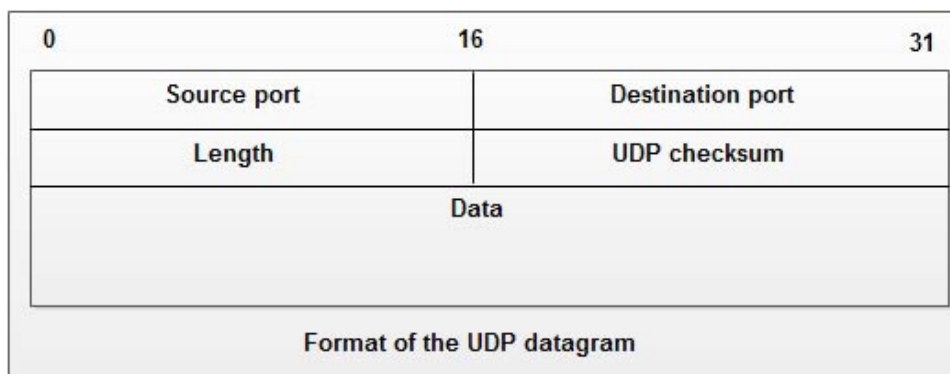


Figure 15: UDP Header

Well known ports

These port numbers range from 0 to 1023; they are mainly used by system processes in our case the Linux operating system. In this system, a process has to execute on superuser privileges to be able to bind a network socket to an IP address using a well-known port [33].

Ephemeral port

This is a short-lived transport protocol port for Internet Protocol (IP) communications. The port is allocated automatically from a predefined range by the IP stack software. An ephemeral port is used by the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or the Stream Control Transmission Protocol (SCTP) as the port assignment for the client end of a client-server communication to a well-known port on a server [33].

Source Port/Destination port

The source and destination port numbers are used to define the services that should take place on the local or remote hosts. In TCP/UDP data communications, it is the responsibility of the host to provide the destination and the source port number. The destination port number will allow the host to select the device that it requires while the source port is provided to the host server or in most cases the internet server for it to give feedback to the correct session initiated by the other side [33]. The use of source and destination ports will allow the applications to know which ports to use.

4.4 Domain Name System (DNS)

Domain Name System (DNS) protocol was developed to translate strings into numbers. Humans are not good at remembering numbers, e.g, one cannot remember all the contact numbers in one's contact list. In the same way, we cannot memorize the IP addresses of the websites. In the Internet, DNS translates the name, for instance, 'aalto.fi' to IP address '193.29.10.1', to which a client can then send the request for the web page. DNS consists of three major components: Domain Name Space, Name Server, and Resolver.

4.4.1 Domain Name Space

Domain Name Space has a structure of an inverted tree. The root of the tree (root domain) is expanded by children (Top-level domain), which then contain many levels of subdomains. Each node has a label of up to 63 characters. The set of domain names consists of an inverted tree where each node is separated from the next by a point ("."). The Fully Qualified Domain Name (FQDN) comprises of domain names of a lowest branch in the tree to the top level domain including all the branches separated by a ".". FQDN is used to uniquely identify a machine on the network of networks.

4.4.2 Name Server

A name server performs the server side operation in the DNS architecture. A name server is a machine that has complete information about a zone of domain name space that it is serving. DNS consists of two type name servers which are primary and secondary name servers. A primary master server or master is a program that creates, maintains and updates the information about the zone in the zone file on its host. A secondary master server or slave also has an authoritative role in the zone and it saves the identical information from the master.

4.4.3 Resolver

Similarly, a resolver acts as a client side in the DNS architecture. Each host has been configured to send its DNS name queries to at least one DNS resolver. The resolver performs various operations such as:

- Querying a name server for specific domain name and record type.
- Interpreting the reply which may contain a record or an error.
- Responding to the ‘host’ that requested DNS resolution.

4.5 Network Address Translation (NAT)

NAT was introduced as one of the short-term solutions to the address depletion problem of IPv4 addresses, until the long-term solution with enhanced address spaces is deployed [31]. NAT is a device or system in a router that allows a private network to use private IP addresses for local communication, and a set of public IP addresses to reach the public Internet. The private IP addresses can be reused in any local network, and these local addresses are only locally unique while the public addresses are globally unique. Besides the importance of these properties of NAT, it also hides the private network from the public Internet.

4.5.1 How NAT works

A NAT acts as a bridge between private network and Internet. It has a translation table that manages the connection state information which consists of information such as the protocol used for the communication, source local IP address, source local port, public IP address and public port. When a client inside local network attempts to connect to a server in public network, it transmits IP packets to that server. In order for the packet to get to its destination, it passes through the NAT gateway. The NAT device changes the local source IP address and the local source port of that machine with the public IP address and port number and saves the connection state in the translation table. Next, the device sends the changed packets to the required server. In the backward direction, the server responds to the public address of the NAT. The NAT at the boundary of the private network maps the modified address to the local address and forwards packets to the private host.

NAT devices can be classified into: basic NAT and Network Address Port Translation (NAPT). The first just performs IP-based translation services while the latter makes use of port numbers as well to perform the address translation [34].

4.5.2 Problem with NAT

Assuming host-A in a local network behind NAT starts communication with host-B that resides in the public network, the NAT device translates the private IP address and the port of the host-A to the public address, public port and stores them in the translation table. After recording information, the packets are sent out to the destination. In the opposite direction, host-B sends replies to the NAT, which applies the mapping before forwarding the packets to host-A.

In contrast, if host-B wants to initiate the communication, the packet will be discarded because there is no connection state in the NAT table. This issue is referred

to as the “Reachability Problem”, which prevents a host on the public network from being able to reach a host behind a NAT [31].

The reachability issue presented by NAT influences different protocols and applications in the public network. For instance, the protocol such as SIP has the IP addresses in their payloads for the establishment of the connection. This is not compatible with NAT. The reason for that is that NAT does not work above layer 4 and therefore it cannot alter the information in the protocol payload. Furthermore, NAT reachability issue affects the peer-to-peer applications since they required bidirectional connectivity.

To solve the reachability problem, several NAT transversal protocols such as STUN, TURN, and ICE have been developed [35][36]. The downside of this transversal solutions to cope with the “Reachability Problem” is that first establishing reachability requires a lot of application layer messaging, such messaging consumes time and finally the method requires keep-alive signaling in order to avoid expiration of NAT binding. This method does not scale well to mobile devices.

5 Customer Edge Switching

This chapter will mainly focus on explaining the concept of Customer Edge Switching (CES). An overview of CES architecture will be presented. Finally, the chapter will conclude with the discussion of the protocol used in CES.

5.1 Motivation

The rapid increased in the number of users, servers and connected devices on the Internet have caused the shortage of IPv4 addresses. The adoption of NAT slowed down the exhaustion of IPv4 by enabling several hosts in the private network to share or reuse set of public IP address. The deployment of NAT on the Internet alleviated the IPv4 address exhaustion problem, but introduced “Reachability Problem”. This problem prevents a host on the Internet from being able to reach another host behind a NAT. In order to solve this NAT reachability problem, several NAT traversal solution such as TURN and STUN have been developed [35][36].

The downside of this NAT traversal methods is that they do not scale well with mobile devices because the keep-alive signaling is required for mobiles phones to avoid expiration of the NAT binding, which consumes the mobile battery. Customer Edge Switching has been proposed and implemented at the COMNET department of Aalto University to replace NAT device and to overcome the problems introduced by NAT devices. CES makes use of global unique domain names to identify end-hosts and afterward uses local or public IP address to address an end hosts. The aim of CES is to solve the reachability problem and to create trust between customer networks and provide end-to-end connectivity as well as to improve the security on the Internet. Additionally, CES is seen as a suitable solution for mobile devices, since it does not require any keep-alive mechanism.

5.2 Architecture

CES architecture classifies Internet into: Customer Network (CN) and Service Provider Network (SPN). Several hosts may be connected to different CNs; each of these CNs is independent of each other and has at least one CES device. The CES device may have one or more interfaces that provide connectivity between CN and SPN networks. A CES device contains a firewall, a pool of private IP addresses which is referred to as a proxy address and a pool of public IP addresses for NAT like operations.

CES uses IDs for end hosts or services identification. These IDs may be provided by the service provider or internal program of the CES device, or generated from unique host name by applying a hash function. CES devices uses proxy address from the available pool of addresses to represent the remote user in the private network technology. The CES device residing at the network boundary contains a translation table which manages connection state information that allows packets forwarding between sending and receiving nodes [37]. Figure 16 illustrates the architecture of CES.

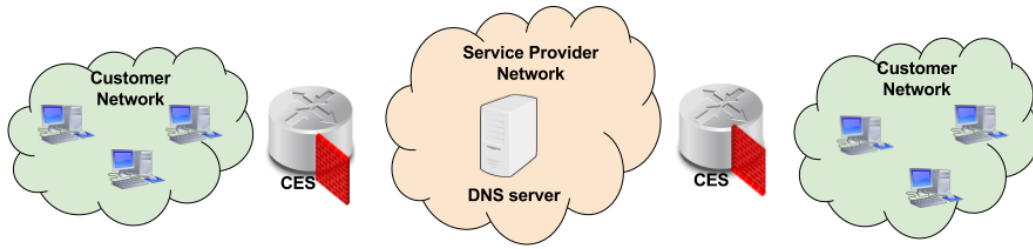


Figure 16: CES architecture

5.3 Communication in CES

The Customer Edge Switching mainly relies on DNS, any communication between hosts that are behind CES is triggered by the DNS name resolution. A valid domain resolution operation leads to a legitimate state in CES, acquiring a proxy address and thereafter, forwarding of the data packets. Communication in CES can be classified into three categories: Inter-CES communications, Intra-CES communication and packet forwarding across PRGW.

5.3.1 Inter-CES communication

Inter-CES communication occurs when the two hosts communicating are behind different CES devices. The source starts by performing DNS NAPTR resolution with a specific end goal to identify the CES-ID that hosts the destination domain. This is followed by both sender and receiver CES completing the connection establishment procedure based on the host admission policies. A connection state is created in each CES device after a successful policy negotiation occurred, where CES acts as the gateway to the remote host that uses a local proxy address. Next, the DNS response that carries the destination proxy address is sent to the source, and both parties send to each other using the states created in CES devices [37].

As shown in the Figure 17, when Host-B that is located behind CES-B wants to communicate with Host-A that is located behind CES-A, Host-B begins by sending a DNS query to Host-A. Since Host-A and Host-B are in the different private network, therefore CES-B sends the DNS query to the DNS server which then further forwards it to the DNS server that is situated in CES-A depending on the DNS server NS resource record. The DNS response from CES-A transports both the routing locator (RLOC) and CES-ID information corresponding to the destination host.

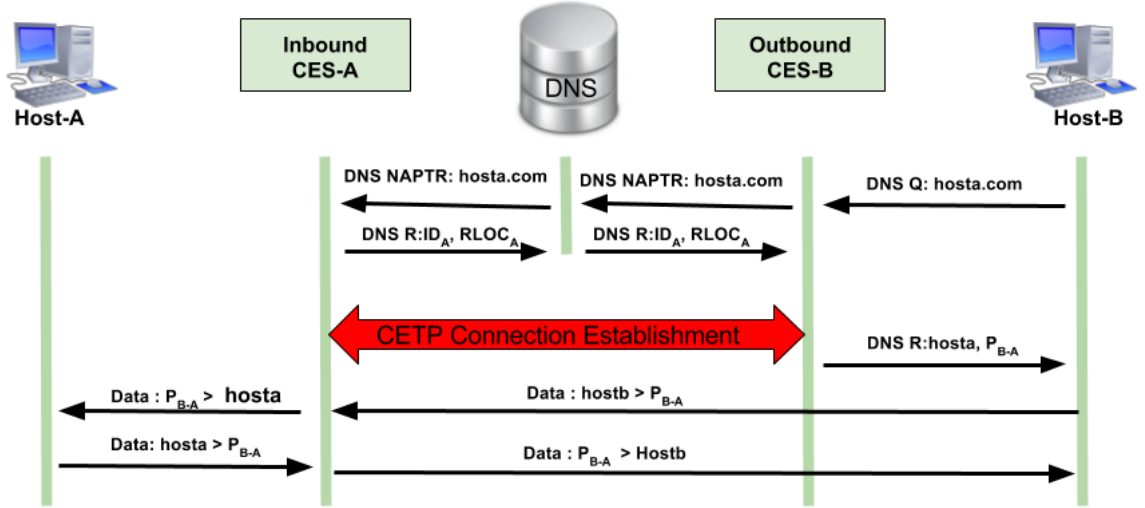


Figure 17: Inter-CES communication

The outbound CES (oCES) identifies the remote CES based on the identifier in the DNS Naming Query Pointer (NAPTR) response, and begins the connection establishment procedure using CETP protocol to negotiate admission policies with inbound CES (iCES). When the policy negotiation process between the two CES devices completes, the DNS response from the local CES conveys the proxy address of the destination to the source, and creates a binding between the local and public IP addresses. After establishing the connection between CES, source and destination hosts can send packets to each other using CES devices.

5.3.2 Intra-CES communication

Intra-CES communication occurs when the two hosts communicating are behind the same CES device. Unlike Inter-CES communication, the CETP policy negotiation happens locally. However, since all the packets pass through the CES device, the direct communication between the hosts is restricted.

An example of such communication can be explained as, Host-A and Host-B that are located behind the same CES want to communicate. Host-A initiates the communication by issuing the DNS query for Host-B. The CES allocates IP addresses for both Host-A and Host-B. The addresses are used to create a mapping in order for the hosts to communicate via proxy addresses. After this, the CES sends DNS response to the source with allocated proxy information that will be used to send packets to the destination [37]. This model works even in the case that Host-A and Host-B are in separate private address spaces.

5.3.3 Packet forwarding in PRGW

Another type of CES communication is known as Private Realm Gateway (PRGW), which allows a legacy host residing on the public Internet to connect to a host behind a CES device and vice-versa.

In addition, PRGW also serves as an authoritative DNS name server for the domain hosted in the local network. This allows PRGW to be able to resolve a DNS request received from a host on the public Internet. A legacy host residing on the public Internet just requires to send a DNS query to access the destination domain behind the PRGW, after receiving the DNS query, PRGW reserves an address from the circular pool.

Figure 18 depicts the detailed operation of PRGW for an inbound connection where Host-P on the public Internet sends a DNS query to resolve the name of Host-A, which the DNS server forwards to the CES where the Host-A is located. The CES keeps the next available address from its circular pool and sends back the DNS response transporting the address reserved to the Host-P. When Host-P receives the DNS response, it sends the data packet to the address reserved for the destination. The data packet is then forwarded to Host-A after mapping the public address to local address. In the backward direction, the Host-A response is sent to Host-P after the mapping of its private address to public address at the PRGW. More detailed on PRGW is described in [37].

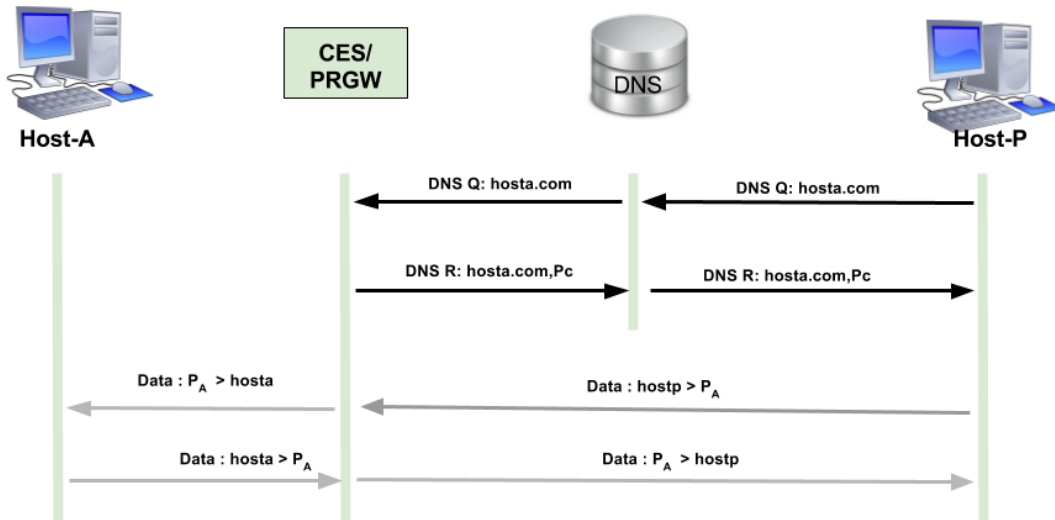


Figure 18: Communication in PRGW for inbound connection

5.4 Customer Edge Traversal Protocol (CETP)

Customer Edge Traversal Protocol (CETP) is basically a tunneling protocol providing communication between CES devices. The protocol has been developed gradually

since it was first implemented by Pahlavan. However, CETP is still in development stage and may be subjected to more changes later on. In this section, an overview of the present CETP is discussed. The details about the first version of CETP protocol is described at [38].

CETP is designed to facilitate the transmission of packets between different CES nodes while still carrying the source and destination IDs. When CETP connection establishment is successful, an edge-to-edge connection is created where session identifiers are used to identify different user connections. In addition, CETP is a protocol that controls signaling between CES devices, it can be described as an edge to edge protocol for tunneling packets from one customer network to another customer network, in this case, each network is given its own local address space.

CETP is part of the Internet Trust Framework (ITF) which aims to discourage the business of creating unwanted traffic by making it unprofitable [39]. CETP protocol provides both tunneling and signaling between customer networks, this helps to implement the idea of collaborative firewalls. The protocol can be implemented to solve issues such as isolation of customer and core network, trust improvement in customer network gives the receiver edge tools to combat spoofing before the flow admission to the receiver.

A CETP packet consists of a mandatory 32-bit fixed header, source, destination session tags and a set of control TLVs. CETP packets are agnostic to the underlying technology and can be transported over a number of network and transport protocols, i.e. Ethernet, IPv4, IPv6, TCP or UDP [40]. The control TLVs consist of three segments which are the type, length, and value. The value of length bytes is always padded to a 4 bytes boundary for faster processing with a minimal impact of overhead. The operation field carries query, response or information that notifies the remote end about the type of operation. A policy in CETP consists of three different vectors: offer, requirement and available. Each of these vectors store the TLV components used to define a specific policy.

5.5 Conclusion and research questions

Having briefly described the concept of Customer Edge Switching it is time to summarize and recap our research questions.

CES is a new architecture for Internet communications that prior to communication from host A to host B allows establishing a chain of trust between the hosts on a given level of assurance. The level of assurance is defined by the respective communications security policies of the sender and the receiver. The role of the CES nodes is to establish long lasting identities for the hosts so that it is always possible to immediately trace back a misbehaving or malicious host once it has been identified in the remote network. Moreover, a CES node can push the responsibility for containing the harm caused by a malicious host to the CES node serving that host.

With this background in mind, it is important to study how is it feasible to create policies in the case of smart phones and mobile broadband services to the mobile users? Is it feasible that each host will have a separate policy that exactly

reflects what the host is expecting to see in the Internet traffic and thus eliminate all unexpected traffic from entering the air interface or waking up the mobile device?

We will assume that the end users have limited technical skills and it would be infeasible to require them to create the policies from scratch. Instead, the policy creation should be as automatic as we can make it. The user should deal with the policies on a level of natural user decisions such as:

- Do I want to receive calls or messages at night?
- Do I want to block some App entirely for the time being?
- Do I want to adopt the default policy for a specif App?

This thesis studies the question of policy creation in Android and Linux based devices. The idea is to glean as much information as possible from the devices and store that to a database for further processing by cloud based policy management System (PMS). The interface to the cloud based PMS was developed during this thesis but the actual further processing is out of scope in this thesis.

The Policy Apps developed in this thesis aim to get full information of all Internet connected applications and stored it in User Policy Database (UPD). The stored information are then used by the end user to set her own policies that are later pushed to the PMS. A policy is valid when it is stored in the PMS.

6 Implementation and Evaluation

This chapter first gives an overview of the overall architecture of the CES policy tools. Next, we present Policy Creation and Bootstrapping System (PCBS) implemented in this thesis with its various components. Finally, the tutorial on how to use the developed system is described.

In an attempt to overcome the shortcomings of Network Address translation (NAT) and protect the interest of the served host over the Internet, CES aims to have tools such as PCBS and Policy Management System (PMS) to allow the end user to set the policy for her device. Figure 19 represents the overall architecture of the CES policy tools. The PCBS is sitting between the end-user device and PMS. The PCBS populates the PMS which also is populating the CES network-based solution. The separation of PCBS from the PMS can be explained as a security measure e.g. an attack or failure of PCBS will not affect policies that are already in the PMS.

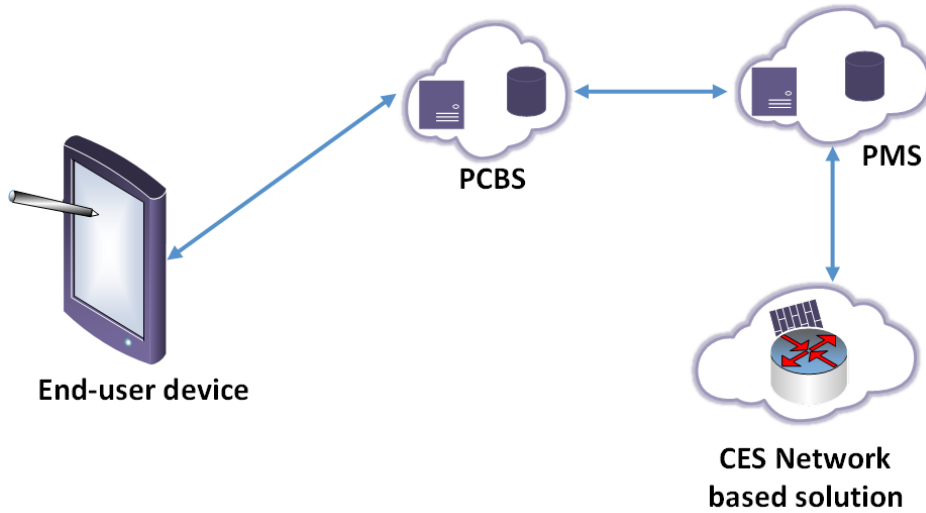


Figure 19: Overall architecture of the CES policy Tools

In this thesis, we developed PCBS which provides the end users with the ability of setting her own policies for the applications installed on the device and thus control who she communicates with. The solution architecture used in this thesis work comprises of four different components listed below as shown in Figure 20.

1. User Policy Agent (UPA) comprising of Policy App and Policy Interface
2. User Policy Server (UPS)
3. User Policy Database (UPD)
4. Policy Management System (PMS) client

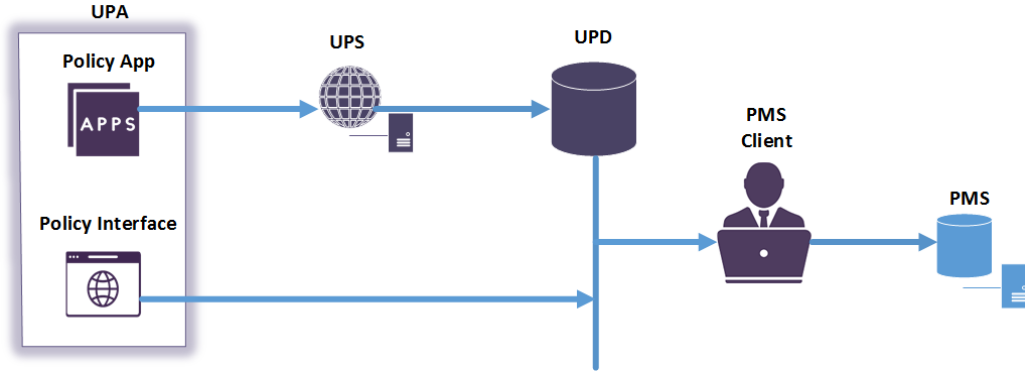


Figure 20: Architecture of PCBS

6.1 User Policy Agent (UPA)

A User Policy Agent comprises of a Policy App and a Policy Interface. The Policy App is an application running on the end user device and its function is to collect all necessary information about the active apps installed on the device. In this thesis, we define active apps as Internet connected apps which include any client server applications that are currently running on user's device. Client server apps are any applications that listen and can send/receive traffic from/to outside a network. The gathered necessary information by the Policy App includes: FQDN, Application name, IP address, Application Port number, remote IP address, remote port number of the application server, protocol used and the status of the communication. Figure 21 shows the data flow between Policy App, UPS and the User Policy Database (UPD). The Policy App uses a REST API to send the collected information or payload to the User Policy Server (UPS) which performs operations such as processing, storing and deleting information in the User Policy Database (UPD). The UPS sends replies to the Policy App for both successful and unsuccessful operation. The unsuccessful operation occurs when the server is down or the user does not send using required format of the data.

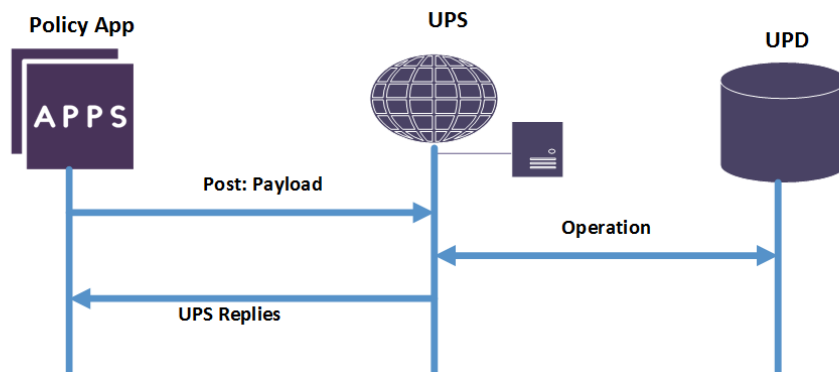


Figure 21: The data flow between PCBS components

The author of this thesis developed Policy App which is available for two platforms, Android and Linux operating system. The Android and Linux UPAs are described in the following section.

6.1.1 Linux Policy App

The Linux Policy App is an application designed for Linux systems and runs on python 2.7. This app collects network information on intalled Apps. The application works similarly as the Linux netstat command. It reads information about the running applications from the proc/net/protocol pseudo-file system based on the protocol, which can be TCP, TCP6, UDP or UDP6. The read function returns an array of a dump of the active protocol socket table in hexadecimal format which is later converted to decimal. An extract of the output in hexadecimal format is shown in Figure 22.

sl	local_address	rem_address	st	tx_queue	rx_queue	tr	rexmits	tm->when	uid	inode
0:	0100007F:1F40	00000000:0000	0A	00000000:00000000		00:00000000	00000000	1000	20 4 1 18 -1	
1:	0100007F:0CEA	00000000:0000	0A	00000000:00000000		00:00000000	00000000	121	100 0 0 10 0	
2:	00000000:9EDD	00000000:0000	07	00000000:00000000		00:00000000	00000000	111	0	

Figure 22: The output format of the Proc/net/protocol pseudo file system

An explanation of the information headers is shown in Table 1.

Header	Description
sl	kernel hash slot for the socket
local_address	the pair of local address and local port number
rem_address	the pair of remote address and remote port number
St	The status of the current open socket of the application
tx_queue	kernel memory usage based on the outgoing data queue
rx_queue	kernel memory usage based on the incoming data queue
tr, tm->when, and rexmits	contains the internal information of the kernel socket state
UID	holds the UID of the application
Inode	describes the file directory

Table 1: Header of proc files

An application on a user device has life cycles or processes which include downloading, installing, using, upgrading, and deleting of the application as show in the Figure 23. Downloading is a process in which the end user gets the application file or program e.g APK in android from the author repository. Installing process is the act to make the downloaded program ready to execute on the device. Using process makes the application to perform a task. In this process there are two types of state which are active and idle. Active state is when the application is running e.g App is connected to the Internet and can send/receive traffic. Idle is when the installed application is not running. Upgrading process is when the application is being updated by the new version on the device. Deleting is the process that removes the installed application completely from the user device. In our case the proc/net/protocol retrieves information when an application is in active state.

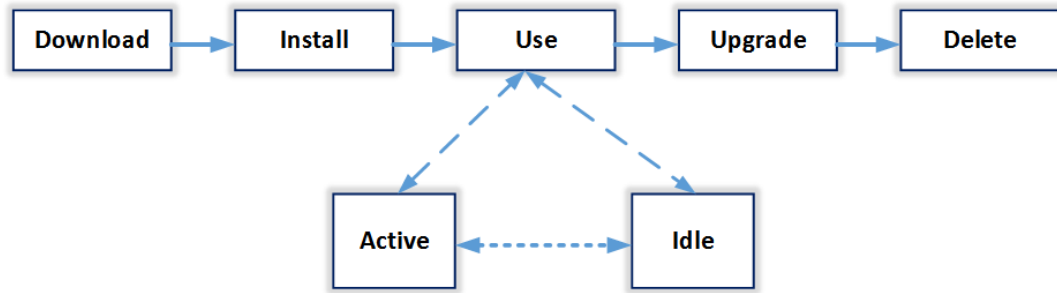


Figure 23: Application life cycle

The header of the proc/net/protocol output has nine fields which define full information about a particular active application. The important field required from the proc file execution are: local_address, rem_address, st, UID and Inode. This data for instance in Figure 22 are converted to decimal format as shown below in the Figure 24.

After conversion, the data are saved in json format i.e, { 'local_port': '46819', 'local_ip': '10.249.201.134', 'protocol': 'tcp', 'remote_ip': '192.168.43.208', 'app_name': 'Policy App', 'status': 'LISTEN', 'remote_port': '8000', 'fqdn': 'CES App' }. These information is later used by the end user to set policies for that particular application.

```
[ 'tcp', 'MySQL', '127.0.0.1', '3306', '0.0.0.0', '0', 'LISTEN', None, None]
[ 'tcp', 'Userx', '127.0.0.1', '8000', '0.0.0.0', '0', 'LISTEN', '11862', '/usr/bin/python2.7']
[ 'udp6', 'avahi', '0.0.0.0', '40669', '0.0.0.0', '0', 'CLOSE', None, None]
```

Figure 24: The hexadecimal array converted to decimal

6.1.2 Android Policy App

Android Policy App collects information on intalled Apps as well as the active apps network information.

As stated in section 2.3.2, Android is a Linux-based mobile operating system. To gather the network information about active apps on Android smartphones, we decided to make use of proc file system as done for Linux Policy App. In this Android Policy App, after executing the `proc/net/protocol` command, the application name and signature of the application is found by passing “UID” to the Android Packet Manager.

As a result we have a UPA mobile application that is available for Android smartphones. It lists all application information as well as connections types on the mobile device. The application contains 3 tabs: All Apps, Connection Logs, and UPI. After launching the app, the first tab displays the list of Installed Apps. As shown in Figure 25, the tab shows information about applications on the device such as Application Name, Version, User Id (UID), signature and Package Name.

The second tab on the UPA shows Connection Logs. This tab lists information about all Internet connected applications on the device using TCP (TCP4 or TCP6) and UDP (UDP4 and UDP6) connections. The information displayed includes, Source IP, Source ports, Remote IP, Remote port, App name, the protocol used, and the state of the connection. Additionally, this tab has a button that allows the user to send the information of apps to the UPS for setting policies. Figure 26 shows the output of connection logs tab of the application.

Finally, as shown in Figure 27, the last tab displays the Policy Interface which accesses web pages in the Policy App to allow the user to directly set his policies.


Policy App		
ALL APPS	CONNECTION LOGS	POLICY TOOL
	Viber 7.5.5.8 120291 com.viber.voip 10157	
	Codes 1.5.1 15100 com.nordea.mobiletoken 10143	
	WhatsApp 2.17.323 451987 com.whatsapp 10152	
	PicsArt 9.18.3 993000183 com.picsart.studio 10166	
	Voice Recorder 20.1.83-92 2018392200 com.sec.android.app.voicenote 10053	

Figure 25: All Apps tab





Policy App			
ALL APPS	CONNECTION LOGS	POLICY TOOL	
	10.243.117.192:37605		
tcp	52.0.253.40:4244		
	Viber		ESTABLISHED
	10.243.117.192:36038		
tcp	192.12.31.78:5228		
	imo		ESTABLISHED
	10.243.117.192:54782		
tcp	130.233.0.9:443		
	Chrome		ESTABLISHED
	127.0.0.1:51826		
tcp6	:::0		
	Messenger		LISTEN
	:::53		
tcp6	:::0		
			LISTEN
	127.0.0.1:48165		
tcp6	:::0		
SEND DATA			

Figure 26: Connection Logs tab

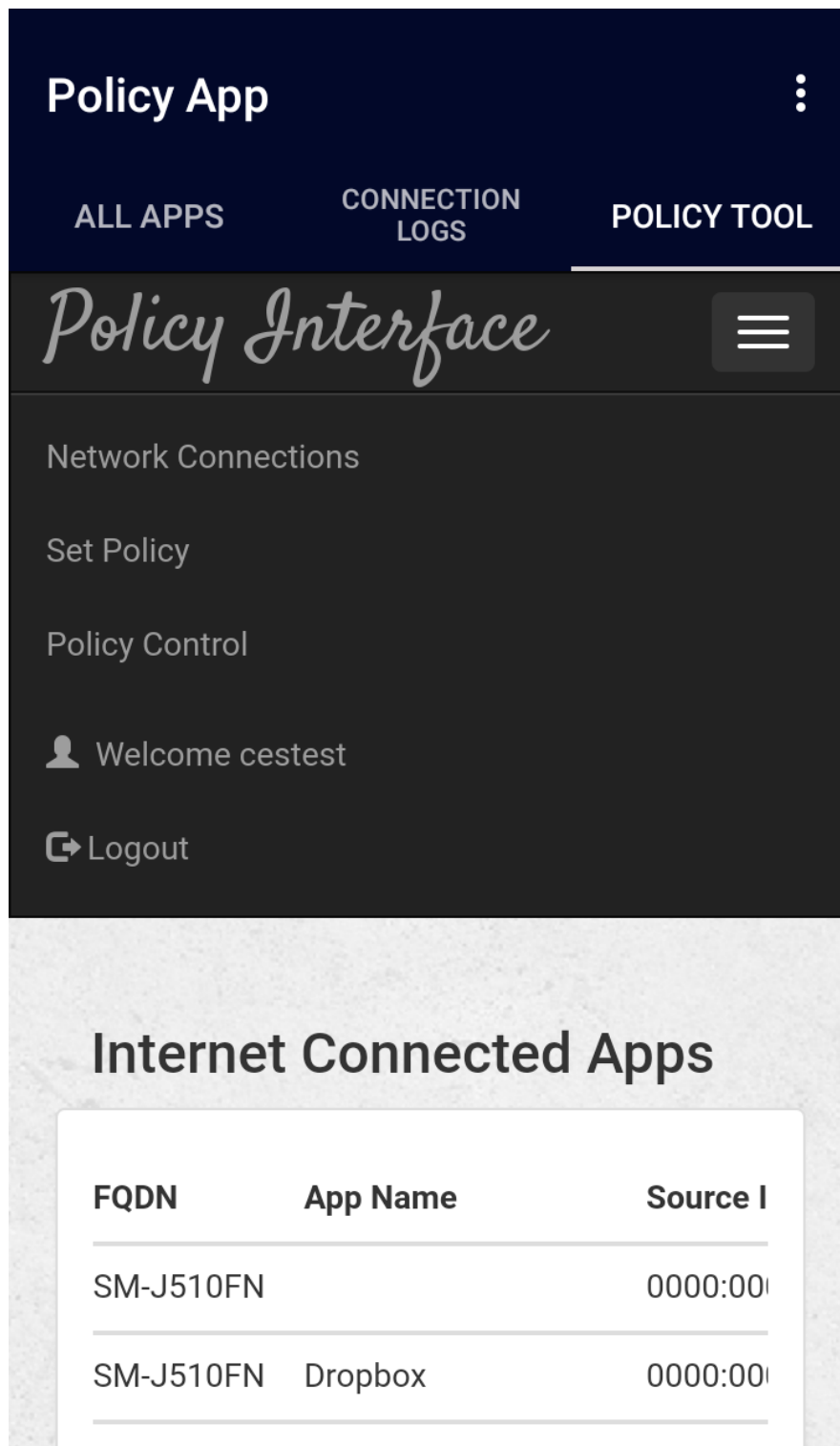


Figure 27: Policy Interface incorporated into Policy App

6.1.3 Policy Interface

The Policy Interface is a web interface allowing the end user to perform different policy management operations such as creating new policies, modifying an existing policy, deleting an existing policy, enabling and disabling policies. This tool provides a simple web-based interface for end users. The Policy Interface uses generated active apps information from the Policy App to set policies. It provides each user the ability to access information about all apps installed on their device as well as set their policies.

6.2 User Policy Database (UPD)

The UPD stores the information collected by the Policy App and Policy Interface. It consists of many tables for storing information about user policies. The tables includes, *HOST_APP*, *USER_CREATED_POLICIES*, *USER_DELETED_POLICIES*, *FAKE_APPS* and *SERVICE_PROVIDER_APPS*.

The *HOST_APP* table stores information about all Internet connected apps on the end user devices collected by the Policy App. In a case of Android Policy App we store only legitimate apps. We define legitimate apps as any application which signature matches with signatures that are in UPD. It contains 8 tuples of data including, FQDN of the host, application name, host IP address, application port number, remote IP address of the app server, remote port of the app server, the protocol used and the status of the communication.

The *USER_CREATED_POLICIES* table stores newly created or updated policies by the user. It contains the same information as the *HOST_APP* table in addition to five other tuples such as, Network Service, Traffic Direction, Action to Perform, Schedule Start time, and Schedule end time of the policy. The network service contains a list of protocol names which the user desires to apply for a particular application. Traffic direction states weather the traffic is incoming(INGRESS) or outgoing(EGRESS), the field has two selectable values which are *INGRESS* and *EGRESS* traffic. The action to perform field has two selectable values which states weather the user wants to *DROP* or *ALLOW* a packet from a particular application. The schedule start and schedule end time define the interval time a policy is valid. This table has also two boolean fields which are *new_policy* and *update_policy* which differentiate between update and creation of the new policy. When policy is created the value of *new_policy* is set to false until when the policy has been saved in PMS then it is changed to true. The *update_policy* field works similarly as *new_policy* field but when the policy is updated.

The *USER_DELETED_POLICIES* stores user deleted policies. This table has the same entries as the *USER_CREATED_POLICIES* table. It has an additional field that stores the status of the deleted policy. It is a Boolean value showing if the deleted policy has been deleted in the PMS.

The *SERVICE_PROVIDER_APPS* table contains information about all known legitimate apps. The information in this table includes, the signature of the app or hashCode, name of the app and the version of the app. Newly installed apps on the

user's device are compared to the contents of this table to verify the legitimacy of the apps. In this thesis, we assume in a case of Android device that this table is populated with all validated Android Apps by the service provider.

The *FAKE_APPS* table stores all the applications whose signature key does not exist in *SERVICE_PROVIDER_APPS* table and are known to be malicious apps. This table will tell PMS to not forward any traffic to any application saved in this table.

The UPD tables ensure that the stored informations are well defined and the policy in the PMS will not contain any malicious policies or malicious apps. In this way, the end user device policies can perform as expected.

6.3 User Policy Server (UPS)

The User Policy Server is an HTTP server running on port 10001. The task of the UPS includes receiving the user apps information from Policy App, validating the data and performing a database operation i.e. storage. The UPS resides between the Policy Agent and the UPD where it has a TCP connection with the MySQL server. When it receives the apps information from the Policy App, it checks whether the information is in the required format using various functions such as IP validation check (validity of IP address), Protocol check (Validity port range), and SQL injection check (the information does not contains any SQL injection attacks).

If the information from Policy App passes the validity check, in case of an Android device, the signature of the app is checked against the *SERVICE_PROVIDER_APPS* table. If the app's signature matches one in the *SERVICE_PROVIDER_APPS* table, the app is stored in *HOST_APP* table. Otherwise it is stored in *FAKE_APPS* table. In this way, the UPS prevents malicious users and apps from infusing malicious policies in the UPD. In addition, the UPS stores only the information about Internet connected apps and deletes the information from the UPD if the app is no longer running on the end user device. But, if the policy is already set by the user, that policy will be kept in the UPD. In case of Linux device the information are saved in *HOST_APP* table.

6.4 Policy Management System (PMS) client

The PMS Client acts as a client to the PMS. It's task is to request the policy data from the User Policy Database and forward the information via REST API to the PMS. The PMS client regularly polls the UPD for changes to policies such as post, update and delete of policies. In addition, the PMS client also updates the UPD tables when the storage to PMS is successful.

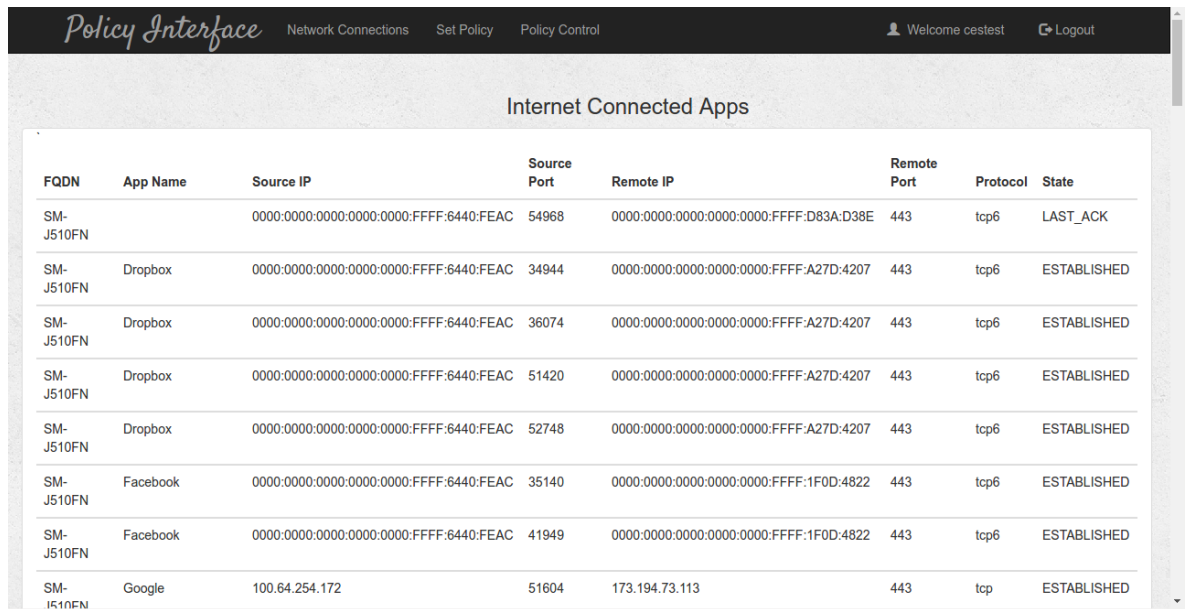
6.5 Implementation and Usage of Policy Interface

The Policy Interface was implemented using Python-Django, HTML, CSS and JavaScript. All new users are required to register before they can use the Policy Interface service. After registration, the user is able to create, update or delete their

policies. Because the majority of the users are subscribed to a particular Service Provider Network (SPN), they are given a unique FQDN by their respective SPN to authenticate themselves. The Policy Interface has several views for end users such as Front Page, Network Connection Information, Set Policy and Policy Control.

Front Page: This is the home page of the website where the user can find instruction how to use the tool such as creating, editing, deleting, enabling and disabling policies.

Network Connection Information: The Policy App sends information about apps currently running and this page displays the detailed view of apps information to the user. The displayed information includes FQDN of the host, Application name, the IP address of the host, port number of the host, remote IP address of the app server, remote port of the app server, the protocol used and the status of the communication. Figure 28 presents the web page that displays all user Internet connected apps.



FQDN	App Name	Source IP	Source Port	Remote IP	Remote Port	Protocol	State
SM-J510FN		0000:0000:0000:0000:0000:FFFF:6440:FEAC	54968	0000:0000:0000:0000:0000:FFFF:D83A:D38E	443	tcp6	LAST_ACK
SM-J510FN	Dropbox	0000:0000:0000:0000:0000:FFFF:6440:FEAC	34944	0000:0000:0000:0000:0000:FFFF:A27D:4207	443	tcp6	ESTABLISHED
SM-J510FN	Dropbox	0000:0000:0000:0000:0000:FFFF:6440:FEAC	36074	0000:0000:0000:0000:0000:FFFF:A27D:4207	443	tcp6	ESTABLISHED
SM-J510FN	Dropbox	0000:0000:0000:0000:0000:FFFF:6440:FEAC	51420	0000:0000:0000:0000:0000:FFFF:A27D:4207	443	tcp6	ESTABLISHED
SM-J510FN	Dropbox	0000:0000:0000:0000:0000:FFFF:6440:FEAC	52748	0000:0000:0000:0000:0000:FFFF:A27D:4207	443	tcp6	ESTABLISHED
SM-J510FN	Facebook	0000:0000:0000:0000:0000:FFFF:6440:FEAC	35140	0000:0000:0000:0000:0000:FFFF:1F0D:4822	443	tcp6	ESTABLISHED
SM-J510FN	Facebook	0000:0000:0000:0000:0000:FFFF:6440:FEAC	41949	0000:0000:0000:0000:0000:FFFF:1F0D:4822	443	tcp6	ESTABLISHED
SM-J510FN	Google	100.64.254.172	51604	173.194.73.113	443	tcp	ESTABLISHED

Figure 28: User apps informations

Set Policy: This page displays all Internet connected applications from the *HOST_APP* table, i.e. only client server application. The application have connection status which are either Listen or Established. This page enables the user to create policies for a specific application. As shown in Figure 29, the user can set policies for an application by clicking on the set policy button in the column for that application. The set policy page contains fields pre-filled information gathered from the Policy App. The user must fill the additional input fields such as Network Service, Traffic Direction, Action To Perform, Schedule Start and Schedule End Time. This input form filled in by the user constitutes the policy for the application and is then stored in the *USER_CREATED_POLICY* table in the UPD.

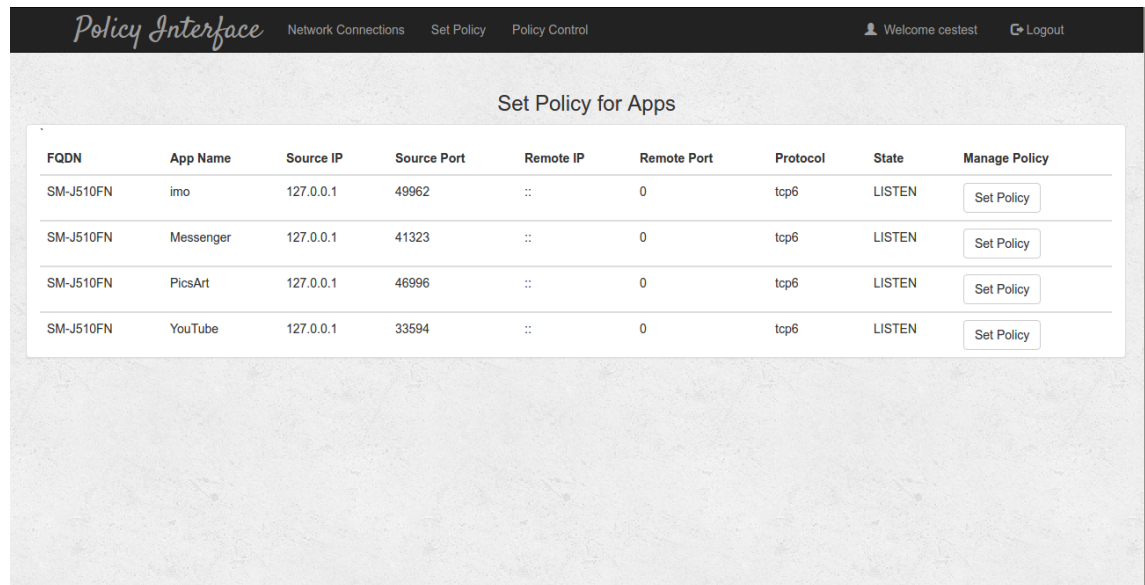
Policy Control: After the user creates policies, as shown in the Figure 30, this page has three buttons that allow users to edit, delete, enable, or disable a particular

policy.

Edit policy: The user can edit existing policies that he/she created previously by pressing the edit button.

Delete policy: The user can delete a specific policy that was created by pressing the delete button. When this button is clicked, the current policy is first saved in the *USER_DELETED_POLICY* table then later the delete request is sent by PMS client to PMS to delete the policies.

Enable and **Disable policy** buttons respectively allows the user to activate and deactivate a particular policy.



FQDN	App Name	Source IP	Source Port	Remote IP	Remote Port	Protocol	State	Manage Policy
SM-J510FN	imo	127.0.0.1	49962	::	0	tcp6	LISTEN	Set Policy
SM-J510FN	Messenger	127.0.0.1	41323	::	0	tcp6	LISTEN	Set Policy
SM-J510FN	PicsArt	127.0.0.1	46996	::	0	tcp6	LISTEN	Set Policy
SM-J510FN	YouTube	127.0.0.1	33594	::	0	tcp6	LISTEN	Set Policy

Figure 29: User set policies page

FQDN	App Name	Source IP	Source Port	Remote IP	Remote Port	Protocol	State	Services	Direction	Action	Manage Policy
SM-J510FN	Messenger	127.0.0.1	42063	::	0	tcp6	LISTEN	SCP	INGRESS	ALLOW	Edit
SM-J510FN	Fabebook	100.64.254.25	4132	193.64.24.25	443	tcp6	LISTEN	SCP	INGRESS	ALLOW	Edit
SM-J510FN	YouTube	127.0.0.1	33594	100.64.254.25	443	tcp6	LISTEN	SSH	INGRESS	DROP	Edit
SM-J510FN	PicsArt	127.0.0.1	46996	::	0	tcp6	LISTEN	SSH	EGRESS	ALLOW	Edit
SM-J510FN	YouTube	127.0.0.1	33594	::	0	tcp6	LISTEN	SSH	INGRESS	DROP	Edit
SM-J510FN	Whatsapp	100.64.254.25	3594	194.64.254.15	443	tcp6	LISTEN	SCP	INGRESS	ALLOW	Edit

Figure 30: User policies page

6.6 Usage of the PCBS

This section provides tutorial for new users how to use the PCBS. The tutorial is divided into two categories Linux and Android platforms.

6.6.1 Usage for Android user

1. Download Policy App APK from
"https://gitlab.cloud.mobiledsn.org/CES/policy_tools/tree/master/PolicyAPK"
2. Install APK
3. Start the app by clicking on Policy App icon
4. In Connection Logs tab, send Internet connected apps information to UPS by clicking *Send Data* button
5. On Policy Tool tab, the Policy Interface can be used in this tab after authentication. The usage of the Policy Interface can be found in section 6.5.

6.6.2 Usage for Linux user

This application run on python 2.7.

1. Download Policy App file from
"https://gitlab.cloud.mobiledsn.org/CES/policy_tools/tree/master/client"
2. Run the file using command Python "UbuntuApps.py start"

3. Copy-paste "http://100.64.254.25:8000/ces" to the browser
4. the Policy Interface can be used after authentication. The usage of the Policy Interface can be found in section [6.5](#).
5. Stop Policy App using command Python "UbuntuApps.py stop"

7 Performance testing

This chapter examines the performance of some components of the PCBS described in Chapter 6. The tested components include the Policy Apps and User Policy Server (UPS). Since the Policy Interface is developed in Python-Django we did not test it because the framework in its own has better performance. The main focus is on the User Policy Server. To measure performance of the UPS, load testing was carried out to determine how many users it can handle simultaneously and also find the crash point of the system.

7.1 Test tools

Since UPS at its core is an HTTP server, already mature HTTP load testing tools can be used to conduct load testing on it. There are several freeware tools available that can be used to test the performance of HTTP servers. In our case we choose to measure the performance of the UPS with *Apache Jmeter* and *httperf*. They are both open source server benchmarking tools which are commonly used to generate workloads to test an HTTP server. *Apache JMeter* is a pure Java application designed and developed by the Apache Software foundation. *JMeter* allows recreation of different scenarios with a number of simultaneous clients for various objectives such as identifying the system bottlenecks and resource issues. It is extensible and with the help of various plugins can be customized to fit different test scenarios. On the other hand, *httperf* is a simple command line tool for load testing and it works with both HTTP/1.0 and HTTP/1.1. This tool provides flexibility to generate several HTTP workloads. It monitors various performance metrics that are summarized in the form of statistics that are displayed at the end of a test run.

7.2 Test environment

The test environment comprises of two Linux virtual machines which are Fofana-VirtualBox and Ibrahima-VirtualBox and two Samsung smartphones Galaxy J5 and Galaxy S4 mini . The UPS was running from Fofana-VirtualBox and the client requests were executed on Ibrahima-VirtualBox. The UPS and the client request both under test make use of 2 cores on their respective machine Fofana-VirtualBox and Ibrahima-VirtualBox. The linux virtual machine specifications are given in Table 2 and the Android smartphones specification are given in Table 3.

Machine Names	Operating System	Processors	RAM	Internet Speed
Fofana-VirtualBox	Linux (64-bit)	2	4Gb	739.01 Mbit/s
Ibrahima-VirtualBox	Linux (64-bit)	2	4Gb	739.01 Mbit/s

Table 2: Test linux machine specifications

Android Phone	Android Version	Model Number	Memory
Galaxy J5	6.0.1	SM-J510FN	16Gb
Samsung S4 mini	4.4.2	GT-I9195	8Gb

Table 3: Test Android smartphones specifications

7.3 Test data

Since end user device may have many applications installed on the device, we assume that each end user device may run the average of 20 Internet connected applications when the Policy App runs. Therefore, we choose 20 as test data for the number of Internet connected apps to measure in our performance test. These test data are sent to UPS via post request in Json format.

7.4 Performance Test of User Policy Server

The UPS performance was tested using both the *httperf* and *Jmeter*. For measuring the server performance, we split the test into three different test scenarios which are described in the Figure 31. The test scenarios consist of test1, test2 and test3. In test1, the UPS receives a request from the client and without processing sends back a reply. Test2 determines the UPS processing times which includes the processing delay and User Policy Database(UPD) delays. Test3 measures the overall delay from Policy App to UPS, including transport delay and processing delay.

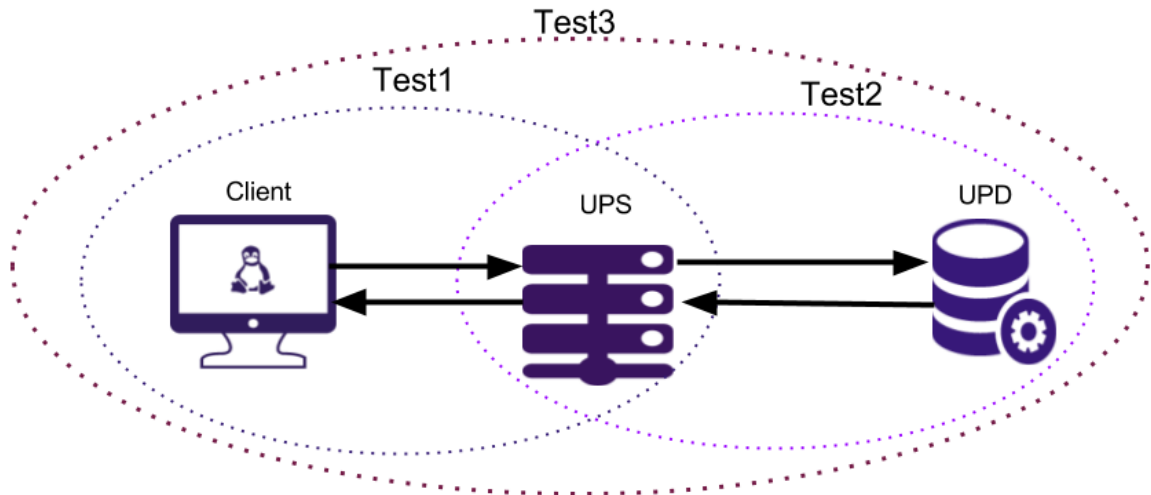


Figure 31: User Policy Server (UPS) test scenarios

7.4.1 Test1: Average number of replies between concurrent Policy Apps and UPS without processing

For measuring the UPS's performance, *httperf* was used as Policy App to send data to the UPS which is configured to send back reply without processing the request. The number of concurrent requests per second was increased till the UPS was saturated and UPS failure was reached. The term UPS failure means that not all http requests got a reply from the UPS. The results of test1 are shown in Figure 32. The graph was generated based on the raw data collected using several experiments for each data point on the y-axis. Figure 32 depicts the average number of requests sent by

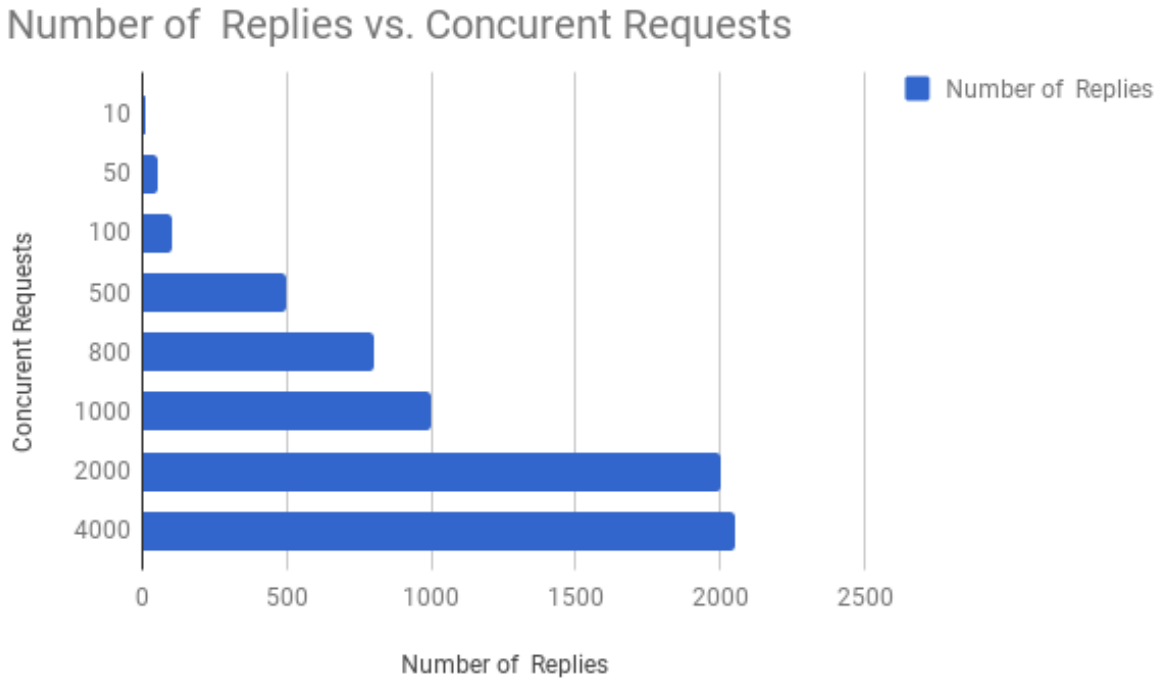


Figure 32: Average reply time versus number of concurrent requests

a client versus the corresponding average number of replies by the UPS. From the figure it is evident that the UPS can reply to a client request without failure till up to about 2000 concurrent requests per second. Up to 2000 concurrent requests, the average number of requests and replies are equal. This implies that the UPS can handle 2000 simultaneous clients per second when not processing the requests. As shown in the Figure 33 and the Table 4, the UPS gets saturated if there are more than 2000 user requests simultaneously per second, e.g. the UPS performance deteriorates and the number of errors, i.e. unreplied requests, increases.

Average reply time (ms) vs. Concurrent requests

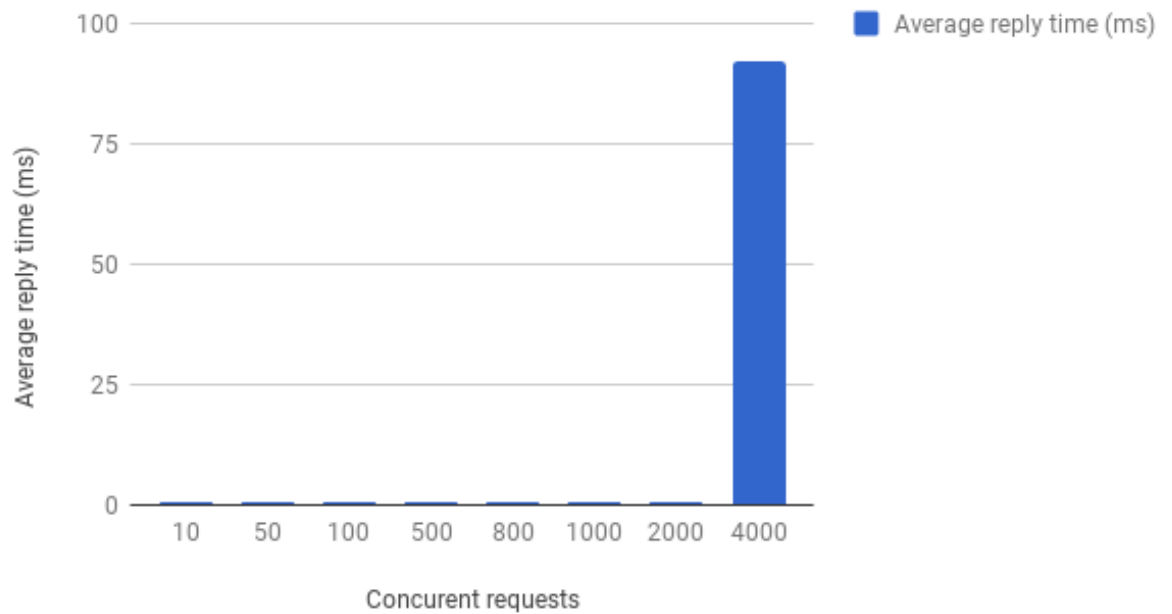


Figure 33: Average reply time versus number of concurrent requests

Concurrent requests	Average reply time per connection (ms)
10	0.5
50	0.5
100	0.5
1000	0.5
2000	0.5
4000	92

Table 4: Average reply time versus number of concurrent request

7.4.2 Test2: Response time between UPS and UPD

In this scenario, the UPS is configured as the client and UPD is configured as a MySQL server which stores policies. In this test, a timer was set from the beginning

of UPS processing until operations e.g. storage on the UPD finishes. The figure 34 depicts the performance between the UPS and UPD.

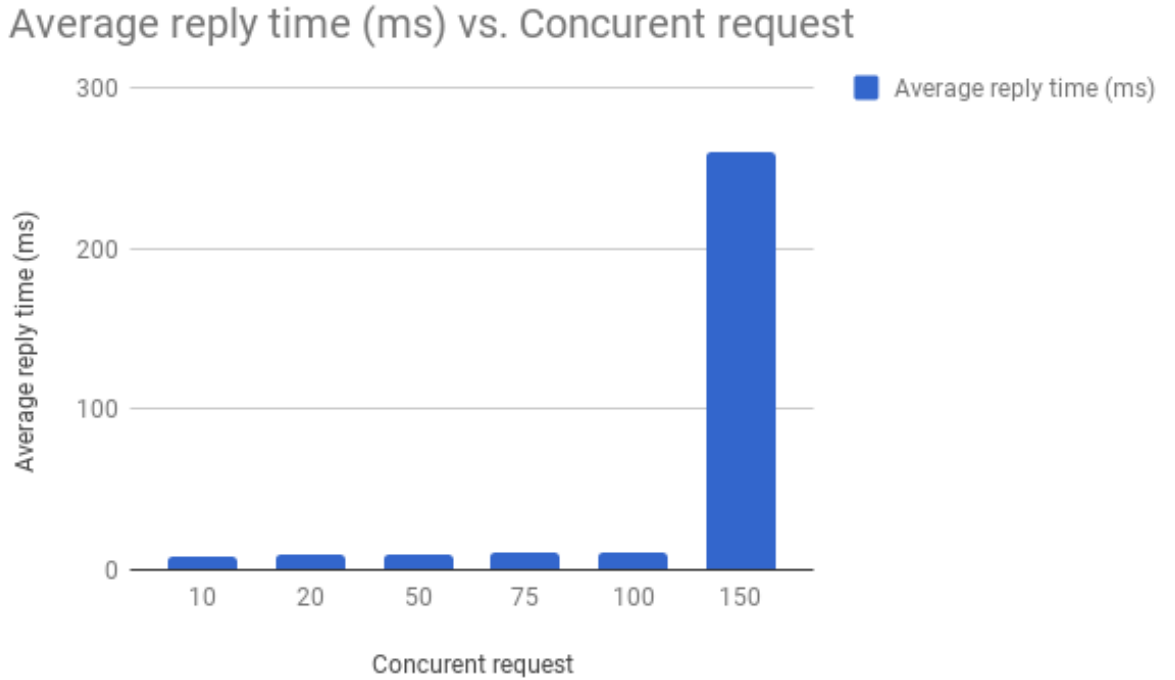


Figure 34: Average reply time versus number of concurrent requests

In this test we found out that the UPS starts displaying TCP connection error when it reaches 100 users simultaneously per second and the fact that in section 7.4.1 the number of simultaneous connections that UPS could handle without processing was 2000. We can conclude that the bottleneck resides in UPD. Therefore the UPS was able to create 100 simultaneous connection to the UPD. For that reason the UPS can handle up to 100 simultaneous users in a second. The limited number of parallel connections to UPD may be due to weak computational resources of the virtual machine on which UPD runs (Fofana-VirtualBox). To have more connections between UPS and UPD, the UPD machine should have higher computational resources.

7.4.3 Test3: Average number of replies between concurrent Policy Apps and UPS with processing

In this scenario the overall performance between PCBS components such as Policy App, UPS and UPD was measured. This test was executed for several different numbers of simultaneous clients to measure how the UPS could handle different loads. In each measurement a total of 10000 requests was sent to the UPS, using both *httperf* and *Jmeter* and this was repeated 10 times in order to reduce the margin of error. Figure 35 depicts the average response time versus the number of concurrent requests. The y-axis shows the average reply time for all experiments and the x-axis

describes the number of simultaneous users. At 100 users we can see that the average response time is between 10-20ms and above that we can see that the response time increases at a higher rate, as well with the standard deviation shown in Table 5 where we can see a more unsteady test results from 200 users and forth.

Average Reply Time (ms) vs. Number of Users

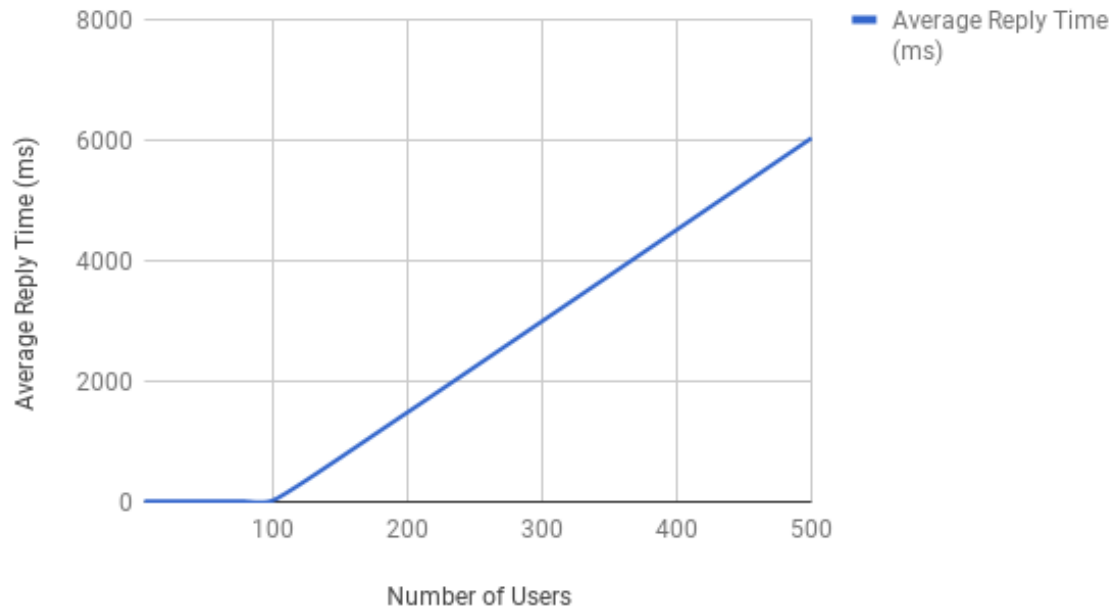


Figure 35: Average response time of the UPS with different simultaneous users

Number of users	5	10	25	50	75	100	200	500
Average reply time(ms)	10.2	10.3	10.3	10.5	10.7	20.1	1481	7035
Standard deviation	0.4	0.9	0.9	1.4	2.9	11.5	1097	2729

Table 5: Average reply time and standard deviation for different simultaneous users

Since we want to have a situation as reasonable as possible without getting misleading results because of an overloaded server, we can conclude from section 7.4.2 and the results shown in this section that the UPS can handle up to 100 users simultaneously per second. The performance of the server will decrease when this limit is reached and the server may drop some of the requests in the queue.

7.5 Test4: Policy App performance

Here we measure the performance of the Android and Linux application. Since each of the Policy App fetch only the Internet connected application on the end user device, therefore, the performance of the application depends on how many running applications are connected to Internet on the end user device. We measure the time interval when the program starts running to when it ends. The Android tests were performed on both smartphones but we consider the test result on the Samsung J5 since it has Android version 6.0.1. The Linux application test was performed on Fofana-VirtualBox. The tests for the two applications were executed for 1000 times and the average times were calculated based on the collected information. As shown in Table 6, the Android application takes the average of 8070 milliseconds to fetch 54 applications and it takes 20 milliseconds to send and get the reply from the UPS. The Linux application takes the average of 1827 milliseconds for 159 applications with 12 milliseconds response time from the UPS.

Since the Linux application runs on a more powerful computer and is connected to the UPS with a wired GbE line in the same LAN, the fetching time as well as the response time from the UPS is less than that of the Android application.

Type	Number of Apps	Collecting Apps Info (time/duration)	UPS Response Time (ms)
Linux App	158	1827	12
Android App	54	8070	20

Table 6: UPA performance test

7.6 Conclusion and Future works

The principal objectives of this thesis was to develop and test a working system to bootstrap policy based management system for the end-users, allowing them to set policies for their devices. This was to ensure that malicious policies will not find their way to the CES network based policy management system. Furthermore, a utility tool was developed for Android and Linux device to get full knowledge of all installed applications running on the device. The collected informations from utility tool are then are passed to the the User Policy Server for further processing and stored in User Policy Database which later will help the end-users for future validation of their policies. The system provides the necessary capabilities that end-users need when performing policy related tasks.

The applications developed as part of the thesis are designed to be user friendly. The implementation is tested with varying test scenarios to determine maximum capacities of various components involved. The testing proves the feasibility of the system as well as illustrates shortcomings, for example, in terms of read/write times of the UPD. The scalability of the system depends on the machine that is hosting the UPD, therefore, with the higher performing device it may scale up.

Policy Creation and Validation System (PCBS) was designed as one of the management tool for CES because the architecture will protect the PMS against malicious users as well as the failure of PCBS will not affect the PMS. We proposed this architecture with an end goal to make the development of future new functionalities easy. Moreover, the research and methodology followed in this thesis could be again extended by creating new Policy Apps for some other platforms such as iOS and Windows to allow PCBS to work with all kind of devices.

In future, the Android application may be redesigned so that it can scan any malicious applications that the user may install on his device, allowing the device to uninstall the application when it finds one as well as send the notification to the MPS about the malware. In this way, the treat to the host may reduce.

The Policy Creation and Validation System implemented in this thesis is intended for research and educational purposes and can be used as a demo tool for the CES policy tool. Therefore, it is not a product but rather a prototype.

7.7 Discussion

The validation mechanism of Apps developed in this thesis is rudimentary. It could be used in practice so that the end users subscribe to a validation service that is provided e.g. by a Mobile services security company. When the company has validated a particular Android App, it will store information about the App to the Service Provider Table in the UPD. This makes it possible for the end user to limit its Apps to the ones that are known to be non-malicious. Such an approach may be enough e.g. for company owned phones. For some consumers, however, this approach may be too restrictive. It is for further study, how it is best to organize the possible overwriting of this limitation in the policy creation.

References

- [1] statista. Number of apps available in leading app stores as of march 2017, [online] april 29, 2017, available at: <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>.
- [2] VantagePoint. Dyn analysis summary of friday october 21 attack [online] september 29, 2017, available at : <https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/>.
- [3] G Davis. Are connected homes mirai's new best friend? | mcafee blogs. mcafee blogs. may 26, 2017, available at: <https://securingtomorrow.mcafee.com/consumer/consumer-threat-notices/connected-homes-mirais-new-best-friend/>.
- [4] G Davis. Threats report, september 29, 2017, available at : <https://www.mcafee.com/us/resources/misc/infographic-threats-report-mar-2017.pdf>.
- [5] G Davis. Cyber security spending 2017, september 29, 2017, available at : <https://businessinsights.bitdefender.com/cyber-security-spending-2017>.
- [6] H. Kabir R. Kantola and P. Loiseau. *Cooperation and end -to- end in the Internet*. Department of Communication and Networking, Aalto University, 2015.
- [7] R. Kantola. *Customer Edge Switching – A Trust-to-Trust Architecture for the Internet*. Department of Communication and Networking, Helsinki University of Technology, 2009.
- [8] L. Khelladi D. Djenouri and N. Badach. A survey of security issues in mobile ad hoc networks, 2005.
- [9] W. Stallings and L. Brown. Computer security. principles and practice, 2008.
- [10] Wikipedia. Operating system, available at: <https://en.wikipedia.org/wiki/operatingsystem>.
- [11] W. Mauerer. Linux kernel architecture, wiley publishing, inc.
- [12] A. Zeinab. *Linux platform functions for embedded target platform*. Master's Thesis, Tampere University of Technology, 2015.
- [13] N. A. Malik A. A. Sheikh, P. T. Ganai and K. A. Dar. Smartphone: Android os vs ios, 2013.
- [14] P. Peris-Lopez G. Suarez-Tangil, J. E. Tapiador and A. Ribagorda. Evolution, detection and analysis of malware for smart devices, 2014.
- [15] S. Yash R. S. Pranav and M. Saurabh. Mobile viruses, 2011.

- [16] D. K. Tushar and T. K. Minal. Denial of service attack techniques: Analysis, implementation and comparison, 2004.
- [17] R. Edwards J. Jamaluddin, N. Zotou and P. Coulton. Mobile phone vulnerabilities: a new generation of malware, 2004.
- [18] E. Kirda M. Egele, C. Kruegel and G.Vigna. Pios: Detecting privacy leaks in ios applications, 2011.
- [19] N. Kumar K. Ahmad, S. Verma and J. Shekhar. Classification of internet security attacks, 2011.
- [20] S. Pillai S. Gadgil and S. Poojary. Sql injection attacks and prevention techniques, 2014.
- [21] S. Guo Z. Chen, R. Duan, and S. Wang. Security analysis on mutual authentication against man-in-the-middle attack, 2009.
- [22] S. Taluja and R. Lal Dua. Survey on network security, threats and firewalls, 2012.
- [23] H. Kozushko. Intrusion detection: Host-based and network-based intrusion detection systems, 2003.
- [24] J. Schnizlein A. Westerinen and et al. *Terminology for policy-based management – rfc 3198*, 2001.
- [25] J. Viega and G. R. McGraw. Building secure software: How to avoid security problems the right way, 2001.
- [26] A. CHIDAMBARAM. *Implementation and validation of network policy services*. Computer Networking, North Carolina State University, 2002.
- [27] TechoPedia. *Network Security Policy available at: <https://www.techopedia.com/definition/29916/network-security-policy>*.
- [28] K. Zeilenga. Lightweight directory access protocol (ldap): Technical specification road map, 2006.
- [29] L. A. Lymberopoulos. *An Adaptive Policy Based Framework for Network Management*. Department of Computing, University of London, 2004.
- [30] B. A. Forouzan. Tcp/ip protocol suite, mcgraw-hill, 2002.
- [31] F. Audet and C. Jennings. Network address translation (nat) behavioral requirements for unicast udp, rfc 4787, 2007.
- [32] R. van der Meulen. Gartner says 6.4 billion connected things will be in use in 2016 up 30 percent from, 2015, [online] available: <http://www.gartner.com/newsroom/id/3165317>.

- [33] University of Southern California Information Sciences Institute. Transmission control protocol, rfc 793, 1981.
- [34] G. Tsirtsis and P. Srisuresh. Network address translation-protocol translation (nat-pt) , rfc 2766, 2000.
- [35] P. Matthews J. Rosenberg, R. Mahy and D. Wing. Session traversal utilities for nat (stun) , rfc 5389, 2008.
- [36] R. Mahy J. Rosenberg and P. Matthews. Traversal using relays around nat (turn) , rfc 5766, 2010.
- [37] J. S. Llorente. *Private Realm Gateway, Master Thesis*. Department of Communication and Networking, Alto University, 2012.
- [38] M. Pahlevan. *Signaling and Policy Enforcement for Co-operative Firewalls, Master Thesis*. Department of Communication and Networking, Alto University, 2013.
- [39] Kantola R. Cetp, 2015, available at: <http://www.re2ee.org/cetp-protocol-v08-2.pdf>.
- [40] J. L. Santos R, Kantola and N. Bejar. *Policy Based Communications for 5G Mobile with Customer Edge Switching*. Department of Communication and Networking, Alto University, 2013.