Abdalla Taha

Software-Defined Networking and its Security

School of Electrical Engineering

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

Espoo 04.08.2014

Thesis supervisor:

Prof. Jukka Manner

Thesis advisor:

M.Sc. (Tech.) Timo Kiravuo

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Abdalla Taha

Title: Software-Defined Networking and its Security

Date: 04.08.2014 Language: English Number of pages:7+58

Department of Communications and Networking)

Professorship: Networking Code: S-38

Supervisor: Prof. Jukka Manner

Advisor: M.Sc. (Tech.) Timo Kiravuo

Software-Defined Networking (SDN) is a new principle in the networking paradigm, which makes a communication network programmable. With this utility, security properties can be redesigned. SDN provides new axioms for security applications to implement security services. In this thesis, security threats troubling Ethernet technology are interpreted and whether the SDN can provide a solution to overcome these is discussed. Furthermore, existing SDN security applications are presented and compared with author's SDN solution. The solution is a theoretical design of an SDN implementation that eliminates broadcast messages in a network, which is major Ethernet vulnerability. Furthermore, development directions are proposed for further investigations for secure SDN development.

Keywords: SDN, OpenFlow, Networking, Network Security, Ethernet, Broadcast, Denial of Service

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Abdalla Taha

Työn nimi: Software-Defined Networking ja sen tietoturva

Päivämäärä: 04.08.2014 Kieli: Englanti Sivumäärä:7+58

Tietoliikenne- ja tietoverkkotekniikan laitos)

Professuuri: Tietoverkot Koodi: S-38

Valvoja: Prof. Jukka Manner

Ohjaaja: DI Timo Kiravuo

Software-Defined Networking (SDN) eli ohjelmoitava tietoverkko on uusi periaate tietoverkkotekniikassa, jonka tarkoitus on tehdä tietoverkosta ohjelmoitavan. Tämän periaatteen avulla, tietoverkon tietoturvaominaisuuksia voidaan toteuttaa uudella tavalla. SDN tarjoaa uusia ulottuvuuksia tietoturvasovelluksien toteuttamisessa mahdollistaen uusia tietoturvapalveluiden suunnittelua. Tämän työn tehtävänä on selvittää onko mahdollista ratkaista nykyisen Ethernet tekniikkaa vaivaavat tietoturvauhat SDN periaatteen avulla. Ethernet tekniikan yleisimmät uhat esitellään työssä yksitellen ja niihin ehdotetaan ratkaisu SDN periaatteen avulla. Lisäksi työ tulkitsee jo olemassa olevia SDN tietoturvasovelluksia sekä esittelee uuden SDN tietoturvaa kohottavan ratkaisun teoreettisella tasolla. Tämä ratkaisu on SDN mallinnus järjestelmästä joka eliminoi Ethernet verkossa yleislähetys tiedotteet (broadcast message), joka on yksi merkittävimmistä haavoittuvuuksista Ethernet verkoissa. Lopuksi työssä ehdotetaan SDN tietoturva ratkaisulle kehityssuuntia jatkotutkimuksia varten.

Avainsanat: SDN, OpenFlow, Tietoverkko, Tietoturva, Ethernet, Broadcast, Denial of Service

Preface

بسم الله الرحمٰن الرحيم

In the name of God, the Most Gracious, the Most Merciful.

At the beginning of last year, I started my research of Software-Designed Networking (SDN). The task was to study its possibilities to improve Ethernet security. At that point, I did not have a real grasp of my thesis topic. After going through some tutorials, reading immense amount of documents, watching dozens of lecture videos, and familiarizing myself with SDN software, I finally understood what SDN really is and what it can be capable of. Through the consultation of my instructor Timo Kiravuo, co-instructor Mikko Särelä and professor Jukka Manner, my thesis problem statement was set and I was ready to begin the journey of my thesis.

Working on the thesis was a challenge for me, as it was one of the major project that required tremendous amount of hard work and dedication. Many times I was concerned that I would not finish my work on time. Nevertheless, I was well supported by my friends and my dearest family. This work represent the final step in my student life towards my first step in my graduated life. Firstly, I would love to thank God, whom all thank and gratitude belong and who provided me with supportive family.

Therefore, I am eternally grateful for all the help, support and love my parents Ouafaa and Imad have given me throughout these years. Without their commitment and encouragement the completion of this thesis would have been impossible. My thanks goes also to my lovely sisters Lamia, Rifaa, Saara, and Yusra who there for me. I must also thank Lamia's husband Saed and my wonderful nephew Hamza for brightening my days. I can not thank enough my wife, my true love, Meriem, who generously and with dedication assisted me through this dissertation. Furthermore, my thanks goes to my extended family and my friends, who gave me a life full of adventures.Finally, goes to my supervisors professor Jukka Manner, instructor Timo Kiravuo, and co-instructor Mikko Särelä, who have been a great inspiration and a pillar of support during this project.

Now is the start of a new chapter of my life, I welcome you!

Finland, Espoo, 31.05.2014 Abdalla I. R. Taha

Contents

Abbreviations

API	Application Programming Interface
ARP	Address Resolution Protocol
CAM	Content Addressable Memory
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service
DPI	Deep Packet Inspection
DPLL	Davis-Putnam-Logemann-Loveland
FRESCO	Framework for Enabling Security Controls in OpenFlow networks
FRESCO DE	FRESCO Development Environment
FRESCO RC	FRESCO Resource Controller
FRESCO SEK	FRESCO Security Enforcement Kernel
HTTP	Hypertext Transfer Protocol
ID	Identifier
IDS	Intrusion Detection Systems
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPS	Intrusion Prevention Systems
ISO	International Organization of Standardization
ISOC	Information Security Operations Center
IT	Information Technology
LAN	Local Area Network
LLDP	Link Layer Discovery Protocol
MAC	Medium Access Control
NAT	Network Address Translator
NIC	Network Interface Controller
ONF	Open Network Foundation
OS	Operating System
OSI	Open Systems Interconnection
PVLAN	Private VLAN
QoS	Quality of Service
S-ARP	Secure ARP
SDN	Software-Defined Network(ing)
SDSec	Software-Defined Security
SE-Floodlight	Security Enhanced Floodlight
SMT	Satisfiability Modulo Theories
SRI	Stanford Research Institute
SSH	Secure Shell
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
VLAN	Virtual LAN
WLAN	Wireless LAN

1 Introduction

Switches, routers, and many other networking devices are the key components in our daily used Internet. The increasing demands of improvement in the networking capabilities and performance have driven the network engineers to innovate new methodologies and developing perspectives for enhancing and improving the network architecture. A new development perspective to continue the network technology evolution has been introduced, the Software-Defined Networking (SDN).

The Ethernet, which is the dominant networking technology in Local Area Networks (LAN), is in crisis. Developed initially decades ago, the prime function of the Ethernet was to serve a flexible and a decentralized network. Since then, Ethernet has progressed to serve expanded networks with high capacity, yet leaving the architectural security with little attention. Furthermore, popular emerging networking technologies such as network virtualization use extended Ethernet segments. Due to the lack of routing in Ethernet, such technologies suffer from inefficiency in transmission.

Otherwise, network security has been improving rapidly over the past years. As new protocols and proprietary technologies are developed to secure communication between and within the hosts and other network devices, many vulnerabilities are still present with no current resolution. Experts face challenges as their solutions are constrained by the concept of how the legacy network operates. The legacy network is generally constructed with distributed intelligence. This appears as the nodes of the network handle their behavior according to the adjacent node. This directs security experts to resolve the security issues in restricted manner. By choice, security issues would be easier to tackle if the control over the whole network would be given to the security entity.

The main unique feature which makes SDN popular is its ability to make a network programmable. The new emerging technology has the potential to revolutionize the networking industry in many ways. Nevertheless, security aspects have to be managed and evaluated in order to provide reliable networks. Therefore, research of SDN capabilities to overcome legacy Ethernet security issues is needed.

This thesis aims to answer the question of how to improve Ethernet LAN security utilizing SDN properties, and which challenges can be expected. The network security is discussed in a broad manner, but the main focus is set on the Ethernet LAN. Existing SDN security solutions are reviewed and security issues prevailing in the Ethernet networks are discussed from the SDN point of view. In addition, a proposal for security enforcing SDN solution is introduced and evaluated. Yet, the further objective of this thesis is to open up a door for discussion and innovation in the research community to confront security challenges in SDN.

The result of the thesis shows that, SDN can provide means to dominate a network,

and hence, improve its security. SDN security can be seen as an own development field which aims to overcome all legacy network vulnerabilities and improve networks reliability and integrity. Although, SDN is not airtight, the central entity (namely controller) of a SDN network represents a single point of failure. The network is in complete submission to the controller which becomes a threat when its compromised. As long as the the access to the controller is restricted, SDN can bring many advantages for network security improvement. This thesis contributes to the further SDN security development and research, by providing possible SDN solutions to overcome legacy Ethernet security threats and by presenting an example of a secure SDN network.

The structure is as follows, Chapter 2 explains the reader the background knowledge needed, with defining SDN, its own standard OpenFlow, the basics of Ethernet and Ethernet's security protocols and also reviews existing SDN security implementations and discusses the possible future trends of the security of SDN. In Chapter 3, a survey discussing Ethernet security is discussed by proposing a solution to improve Ethernet's security with SDN. In Chapter 4, a proposal for a SDN security implementation is presented and evaluated. Finalized by Chapter 5, which discusses the development directions to SDN security solutions and concludes this thesis.

2 Background

In this chapter, the required background information is explained in order to understand the thesis' notion. Firstly, Software-Defined Networking is defined and explained. This is followed by explaining its standardized protocol OpenFlow and by defining the overall view of the existing Ethernet security protocols and techniques. Related work are presented and interpreted and additionally the future trends and possible challenges will be investigated.

2.1 Software-Defined Networking

Software-Defined Networking (SDN) concept was first time introduced in 2010 [1] as the new networking paradigm which aims to ease the control and the management of a computer network environment.

SDN can be explained as an architectural principle where the networks control and the management are centralized and decoupled from data plane, thus making the network programmable. To quote the definition paraphrased from the HotSDN '12 Solicitation:

> *Software-Defined Networking (SDN) is a refactoring of the relationship between network devices and the software that controls them.*

Traditionally, the data and the control planes in the Ethernet networking devices (and most of the communication principles) have been tied together. This means, the prevailing operating system and its features with the provided hardware are implemented in a single device. Therefore, network devices, such as switches, routers, firewalls, etc., are built with the intelligence of handling traffic relative to the adjacent devices. This makes the intelligence distributed and scattered in the network. In addition, most of the network devices are Command Line Interface (CLI) based and configuration is done separately per device, making configuration slow and prone to errors. This prevents the networking industry of responding quickly to feature requests or innovate new management abilities. [2]

The **data plane** has a sensible layering model which is known by the name Open Systems Interconnection model (OSI model)[3]. It is well know model in the networking industries and academies. It is standardized by the International Organization for Standardization (ISO). The OSI model enables network applications and services to isolate the data operations to a single layer and provide interfaces between layers. This has enforced developers to develop and improve the operations without concerning the other layers. This type of layering models are used in many other fields (e.g. operating system) and it has provided simplicity to understand the overall view and the interactions. As a result, we can witness increase in the development and research in these fields. As it seems, similar layering model is essentially needed for networks control and management plane, which was not available. This creates

the need to invent a new networking architecture, SDN.

SDN architecture decouples control from data plane and provides it a new layering model. Open Networking Foundation (ONF) [4] is a non-profit industry consortium which has taken the lead in standardizing critical elements for SDN architecture. One of these standards is OpenFlow, which will be covered in the next section. ONF has defined the following layering scheme for SDN architecture.

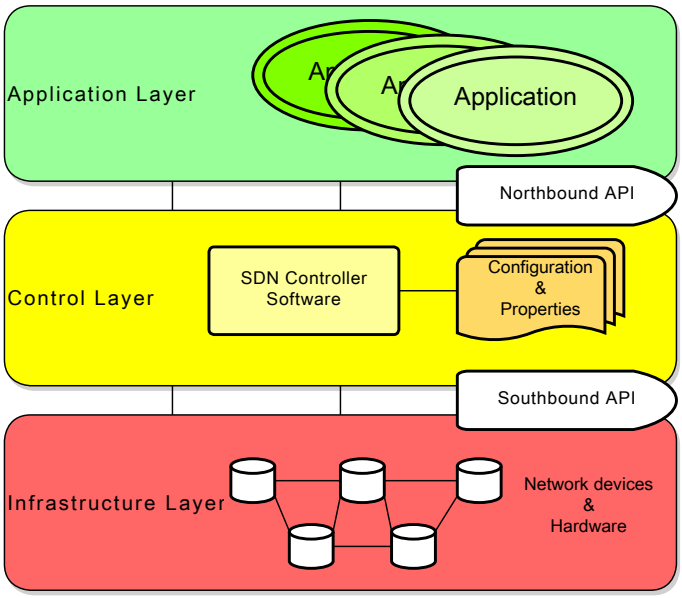

Figure 1: The SDN stack.

As seen from the Figure 1, SDN architecture is divided into three layers: application layer, control layer, and infrastructure layer. This architecture and arrangement of control and management, provides the possibility to centralize the state of the network and the intelligence into one part of the network. This, enhances the property of network programmability, the network industry can start to innovate and enable differentiation in the developing process. Furthermore, programmability accelerates creativity and introduction of new network features and services. With centralization, SDN simplifies provisioning while optimizing performance and granularity of the policy management. Therefore, SDN can make networks become more scalable, flexible and proactive. SDN architecture stack abstracts and decouples hardware from software, control plane from forwarding plane, and physical from logical configuration.

Infrastructure layer is the layer where all the hardware exists and are connected physically. On these hardware devices runs a software which provides a control data plane interface (*Southbound API*) which is used to communicate with the upper level: Control layer.

Control layer is the most important layer in the architecture. There is a controller which talks to all the network devices in the infrastructure and keeps track of the topology. While exchanging information of the network state with upper layer applications (through (*Northbound API*), the controller translates their commands to the network devices to have respective and desired network behavior.

Application layer is the layer where all the features, services and policies are defined. Applications request the information of network devices and the topology in order to to act upon it. These applications can create features end-to-end and make big picture decisions according to the changes in the network. When the network topology, feature, or policy requirements changes, applications have the control to change dynamically the network behavior from one single point.

Between these layers, there are Application Programming Interfaces (APIs) which provide the essential communication tools between the layers. The *Northbound API* is provided by the controller and the applications have to manage their communication to the controller through it. At the time of writing this thesis, there was no standards for this interface, yet many SDN controllers were settling down with REST API [2]. Other popular API's are C++, JAVA and Python. The *Southbound API* is the communication between the controller and the network devices. In the following section, a protocol providing this feature shall be further discussed.

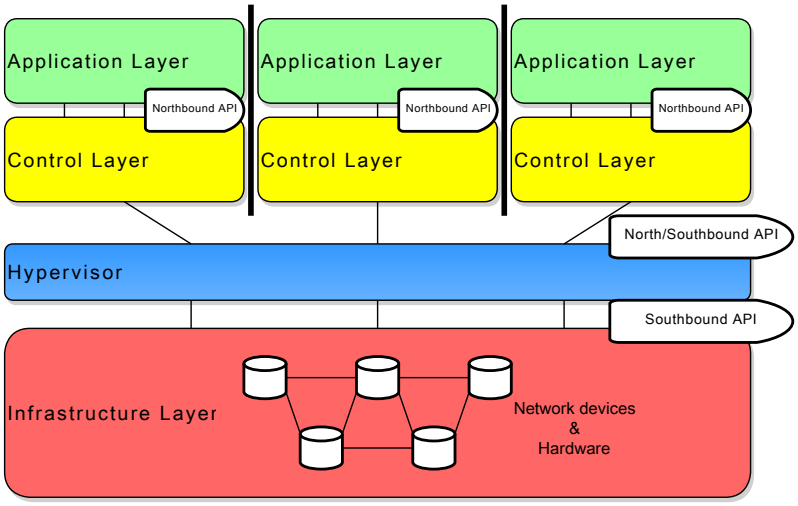

Figure 2: The SDN stack with Hypervisor(slicing).

The SDN layer model can be adjusted to fit and satisfy a data center scheme. Where virtualization may need logical network segment slicing. In this case an extra layer is added in the model (see Figure 2) and the controller of controllers, also called the **Hypervisor**, can provide logical slicing (while physically network is untouched). In

other words, the Hypervisor allows every individual controller to control only their own hosts (physical or virtual) on the network without affecting other parts of the network.

SDN brings new challenges in networking technology and in this thesis the focus is set on the network security. A programmable network provides full control of a network. Thus, bringing more capabilities to handle security in the network, whether in a local area network (LAN) or in the core. Similarly, new unexplored threats against a SDN network can be unveiled or discovered. Nevertheless, SDN has gained rapidly its reputation and is the biggest hype word in networking business.

2.2 OpenFlow

As mentioned in the previous section, the communication between network devices and the controller is handled through the southbound API of the controller. As the most dominant networking technology is the Ethernet, the first standard for SDN was created to manage the controlling of Ethernet switches. OpenFlow [5] is a standardized (by ONF) protocol for SDN supported networks to handle the communication between Ethernet switches and the SDN controller. OpenFlow was derived from SANE [6] and Ethane [7], which were one of the first projects to decouple control and data plane. OpenFlow shortly started to become more popular and as an open standard, it developed quickly to support more and more functionalities.

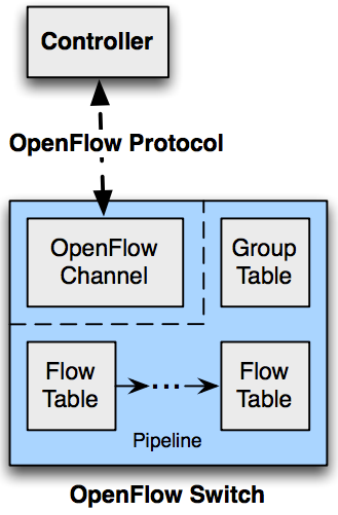

Figure 3: The three main parts of an OpenFlow switch [8].

A switch which supports OpenFlow, initially consist of three essential parts [8](see Figure 3): flow and group table(s), OpenFlow channel, and OpenFlow protocol. As OpenFlow is relatively young and it continues on developing, new functionalities

and improvements change OpenFlow's properties to suite current network demands. Therefore in this section, only the essential parts of OpenFlow will be explained and other improvements and version differences are explained later on according to referral need.

Using OpenFlow protocol, a controller can learn Ethernet switches' hardware details, connectivity status, and the network topology; while commanding its forwarding table behavior by giving flow rules.

Flow and group tables define switches behavior with data flow coming from different interfaces (physical, virtual). The table consists of a set of rules, where the flow of the communication data is defined. The Switch reacts upon every flow according to the rules, called flow rules. An OpenFlow switch has one or more flow tables and a group table for frame lookups and forwarding. A flow rule consists of three fields (see Figure 4):

- *Rule*: a header to match with the frames of the flows. There are several supported Ethernet headers in OpenFlow specification [8], but as OpenFlow is made to be extensible, custom headers can be additionally defined. The switch merely performs a bit mask match. Therefore, OpenFlow switch is open for innovative non-IP traffic.
- *Action*: as a rule is matched with traffic, the action which should be performed for it has to be defined. The actions are also open for extensions, but some basic actions are already provided in the specification. Such as, forwarding to one or more ports, forward to the controller, drop the frame, and modify frame fields. The only requirement for adding customized actions is that the data path must have flexibility while providing high performance and low cost.
- *Statistics*: Each and every time when a flow rule is matched, the switch has to update the frame counters, which indicates the popularity of a specific flow. There are counters for every table, each flow, all the ports and every queue. Also a timer of last activity and initial set of the flow are maintained.

An **OpenFlow channel** is the connection between the switch and the controller. This channel is usually encrypted with Transport Layer Security (TLS) protocol, however, the channel can also be run by using plain Transmission Control Protocol (TCP). The OpenFlow channel provides an interface for the controller to manage and adjust flow and group tables of the OpenFlow switch. Concurrently, the switch also supplies the controller with its hardware information, the connectivity status of ports, and meter statistics of every flow rule.

The **OpenFlow protocol** predefines the communication message pattern which is used when communicating between the controller and the switch or between the switches.

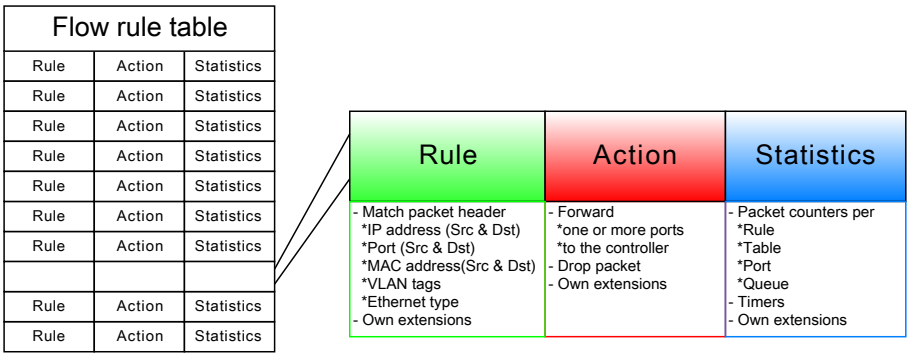

Figure 4: An example of an OpenFlow flow rule table.

2.3 Ethernet and Security

The IEEE 802.3 [9], better known as Ethernet transmission, started out in the mid-seventies as a simple method of communicating between computers [10]. Rapidly, it became the de facto standard [11] for the local area networks (LANs) due to its simple nature. Regardless, Ethernet's security aspects were not considered or designed at its early stages. As the networks grew and security became more critical, some security enhancements began to affect the Ethernet. The different security issues troubling Ethernet cannot be detected from upper layer protocols. Therefore, focusing on improving Ethernet's security and reliability is a great importance. Thus, with SDN in the picture, questions can be raised: is it necessary for Ethernet to get an architectural upgrade? Does it improve the security?

In this section the basic principles of legacy Ethernet are briefly explained and those standards and protocols which are relevant to the notion of this thesis are presented.

2.3.1 Plain Ethernet Network

A plain Ethernet network (mostly LAN) consists of hosts and switches. Switches are used to connect the host with each other creating a network. The topology in simplest form is a star, but while connecting switches with each other, it creates a mesh topology. The hosts are required to have a 48-bit universal LAN MAC address [12] (shortly MAC address) to each of its Ethernet interfaces connected to the network. The switches need to keep track of host MAC addresses connected to it to be able to guide the traffic.

As the communication traffic is packet based (IP packets), in the Ethernet, the packets are encapsulated in an Ethernet frame. These frames are packed with Ethernet headers containing the source and the destination. The creation and resolving of Ethernet frames is handled by the hosts and the switches inspect only the headers for forwarding purposes.

An Ethernet switch has multiple interfaces (Ethernet ports) for internal switching. The switch's main task is to forward Ethernet frames from one port to the other, with the respect of its destination. Therefore, the switch maintains a record of which MAC addresses are connected to which port. This table is called Content Addressable Memory (CAM). Initially, the switch does not know the MAC addresses of its connected hosts, the information is collected when the hosts send an Ethernet frame. The switch will learn, while forwarding the frame, the source MAC address. The MAC port pair is added if it was not added earlier. Using this same table the switch knows to which port it needs to forward the frame. In case there is no match, the switch floods it, which means that it sends the frame out of all of its port (except the port where the frame came from). According to the Ethernet's principle, the host should only take frames which are destined to itself, others are dropped.

Generally, an Ethernet switch has three main components:

- *Data plane*: Is responsible for the actual forwarding of the Ethernet frame from one port to the other.
- *Control plane*: Takes care of updating the CAM and manages the network topology such as Spanning Tree Protocol (STP).
- *Management plane*: Is used to carry out the personal network configurations and other features such as virtual LAN's (VLAN's).

Even though, physically, LAN's have star or mesh shaped topology, logically, the network is always viewed as a tree. It is also possible to divide the physically created network into smaller logically divided networks. These are called Virtual Local Area Network (VLAN). This is the means to connect the host to the same LAN without the locational restriction. [13] [14]

In the case of a mesh network, for redundancy reasons, in the network may occur loops. This is not so convenient in the Ethernet's point of view since the network is logically a tree. This is why a protocol called Spanning Tree Protocol (STP) is used. STP gives each switch a priority and it detects the loops in the network. One of the switches is initially selected as a root switch, and by sending specific messages it can detect the network and disable the connections which create loops. If an active link is suddenly not available (i.e. due to link failure), the switches give notice about their state and the disabled connections can be reactivated for repair. [15]

2.3.2 Host Connectivity Management

For host connectivity in the IP, some layer 3 protocols are tightly related to Ethernet's behavior. Relevant for this thesis we have to inspect two of them: Dynamic Host Configuration Protocol (DHCP) [16] and Address Resolution Protocol

(ARP) [17].

As explained before, hosts has a unique MAC address for each networking interface. To locate other hosts in the same LAN, IP addresses are needed. **DHCP**, is the protocol for assigning IP addresses to the hosts. When a host requires an IP address, it has to locate the DHCP server and communicate with it. In return DCHP server responds with an IP address with its lease time and other network related information (such as netmask, gateway address and Domain Name System (DNS) [18] [19] servers).

After gaining an IP address, the host can connect to other hosts in the same LAN. Nevertheless, since Ethernet frames use MAC addresses instead of IP addresses, **ARP** is used. With ARP, a host sends a broadcast message to the LAN requesting the recipient host (with the specific IP address) to reply. After the reply, the host assigns the MAC address to the specific IP address.

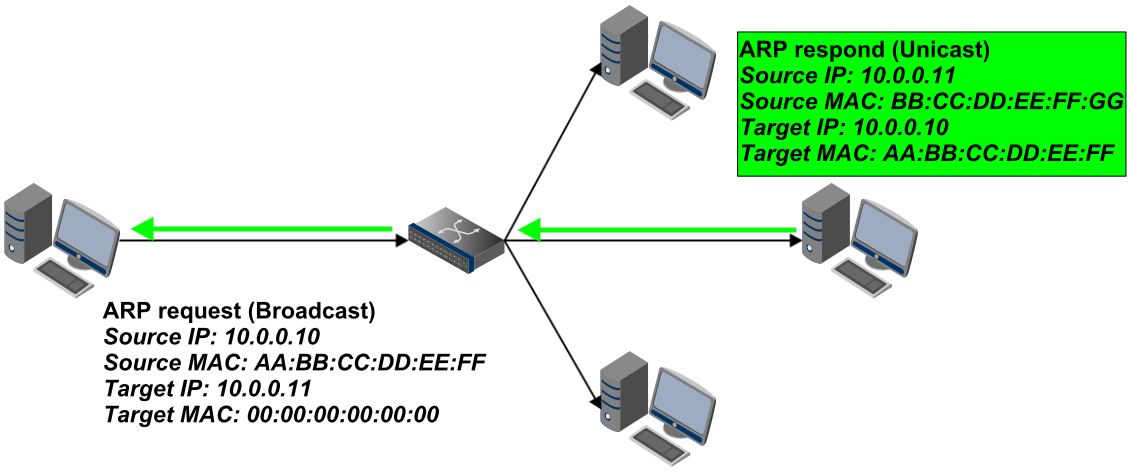

Figure 5: An example of ARP communication process.

2.3.3 Legacy Ethernet Security Solutions

Most of the network security solutions are implemented in the higher layers (layers 3-5). Some lower level solutions are also available but most of them are vendor provided proprietary solutions. From the popular and standardized solutions, in this thesis are discussed: firewall [20], Deep Packet Inspection (DPI) [21], Intrusion Prevention Systems (IPS) [22], and IEEE 802.1X [23] [24].

One of the most popular security systems is a **firewall**. Firewalls' main task is to filter and protect a host or a LAN from unwanted communication traffic. The term *firewall* is very generic as there are dozens of different implementations. The implementations vary depending on how deeply the traffic must be investigated (layers

3-5). A simple firewall can prevent traffic according to initially defined IP header variables (layer 3) and it is more suitable for network protection and a bit more sophisticated firewall at the host can follow connection sessions and prevent initially defined illegal connections.

Sometimes firewalls are enhanced with **DPI** functionality which scans the traffic at the application layer (layer 5). This is done by examining each passing IP packet's content (past the headers) for suspicious data. Thus, firewalls with DPI have the ability to identify applications and higher level protocol attacks.

DPI is also used in **IPS**. IPSs' task is to investigate the passing traffic and as the name reveals, prevent intrusion into the system or the network. IPSs' detection system compares the passing traffic with its known attack database and thus protects the traffic from malicious attackers.

Although above mentioned solutions secure higher layer communication, they are frequently used while implementing Ethernet LAN. Regardless, **IEEE 802.1X** is a standardized solution to upgrade Ethernet security level with the authentication feature. As a host connects to an Ethernet switch, it has to send its credentials in order to acquire connectivity. The switch then confirms the authenticity of the host from a remote authentication server. Thus only the authenticated hosts may communicate in the network.

2.4 Related work

Currently, at the time of writing this thesis, a state of stillness prevails regarding the solutions to improve network security with SDN and OpenFlow. However, there are some current trendy implementations which are attracting network security specialists. In this chapter, these implementations will be presented and discussed.

2.4.1 FRESCO and SE-FLOODLIGHT

The Framework for Enabling Security Controls in OpenFlow networks (FRESCO) [25], was introduced by SRI International and Texas A&M University in the ISOC Network and Distributed System Security Symposium in San Diego on February 2013. With the focus on the key issues of designing new security services in the OpenFlow enabled network and enforcing creativity, adjustability, and deployability with creating network security solutions, FRESCO provides a new framework for developing security applications in an OpenFlow network. In SDN architecture, FRESCO is situated on the control and application layers, providing an OpenFlow controller and a new application developing environment to create FRESCO supported security applications. Applications are created using FRESCO's own modular libraries and scripting language, inspired by the Click [26] router architecture.

FRESCO Security Enforcement Kernel The FRESCO's controller is a modified version of NOX-classic controller. NOX-classic is a controller programmed in C++ and Python, providing the possibility to create OpenFlow applications in both languages and run many applications simultaneously. Unfortunately, NOX-classic doesn't handle applications flow rule creation conflicts as they can compete, contradict, override each other. This creates vulnerabilities within the controller. Ergo, the NOX-classic controller's core is modified with a structural change by adding an extension, *FRESCO Security Enforcement Kernel* (FRESCO SEK) (formerly known as FortNOX [27]).

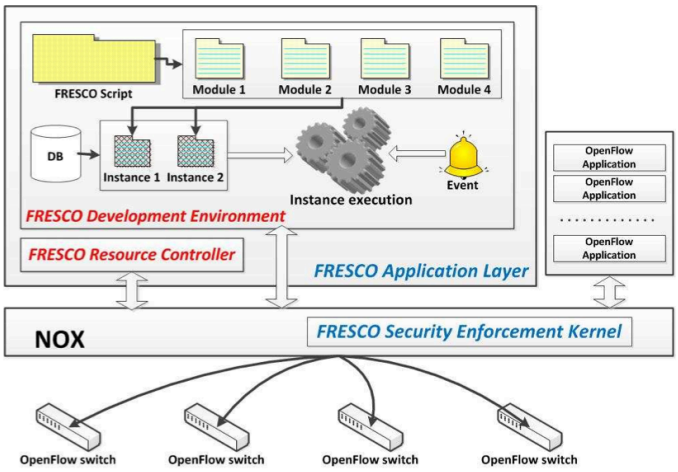

Figure 6: FRESCO architecture [25]

The FRESCO SEK's task is to provide the controller with the ability to give OpenFlow applications priorities and authorization. FRESCO SEK supplies the controller role-based authorization and rule conflict detection, not only for FRESCO applications, but also for other OpenFlow applications compatible with NOX-classic. This offers FRESCO applications assurance that their flow rules are prioritized and enforced over competing flow rules by non-security-critical applications. With role-based authorization, FRESCO SEK provides a FRESCO application the possibility to digitally sign the flow rule, hence, the controller can identify the flow rule and determine if the rule originated from a FRESCO or an OpenFlow application, or from a network administrator. As follows, upon a flow rule conflict, with FRESCO SEK's rule conflict detection mechanism, the controller knows whether a new flow rule should be overwritten or should the old flow rule stay active on the network. Obviously, non-security-critical OpenFlow applications have the chance to overwrite a flow rule set by a FRESCO application, unless the network administrator gives the authority. Nevertheless, when FRESCO applications conflict with each other, the priority between them has to be set by the network administrator.

FRESCO Application Layer FRESCO enables composing various network security functions, which are build using FRESCO's own scripting API. Utilizing FRESCO's scripting language, security practitioners are enabled to use modular libraries to set up security monitoring and thread detection logics. These modules are the basis processing units of a FRESCO application. A module is implemented as an event-driven processing function which is defined and specified in Python. The interactions of the module towards other modules and OpenFlow actions include five different types of interfaces: (i) input, (ii) output, (iii) event, (iv) parameter, and (v) action (see Figure 7). Only one event interface can be used per module, whereas all the other interfaces can be defined multiple times per module (see Figure 7).

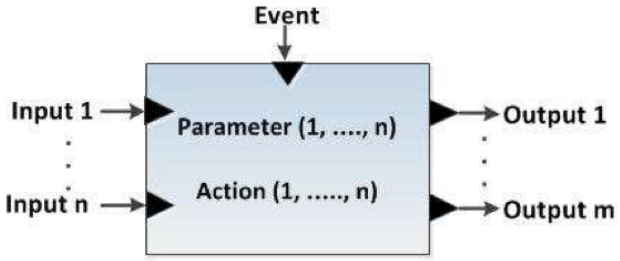

Figure 7: FRESCO Module [25]

Input and output are used for transmitting values from and to the module, event interface is used to indicate when is the module's action instantiated, parameter interface for the module's configuration on initialization values, and the action interface is for indicating the desired OpenFlow command for the specific flow or a packet. Utilizing these interfaces, modules can be combined, thus the possibility of developing security applications while having centralized control over the whole network, is opening up new varieties of possible security solutions over a network.

To introduce how creating a FRESCO application works in principle, a simple example is examined. Lets create an application which works as a port comparator over a network and lets give it a task of dropping all HTTP packets (port 80). This can be created using only two modules. First module will be triggered with event *incoming flow*, which means that every time there is a new flow coming in to a switch the module gets instantiated. The input value will be packets' *destination port* and as a parameter we select *80*. We shall trigger an action in the second module, so in the first one we are left with the output interface which is the *result* of how the second module will react. In the second module, the module is triggered with the *push* event. This means that when there is a value on the input interface, this module starts to operate. Here we do not need parameters or output interfaces. Nevertheless, there will be two action interface values: drop and forward. The function of the port comparator can be observed from figure 8 below.

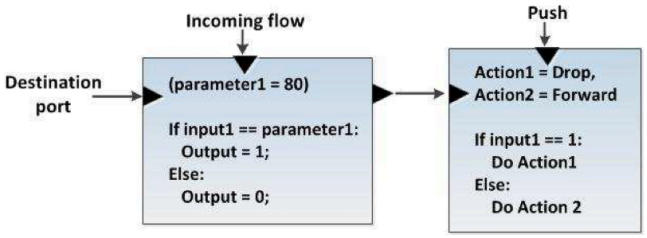

Figure 8: FRESCO Port comparator [25]

The FRESCO application layer consists of two parts: FRESCO Development Environment (FRESCO DE) and FRESCO Resource Controller (FRESCO RC). FRESCO DE provides the security researchers the tools and a platform to create and innovate new network security monitoring and thread detection services. FRESCO DE subsist of four main functions:

- **Script-to-module translation**, automatic conversion from FRESCO script to module instances, where all the essential information is defined such as which modules are used and how they are connected and what are the values of their interfaces. Also, the validation of registering modules is performed. Each FRESCO application gets a unique ID and encryption key pair. This allows the encrypting of the FRESCO script and signing it to preserve authentication.
- **Database management** handles the assembling of the information concerning the network and switch states. It provides sharing this information to modules and between them. The database can be also used as a temporary store for an instance.
- **Event management** is responsible for triggering the module instances which are waiting for the predefined event to occur.
- **Instance execution** loads the generated module instances into the memory in FRESCO framework. It also takes care to check if the application is authorized to run.

Since FRESCO was built upon NOX-classic which supports up to the OpenFlow 1.1.0 specification, FRESCO does not use the new feature presented in OpenFlow 1.3 where switches can have multiple flow rule tables. Ergo, the flow table entries are limited, FRESCO RC was created to keep track of the flow rule count in every switch. FRESCO RC needs to handle the situations when a flow table fills up and FRESCO RC has to decide which of the old flow rule will be replaced. Additionally, when an old security sensitive flow rule gets removed from a switch, FRESCO RC takes care that no new rule conflicts with the removed rule. This requires two main functions:

- The **Switch monitor**, which checks periodically the status of every switch in the network and keeps track of the flow rule amount on the flow table.
- **Garbage collection**, monitors *switch monitor* details to react upon filling up a switch flow table. When a predefined threshold of flow rule amount is achieved, the *garbage collector* starts to remove the least active flow from the switch.

FRESCO Example Application FRESCO developers have presented examples of FRESCO security implementation in their paper. One of the notable implementations is **Reflector Net**, which discovers malicious scanner attacks and redirects the attack to a third-party remote honeypot. Reflector Net was implemented using two modules, first module which discovers the scanner and the second module which redirects the scanners flow to a honeypot for further investigation possibilities.

The malicious scanner's intention is to find a host on a network with vulnerable port, thus, the attacker will try to create a TCP connection on multiple ports per host. Assuming that there is an external entity, which notifies FRESCO's database of large numbers of TCP connection fails. This information could be used to create the event TCP_CONNECTION_FAIL, therefore triggering Reflector Net's scanner detector module. Which is followed with the triggering of a redirection of the scanners flow to a designated honeypot. The flow of the attacker is identified using the source IP of the flow that created the TCP_CONNECTION_FAIL event and Reflector Net's scanner detector computes if the desired threshold of unsuccessful TCP connections have been created by the same flow. When the threshold is reached, the redirection module is triggered and therefore the flow is directed. The magic behind this is that the attacker wont realize the detection and trapping of honeypot.

SE-FLOODLIGHT Security Enhanced Floodlight (SE-Floodlight) is a derivative of FRESCO, whereas FRESCO was built on NOX-classic OpenFlow controller, SE-Floodlight is build on, as the name implies, Floodlight controller. SE-Floodlight is still under development and beta releases are available. Principally, it is quite similar to FRESCO except there is more functionality due to the extensions set by the new OpenFlow specification. Also, as Floodlight is an actively developing controller, SE-Floodlight has more possibilities to continue on its development after initial release. [28]

2.4.2 FLOVER

SDN allows a network to be operated by many applications simultaneously. With OpenFlow, these applications can communicate with network switches and command them to handle specific flows according to the desired functionality. Unfor-

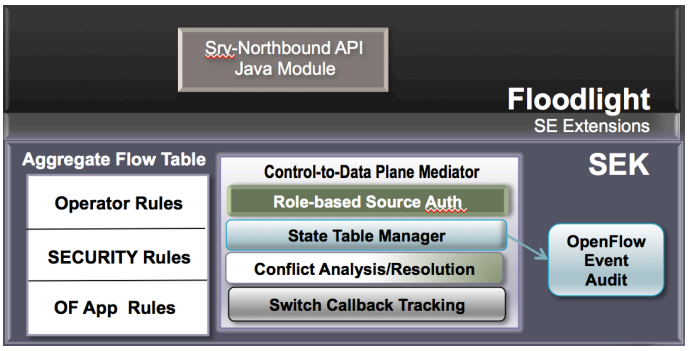

Figure 9: The architecture of SE-Floodlight [29]

tunately, these applications may not have initially cooperation managed with each other by the controller or any other element, and thus, conflicts and problems may occur when they are set to operate. In addition, OpenFlow standard has not specified nor provided a built-in mechanism for checking application flow rule production (statically or at runtime). This issue has an emphasized effect if one of the applications is a service maintaining security.

To address these type of issues, FLOVER was introduced at IEEE International Conference on Communications June 2013 by SRI International, Texas A&M University, and University of Texas at Austin [30]. FLOVER is a model checking system, which the main task is to verify that flow policies deployed by an OpenFlow application do not violate the network's security policies.

In the SDN architecture, FLOVER is implemented as an OpenFlow application but logically it is situated parallel to the controller. The controller is modified to request FLOVER's approval on every new flow rule generation or modification. Initially, the controller provides FLOVER with the security property information crucial to the network, so FLOVER can inspect requested changes and respond to them. FLOVER also uses the controller to access the current state and information about the network, such as flow rule tables on the switches.

FLOVER introduces a new effective mechanism to ensure that a new flow rule, generated dynamically by an OpenFlow application, does not offend or contradict security properties set by the administrator or a security application, especially *non-bypass properties*. By definition, a *non-bypass property* is a flow allow or deny rule, which are defined statically to prohibit or permit flows through the network. To examine this property, FLOVER addresses two types of *non-bypass property* violations that may occur in an OpenFlow rule instance.

– First type is **coverage violation**, which occurs when in a flow table there is an entry that violates a *non-bypass property* rule. For example, we assume an OpenFlow switch with the flow table below. And we have a *non-bypass*

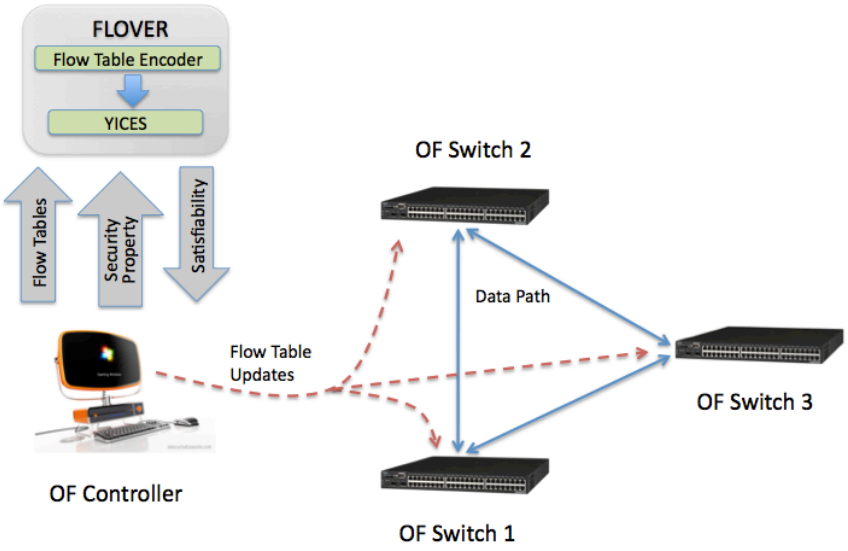

Figure 10: The architecture of FLOVER

property rule as *from source IP [5,6] no packets are allowed to destination IP 6*. As seen from the third entry, it violates the rule, making it inconsistent.

– Second type is a **modify violation**, which arises, using the same example, when a *set* command is used, as seen in the second entry. This violates the *non-bypass property* rule by modifying the packet headers and so allowing the flow to arrive at its original destination.

Table 1: Example of OpenFlow Flow Table with coverage and modify violation. [30]

	Condition				
Flow Table	Field 1 Src IP	Field 2 Src Port	Field 3 Dst IP	Field 4 Dst Port	**Action Set**
1	5	[0,19]	6	[0,19]	drop
1	5	[0,19]	[7,8]	[0,19]	set *field 1* 10, goto 2
1	6	[0,19]	[6,8]	[0,19]	forward
2	[10,12]	[0,19]	[0,12]	[0,19]	set *field 2* 6, forward

If the prevailing OpenFlow specification version is at least 1.3.0, the switches flow rule tables are enabled to be flexible and dynamic. This is due to multiple flow tables and flow rule constructs created with *set* and *goto table* operations. Therefore, FLOVER has selected Yices Satisfiability Modulo Theories (SMT) solver [31] to compute the satisfiability of the flow rule changes requested. Yices SMT solver

uses Davis-Putnam-Logemann-Loveland (DPLL) algorithm to reach efficiently the computing results. To compute with Yices SMT solver, FLOVER uses its **Flow Table Encoder** which encodes the flow rule set, a collection of flow rules and specified non-bypass properties into Yices code. Yices then computes whether non-bypass properties are violated.

FLOVER runs as an OpenFlow application on a SDN controller and it has two different execution modes to select from, **in-line** and **batch**.

- With the **in-line** mode, FLOVER is allowed to perform flow rule validation for every new flow rule change.
- **Batch** mode is used to improve controllers' response time, by allowing FLOVER to run periodically. In this mode, FLOVER collects the current flow rule table sets from every switch and checks if any violation has arisen.

FLOVER has taken up the challenge of resolving security issues in OpenFlow networks. In the paper, FLOVER developer team has run simulations and evaluated the performance of FLOVER. It is critical to examine the effect of FLOVER overhead on the flow rule approval and to realize if there might be small moments of vulnerabilities present when using FLOVER in batch mode.

2.5 Future Trends

The presented security enforcing implementations are pioneering work performed in the SDN security field. They provide extensive tools for creating security enforcing SDN applications and the means to inspect SDN applications flow rule production. Nevertheless, more pioneering and research needs to be done in the field to achieve more secure and reliable SDN implementations, which prepares SDN solutions more towards the production implementations.

Even though, at the writing time of this thesis, the development and the production of SDN security solutions are still quite few in number, it is still a hot discussion topic in many network security specialist circles [32], [33], [34], [35]. Considering that SDN era is still at its beginning and there is still ground work to be done, it is not wrong to assume that numerous companies and universities are participating, developing, and investigating the matter and have not come out with their work to public yet. The significance and importance of security in the SDN architecture is crucial. As guaranteeing a protective communication is one of the core principles of the networking technology, SDN has provided a new development principle and it has inspired the security practitioners to proceed to invent new ways to provide more secure way to communicate. In previous sections, some first publications of security enforcement solutions were presented and interpreted. From these, common focus issues and attention points can be extracted.

In the future, it appears that the most security development in SDN will focus on critical analysis of the security challenges posed by SDN and the exploration of potential new defensive capabilities that SDN may enable [32]. It is clear that the ground work is still ongoing and when most of the security challenges in SDN technologies have been tackled, SDN will have more influence on current network and the commercial market.

According to a recent Gartner research [36], leading-edge enterprise data centers are decoupling IT services from their hardware by shifting into a software-defined model. This requires evolving of the security services to become more adaptive and programmable. The vision of a software-defined data center is to virtualize all of its IT infrastructure (such as storage, networking and computation) and shift them to software defined paradigms. Therefore, it is necessary and essential to secure its all software-based architectural elements and their APIs as well as necessary protocols used in data center environment. Gartner proposes a new term, Software-Defined Security (SDSec) and the implementation of it in to the practice in three phases:

- Phase 1: **Securing Software-Defined Data Centers**. In this phase, focus is to appropriately secure all software-defined elements, their interaction, and the desired networking protocols and software defined applications. Proposed measurements are to decryption, decoding, inspection, re-encryption, authentication, and authorization.
- Phase 2: **Integrating with the Software-Defined Infrastructure**. To have SDN enabled security policies, information security services and policy enforcement should become more adaptive. Thus allowing, (SDN enforced adaptive) networks to have their security policies while topology is changing and adjusting. Demanding tremendous planning in security policy designing principles and routines.
- Phase 3: **Evolving into Software-Defined Security**. As the last phase, SDSec will evolve itself and become more software-defined. The security management plane will be decoupled from the security data plane and the security policies will be managed holistically, enforcing security management with the independence from the physical network topology and location.

2.6 Summary

In summary, SDN brings a new architectural platform to construct communication networks. The core of SDN relies on the controller as it represents the operating system of the network. The controller creates an interface for SDN applications to communicate with the networking hardware and thus implementing new features to a network becomes seamless. OpenFlow is the only standardized protocol of SDN which task is to systematize the communication between the the controller and the

switches.

Ethernet is a simple solution to create networks. The protocols which have been built on top of it has made the security aspects challenging, due to their distributed nature. It is still the dominant technology in the networking industry, and SDN did not come to replace it, but rather change the method of operation and development. Pioneer work in SDN security has started the pursuit to create more secure networks with SDN. FRESCO, which evolved to SE-Floodlight, are platforms to create SDN network upon a secure framework. The framework takes care of interoperability and priorities the applications according to their secureness. FLOVER performs model checking to a SDN network to recognize contradicting rules generated in the network.

Future of SDN depends on major companies implementing and securing Software-Defined Data Centers, and integrating them with the SDN infrastructure. The security will evolve into more software defined and thus fit better with SDN networks.

3 Solutions for Increasing Ethernet's Security with Software-Defined Networking

IEEE 802.3 Ethernet technology is ubiquitous. In the Local Area Networks (LAN), it is the most commonly used computer networking technology regardless of the use-case environments [37]. Initially Ethernet was designed to create distributed packet switching for low cost LAN's [10]. Therefore, Ethernet did not include any architectural security properties. Nonetheless, this allowed Ethernet to be easy to deploy and to use, and hence, it becames the dominant technology.

In this chapter, Ethernet security vulnerabilities will be investigated according to the survey [38] and a possible solution to overcome them using SDN will be presented. In the legacy Ethernet network, the control and data planes go tight together, making it easier for the attacker to find ways to attack these networks. In SDN, the control plane is decoupled from the data plane, isolating the data flow from the control. This makes it harder to target with several legacy network attacks. Nevertheless, as SDN can tackle most of the Ethernet security issues, it brings new issues to discover and to deal with.

The next sections answer each Ethernet threat presented in fore mentioned survey [38]. In each section there will be a brief description of the threat and the issue in legacy Ethernet network, which is followed by a possible solution to overcome it using SDN.

3.1 Network and System Access

To perform any attack to any network, the attacker has to gain access to it [39], either by physically or by controlling existing resources. After the access, the attacker can perform several different types of attacks. According to the paper [38], an attacker can have five different motives to the attack: (1) learning about the private network topology and the network traffic for use in a later attack, (2) gaining control over switches, routers, or servers in the LAN, (3) eavesdropping, (4) manipulating information, or (5) disrupting the availability of the network.

Unauthorized Joins and Expansion of the Network. The basic physical access to a network is a threat if it has been done by an unauthorized person. Connecting to an Ethernet segment in a plain legacy network is easy since the purpose of Ethernet was to provide as little as possible administration overhead. This means that if an unauthorized person access a switch, an existing host or a wall socket, which are connected to the target segment, an unauthorized join is possible. From the design of Ethernet network, expanding the network is possible by connecting a switch to another switch. Therefore, if an attacker gains access to the network, the attacker is permitted to expand the network for providing multiple access points (wired or

wireless) to the target network.

Initially, the issue can be solved by limiting the access points. Keeping switches and other network devices in secure and secret locations, providing the existing hosts and switches authorized connections (i.e. with IEEE 802.1X [23] and EAP [40]), and by maintaining the unused wall sockets to be disabled for connections. Nevertheless, the maintenance of such a setting requires multiple configurations in several different devices. This demands costly professional personnel and a considerable work effort.

In SDN, the solution would be similar, but it would require less effort. Since the controller is centralized, switches' ports can be controlled off and on from a single point, and the connectivity status to all hosts can be followed. Furthermore, the authorization of hosts can be provided by an application running over the controller, which also authorizes switches when connecting to them (i.e. OpenFlow uses TLS [41]).

VLAN join, tagging and hopping. Legacy Ethernet switches can listen and advertise its VLANs with VLAN management protocol messages from any of its ports. This allows an attacker to make a host or a compromised switch to join all VLANs and access the restricted segment of the network. Even if a switch can be configured to not to advertise (its VLANs), the switch still listens to them and thus makes possible for an attacker to join a VLAN by probing. Furthermore, with tagging and hopping an attacker can access other VLANs. This is done by modifying packet frames with specific MAC and IP addresses.

Nevertheless, with careful configurations of individual VLAN segmentation in the network, using IEEE 802.1ad Q-in-Q double tagging [13], or by using vendor provided Private VLAN (PVLAN) technique [42] these vulnerabilities can be prevented. These methods have still the impact of creating overhead in Ethernet frames, reducing switching performance and restricting virtual mobility.

As for SDN, VLANs are not configurations created in switches, but they are software defined in the controller and flows are created in the switches accordingly. As a matter of fact, SDN removes the need to use VLANs as it is done traditionally. Therefore with SDN, there are no VLAN threats as in legacy Ethernet. Additionally, since programmability, SDN VLANs can be defined by port, IP, MAC, time, or any other variable where a flow can be identified. Also the combination of variables can be used, thus restricting the possibility of spoofing attacks.

Remote Access to the LAN. Using social engineering [43], one can access an Ethernet segment through higher layers. By compromising a host inside the target segment, the attacker can use the compromised host to access administrative services and commence the attack further.

In this case, the vulnerability is created with social engineering (i.e. Phishing). To

prevent them, the utilization of security services has to be conducted, such as firewalls, Spam filters, anti-virus, and anti-spy-ware software. Although, these services are not airtight and can be bypassed as new vulnerabilities are discovered. Another way is to educate users to be more cautious when using the Internet. It is worth to mention that, this issue is out of Ethernet's scope. As initially, the host has to have an access to communicate with the network.

In principle, SDN can not provide more protection against social engineering, but then, when a host is compromised, administrative access is not possible from a host (unless access is separately granted, which is not the norm). The main target of SDN is to isolate the control from the data flows, giving its host a transparent network.

Topology and Vulnerability Discovery. An attacker can send to a network messages to investigate the responses and to discover the topology and services provided by the hosts. From simply listening to broadcast ARP requests, an attacker can find out used host IP addresses, servers, and gateways where the hosts are connected to. The IP range used can be also investigated from the DHCP replies. With these bits of information, an attacker may scan throughout the network and reveal vulnerable hosts, enabling the attacker to access and do further attacks.

Obviously, this threat presents two issues: broadcast messages and network scanning. In legacy Ethernet, solutions are not directly provided to solve these issues individually, and the solutions are higher layer (layers 3-5) oriented. Against network scanning and intrusion there are solutions to prevent them such as firewalls, Intrusion Detection Systems (IDS), and Intrusion Prevention Systems (IPS). These mechanisms are designated to detect and stop intrusions and unauthorized traffic.

A solution to remove broadcast messages from a network with SDN is presented in Chapter 4. As it turns out, with SDN, it possible to remove broadcast messages from a network without affecting the performance or functionality. For network scanning, SDN presents again a way to overcome it. As presented in Section 2.4.1, with FRESCO Reflector Net application, the network scanning attempt can be detected, prevented, and redirected to a honeypot for further study. Additionally, since the prevention was implemented in the controller with an application which simply follows failed TCP connections, there is no need for costly IDS and IPS devices. Hence, as SDN can make the whole network to function as a firewall, with possibility of IPS and IDS capabilities, it shows that programmability brings many benefits to customize the network to your own needs.

Break-Ins. When an attacker has the information of a vulnerability in a host, an attack can be executed using Ethernet as a medium. Usually, this appears in the target as a normal network session, thus the attack remains undetected. Here, as with *Topology and Vulnerability Discovery*, a firewall, IDS, IPS can be used to prevent some of the break-ins. Once the attacker has broken in to a system, it can not be prevented from Ethernets' scope anymore and thus it falls down to the computer

security system, which is out of this thesis's scope.

Generally, with SDN, host break-ins are also hard to detect. If an attacker has found a host and its vulnerabilities, then it is likely that the flow created for the attack was successful (i.e. the attack comes from a trusted source). In this situation, SDN has little to do since its main operations are on the lower layers (2 and 3).

Switch Control. When switches are newly installed, as default they are provided with a default password (or none) for accessing the administrative tools and services. If (1) the network administrator was careless and did not change the default password, (2) the attacker gained the physical access to the switch and reset the password with physical reset, (3) attacker gained the password by stealing, (4) or found a vulnerability with the switch, an attacker can access and control the switch. The attacker could reroute the traffic by shutting down links, by claiming STP root, or by performing a Denial of Service (see Section 3.4) attack to selected links. Nonetheless, a legacy Ethernet switches are generally limited by its hardware and OS to only perform switching and flow control actions. While, performing eavesdropping or generating spoofed frames are usually not possible. Anyway, the attacker can make a switch redirect or mirror its traffic to the attacker, which makes eavesdropping possible and also unauthorized VLAN access.

A switch is preferred to be placed in a secure and secret location. This way an attacker can not perform a physical password reset, thus making switch control attack harder. The administrative tools and services should always be kept behind passwords, and preferably if it is possible, behind authentication services with certificates such as TLS, SSL, or anything similar. This way the intrusion into a switch is limited and restricted, making an attacker harder to gain the control of the network.

In SDN, a switch is tightly connected with its controller. Without the controller, the switch is dumb and thus can not act on any network. Nevertheless, a SDN supporting switch is protected with password in the same manner as the legacy switches. The hardware of a SDN switch is more powerful than in a legacy switch, since handling flow generation and matching packets requires more computation power. This means that a SDN switch has more abilities to tamper traffic and get control of it. Assuming that an attacker gained the control of an SDN switch, he could look up the generated flow rules from the flow rule tables and save it for later usage and then change the switch's controller to the attackers' own controller, which shall keep the same flow rules available and mirror every traffic to the attacker for eavesdropping. Obviously, the complete control of the network (switches) belongs to the attacker.

As in legacy Ethernet, the best advise is to isolate the hardware in a secure and secret location. In SDN, it is recommended to isolate physically the control network from the managed data network. Else, in the case of logical control traffic isolation, the attacker could eavesdrop switches' control traffic by passively listening to inter switch links and in addition try to connect to the administration interface. Clearly,

physical isolation is necessary for a secure SDN network. In addition, OpenFlow provides means for a switch to authenticate to the controller and therefore provide the assurance of a reliable switch. As for unauthorized administration access, a possible solution could be a mechanism where the switch notifies the controller of an access attempt providing the controller to act upon it.

An additional thereat to the survey [38] has to be discussed. As SDN differs not only of its programmability, but also in its physical and design structure. The center node of a SDN network is the controller, which brings one more threat to the network and system access topic: **Controller Control**. The SDN controller is not currently standardized and existing implementations vary in many levels. More importantly, the controller is a network device, practically a computer, which possesses multiple Network Interface Controllers (NIC) with multiple ports for efficient switch connectivity. In case an attacker compromises the controller, the ultimate control of the whole network managed by that controller is accessible. As critical as it sounds, this calls for drastic measurement to protect this central orchestration unit. Since the controller is a computer, it is highly recommended that security levels of accessing this device is with the highest priority.

This is currently the weakest point of SDN. As there are alternative ways of implementing a controller, one solution can be the distributed controller platform, such as in Onix [1]. Similarly, the FlowVisor [44] presents a platform to have a controller to multiple controllers.

3.2 Traffic Confidentiality

Since confidential messages and information are transmitted in the network, one of the biggest interests of an attacker is to listen to the traffic (eavesdropping). In legacy Ethernet network, a switch sends to a host its own traffic, broadcast messages, and frames which are flooded at the time of the switch's MAC table timeout [45]. In addition, a switch floods frames in case of a unknown destination. This means, an attacker can generate frames with random destinations to overwrite switch's MAC table and let the switch flood all frames, thus enabling eavesdropping.

In SDN, broadcast messages can be removed and random flooding does not happen except if the controller has applied such a mechanism in the switch. By default, in SDN, unknown traffic is directed to the controller for further processing. In the case of the controller not knowing the recipient, the controller can flood out a broadcast message to determine the location of the unknown host. On the contrary, if the controller is implemented in the manner to keep track of all its connected hosts, this situation never comes up. This is further elaborated in Chapter 4.

Another way to eavesdrop is to overwrite a MAC table entry by sending a frame with a forged sender address. This case is illustrated in Figure 11. Even thought

the attacker can gain targets identity, maintaining it can become challenging. As if the target is on-line, there will be a race condition in the switch and eavesdropping can be performed only randomly. Whereas, if the attacker has the knowledge that the target is off-line, eavesdropping is possible.

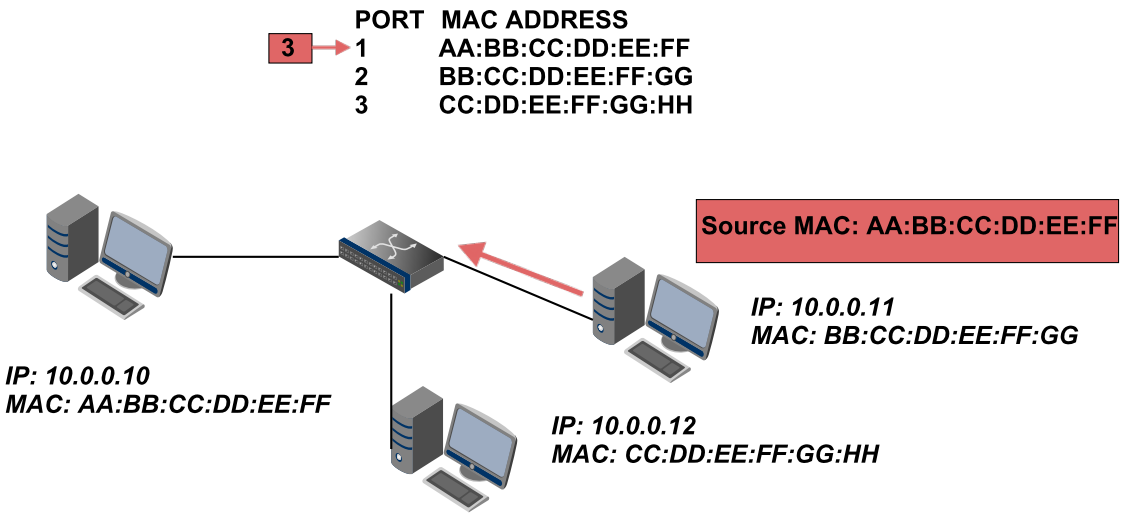

Figure 11: An example of MAC spoofing, where the host 10.0.0.11 pretends to be the host 10.0.0.10 by sending a frame with spoofed MAC address which updates the switch's MAC table.

In SDN, MAC spoofing would be similarly possible, but in the case of a race condition, there is a possibility to create a policy to react to such cases. Although, an authentication mechanism for the network access can restrict MAC spoofing possibilities.

An efficient way to eavesdrop is to perform it passively. **Passive eavesdropping** is performed between two host, two switches, or between a host and a switch. The attacker attaches a listening device in the medium (i.e. cable) and collects everything that goes through. Since there is no architectural change in the network, passive eavesdropping is fairly hard to discover. This threat affects both SDN and legacy the same way. Yet, a slight advantage in SDN is that it can prevent a switches flooding any traffic and hence bypass passive eavesdropping.

3.3 Traffic Integrity

The integrity of the traffic is what the users rely on for genuine networking experience. An attacker, after gaining the control of traffic, can modify and imitate reliable services and acquire confidential information from the host or the user. This

is a threat both in SDN networks and in legacy networks.

ARP and DHCP Poisoning. Due to broadcast messages sent in DHCP and ARP protocols, an attacker can acquire the advance to redirect hosts' messages to a fake gateway and DNS, and to capture traffic intended for someone else. The first mentioned is called DHCP poisoning, where an attacker will race the DHCP server to respond to a host's requests for DHCP information (with a broadcast message). The attacker tries to be the first replier and thus provide the confused host with a fake gateway and DNS information. The attacker can then imitate real network services and receive from the host confidential information. The latter advance is called an ARP poisoning [46]. Due to the concept of ARP protocol, a host can listen to ARP replies sent by others even if the listening host did not send a request for it. Using this vulnerability, an attacker can simply send an ARP reply message to a host with forged source IP (to represent some other host or service), and source MAC address. In Figure 12, this case is illustrated.

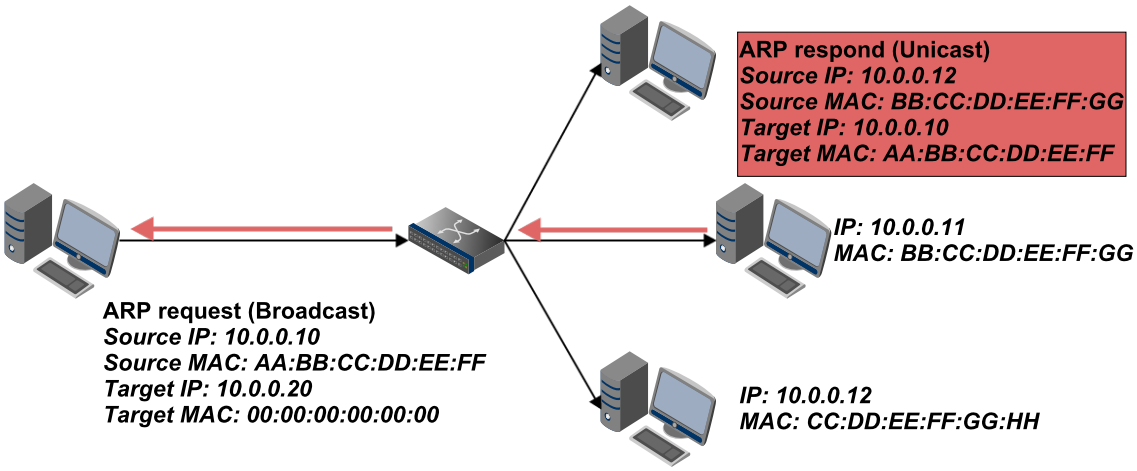

Figure 12: An example of ARP spoofing, where the host 10.0.0.11 poisons host's 10.0.0.10 ARP table.

Since DHCP poisoning is an attack performed with a race condition, it can be difficult to detect it afterwards and recognizing a genuine DHCP server requires prefabricated measurements from both sides (host and server). As for ARP poisoning, DHCP snooping and S-ARP [47] are possible solutions for prevention. With snooping DHCP information of a network, the MAC and IP addresses can be tied with the corresponding port on the switch. Whereas, S-ARP provides cryptographic authentication measurements to identify hosts' identities.

With SDN, as explained before (see Section 3.2), by removing broadcast messages, the host can be secure from these threats. Although, ARP poisoning might be possible if an attacker has access to the same Ethernet segment and know the IP

and MAC addresses of the target host. A possible prevention of ARP poisoning can be acquired with mechanism where the host is modified to accept only ARP replies from a controller or the controller orders the switches to drop all ARP reply messages from an unauthorized sender.

Man in the Middle (MITM). Using ARP or DHCP poisoning, an attacker can perform MITM attacks. The attacker needs to either poison two hosts ARP table to direct the traffic intended between the host through the attacker, or imitate a DHCP server to create a scenario where the attacker is a proxy (hidden from the host) between the host and the real network. MITM attacks are mainly to performed for eavesdropping or to for tampering and modifying traffic. ARP or DHCP poisoning can be contained the same way as mentioned before.

Another way to perform MITM attacks. As seen from Figure 13, by connecting two switches which are in the same network to the attackers' switch, it will create a loop. As LAN's are tree topologies, loops are prevented by STP and as the attackers switch claims to have the highest priority, the attackers' switch can claim the root position. This way the attacker can eavesdrop or modify traffic which flows through the root switch.

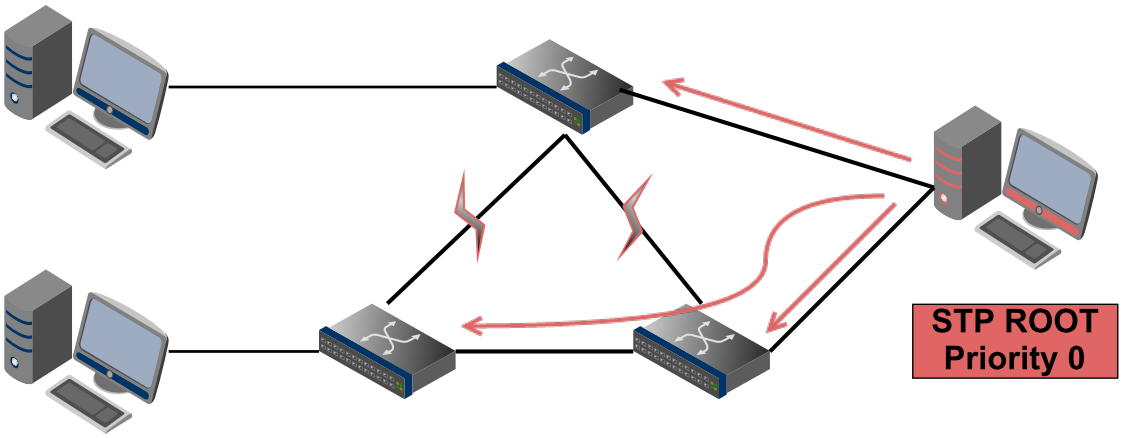

Figure 13: An example of Man in the Middle attack, where the attacker captures STP root and divides the network into two splits where the the traffic flow through.

STP uses Bridge Protocol Data Unit (BPDU) messages to select root. Therefore, to prevent STP root attack, a BPDU or root guard can be implemented. A BDPU guard blocks all STP messages from a designated port. This way network administrators can reserve the ports to be used by inter-switch connections and use BDPU guard on the other ports. Another tool is called STP root guard, which allows STP messages from a port but prevents claiming root. [48]

In SDN, the controller has the topology knowledge and hence STP is not needed. The controller can decide the best route for each individual flow according to desired requirements. This leaves SDN out of scope for MITM attacks with STP. Nevertheless, MITM attacks with *passive eavesdropping* is still possible in SDN networks, as explained previously (see Section 3.2). If the network contains legacy segments, guard functionality can be supported by allowing the BPDU messaging in the flow rules.

Session Hijacking and Replay. Many higher layer protocols create sessions to handle their communication services. If an attacker has a possibility to do a MITM attack and eavesdrop the traffic, the attacker can collect information about the sessions and thus recreate it or hijack it. The attacker can then pretend to be one of the endpoints and use the service disguised as the other endpoint. To quiet down the pretended endpoint, the attacker can simply execute a Denial of Service attack (see Section 3.4) or divert the traffic. A *session hijacking* attack can be used to break in to a service which requires initial authentication but no further verification [49]. An authenticated session can be recreated if an attacker gained packets which are used for initial authentication. This is called the *Replay* attack. Even though the packets might be encrypted, the attack does not require more than the knowledge (or a guess) of the content, and the attacker can reuse it without altering it.

As sessions are created in the higher layers, these attacks can not be prevented on the Ethernet layer more than a MITM attack. To prevent session hijacking and replay attacks, higher layer protection services have to be used. Services which promotes encrypted connections and timestamps (such as MACsec [50], SSH [51], or TLS [41]).

In SDN, these attacks can not be directly addressed. As the session is created, it is assumed trustful from the lower layer connection, thus leaving it to higher layers to take care of the authenticity. Although a SDN application could be created to follow up sessions, triggered and terminated by the host. This solution would require host to controller connectivity, ergo attention is needed for protecting other arisen threats against the solution. Generally, as in this threat it is assumed that MITM attack has succeeded, SDN has evenly a little role to stop the attack as the Ethernet has.

3.4 Denial of Service

For disrupting traffic or networking performance, an attacker can perform a *Denial of Service* (DoS) attack. It can be performed either to suspend or interfere with a service. At the hardware level, a DoS attack can be performed by cutting a cable or by harming a network equipment(such as a switch). However, a DoS attack in layer 2 can be more damaging and structured [38].

Resource Exhaustion Attacks. When a switch makes forwarding lookups and

forwards traffic, it requires from the hardware computing power and resource allocation. The switch's management and control traffic requires some additional processing and handling, especially when the configuration is being changed. Since hardware has its limitations, it is possible that the switch might get over exhausted when performing many additional computations requiring tasks. Also, the nature of the incoming traffic to the switch, has a big impact on how much it enervates the switch. Additionally, a switch might be set to process a certain type of data plane traffic (e.g. for logging purposes), making it vulnerable to exhaustion attacks (overloading). This can occur, if the attacker knows (or guesses) the switch configuration and starts to overload the switch by flooding a particular kind of traffic to the switch. When a switch gets overloaded, the performance to process decreases and the traffic gets slower or it stops.

Initially, Ethernet has not addressed this attack directly, in SDN a resource exhaustion attack can be controlled with decreasing traffic on a certain node or blocking misbehaving port completely. Since the management and control are decoupled in SDN the computation is outsourced and switches have only behavior instructions regarding the traffic. Nevertheless, a switch can be overloaded if the switch is set to process certain frames (OpenFlow operation *set*) before forwarding them. Ergo, by limiting the traffic (by dropping packets or closing the port), or by redirecting the traffic through other path, resource exhaustion attack can be controlled.

Protocol Based DoS. In legacy Ethernet, the network topology is physically a mesh figured network. With STP, the topology is transformed logically in to a tree topology. As explained in MITM attack (see Section 3.3), an attacker can add a fake switch to the network and disrupt the whole network by flooding useless traffic [52].

Since STP is not used in SDN in the same manner as in the legacy Ethernet LAN, SDN falls out of scope for this threat. Furthermore, the concept of SDN leaves the network functionalities completely to the controller and its applications. Thus, SDN networks can be built completely from the scratch, discarding all existing protocols. The network functionalities can be designed specifically to satisfy a specific type of network (LAN). In this sense, DoS attacks which are performed by misusing a specific protocol's vulnerability, can be completely disregard in SDN networks. It does not mean that SDN prevents DoS attacks, but it makes them certainly different.

3.5 System security

Building complex networks requires careful design of configurations. For every network device, precautions have to be taken against security threats while preserving network functionalities. To enable certain functionalities, the right configurations and hardware installations are in a network device essential. **Configuration and Installation Issues** may arise, as humans occasionally tend to make mistakes. An attacker can misuse or exploit network vulnerabilities in case of gaining the infor-

mation about a configuration or installation mistakes.

In legacy Ethernet network, every switch needs a separate installation and configuration procedures. Thus, making it essential to carefully prefabricate configurations and professional dependable installation. On the contrary, SDN requires only careful hardware installation and a simple configuration (e.g. locating controller). The rest configurations are done by the controller, hence, configuration mistakes are simpler to repair.

Implementation Issues and Vendor Extensions. As Ethernet control and management are not defined in detail, it leaves room for implementation. For example, vendors use this freedom to provide own proprietary solutions and services. New implementations often have undiscovered weaknesses or issues and thus provides an attacker with the possibility to find them.

Although, in SDN the control and management configurations are not implemented in switches, they still are still at risk of being exploited. Nevertheless, as these exploits are discovered, it is much easier to repair and improve them, as the implementations are in the controller. Vendors can easily update their software by updating the SDN application.

In the paper [38], two additional issues were defined regarding system security: **issues with Legacy Technology** and **Architectural Issues**. These issues are both out of SDN's scope to solve. The issues are reflecting upon the Ethernet's extensibility (e.g. with software and equipment) and its relationship to upper layer protocols which assume Ethernet is protected. Usually, protocols and communication technologies are embedded with security mechanism which requires implementation on both ends of communication, relying on the middle ware to be secure.

3.6 Summary

In summary, SDN can overcome many Ethernet security issues, yet undiscovered SDN vulnerabilities bring new threats to the network. Nevertheless, the advantage of SDN is that the network is centralized. Thus, managing the traffic and the security of the network is done efficiently from a single point. As opposed to the legacy Ethernet networks, where the management is distributed and hence require more complex solutions.

Due to the architectural aspect of how SDN networks are constructed, protocols such as STP and VLAN can be completely discarded, and hence their security issues are not anymore relevant. Even the issues troubling host connectivity protocols, such as ARP, can be surpassed. A SDN network is aware of the status of the network structure and of its passing traffic. This helps improving the network security while preventing traditional methods to of attacks. It is noteworthy to mention that the SDN suffers of a single point failure issue. Meaning if, the controller or a SDN appli-

cation is compromised, the whole network is under the control of the unauthorized party.

Each issue concerning Ethernet was briefly discussed, and compared if SDN is concerned with the same issue. Generally, by adding authentication, encrypting the traffic, and securing device premises, network security is can be enhanced significantly. Yet, the issues which come with networking protocols, have to be addressed separately.

4 Example Implementation: Broadcast Remover

In this section, an example SDN implementation is introduced. The aim of the implementation is to increase the status of Ethernet LAN's security with SDN provided capabilities. The section will start by defining the goals of the implementation along with the issues which it is going to address. Continued, by describing the design and the architecture of the implementation. Following by a comparison with the existing SDN security enhancing solutions presented in Section 2.4. Finally, the implementation will be evaluated concentrating on the possible focus points for further work.

It is worth to mention that this implementation, is only theoretical and it requires extensive testing and benchmarking.

4.1 Goals

The implementation represents a SDN solution to improving legacy Ethernet LAN security. The solution aims to tackle known broadcasting issues which are IPv4 based protocols such as ARP and DHCP, IPv6 is excluded from the design. The target is to implement an application (called Broadcast Remover) and design it to run on a controller which controls a set of switches in a LAN. The goal is to implement Broadcast Remover from multiple modules where each module has its responsibility. For instance, one of the modules will handle the authentication of the hosts connected to the network and another module will manage the broadcast messages.

Additionally, another aim of Broadcast Remover is to be backwards compatible. Meaning, that connecting hosts (users) do not have to change any of their hardware or add new additional software. The host communication with other hosts will be as transparent as in the legacy but only more secure.

As seen in the previous Section 3, the legacy Ethernet LAN has multiple security issues which need critical attention. In the section, every issue was provided with a brief discussion whether SDN can offer a solution or not. At this point, a solution will be provided with a practical SDN environment design. The target is to utilize SDN properties to improve Ethernet LAN's security.

4.2 Broadcast, the threat?

As the name of the solution indicates, the main target of the Broadcast Remover is to address mainly the threats that come along with Ethernet's broadcast messaging feature. Additionally, it aims to tackle the switch's over exhaustion DoS threat and eavesdropping possibilities to switch's flood traffic (see more about DoS in Section 3.4 and about eavesdropping to switch's flood traffic in the Section 3.2).

However, the broadcast messages have their initial purposes, and therefore, the Broadcast Remover will not essentially remove broadcast functionality, but it will change those broadcast messages which unnecessarily expose host's identity to more secure alternative. The purpose is not to decrease the networks functionality but to increase its immunity and security to external menace. Hence, the purposes of using broadcast messaging has to be investigated to see whether the same functionality can be provided likewise utilizing SDN properties. In principle, the purpose of broadcast messaging can be divided in to three types: (1) to locate a host in the network (i.e. ARP), (2) to provide a service (i.e. printer), and (3) for personal use.

For the first type, the possible security threads were discussed in Sections 3.1 and 3.3. Protocols such as ARP and DHCP use broadcast messages to find hosts and available services in an Ethernet LAN. The principle being that a request is always answered regardless of the requester, also, a broadcast message is always received despite the sender. In other words, anyone can detect hosts in the network by probing different IP addresses. Also, using ARP and DHCP poisoning, an attacker can perform Man in the Middle attack and hence eavesdrop passing traffic. ARP poisoning is an attack where the attacker alters the victim host's ARP table and the DHCP poisoning is where the attacker races the DHCP server to respond a host's request for DHCP server information.

The solution for the first type is to transform the broadcast messages into unicast messages. Originally, in legacy Ethernet LAN the host's location and hardware identity was not necessarily known by any part of the network, hence, by sending a broadcast message to all connected devices and trusting their honesty, hosts could find each other without a problem. However, now in SDN it is possible to recognize each connected device in the network and redirect the traffic directly to its destination without the need for locating the host.

As for the second type, a service which uses broadcasting is usually designed to be known by everyone connected in the network. Therefore, Broadcast Remover will allow these kind of messages to be broadcasted. Obviously, a white list of allowed hosts for broadcasting has to be implemented. Additionally, to protect the misuse of spoofing, only the identified host are allowed to broadcast. For example, by allowing only a certain host from a certain port with known identity (such as MAC address) to broadcast.

The third type is a type of a protocol which uses broadcasting for its own reasons. These types of usage have the same position as the second type, a white list implementation.

4.3 Authentication possibilities

The implementation will manage the connected hosts by authentication. Authentication is not a requirement for operation but it has a great impact on increasing security. As authentication aggravates the difficulty of a host identity spoofing, it improves additionally the integrity of the traffic.

As for authentication possibilities, there are two; (1) MAC address based, and (2) Web server based. The first has the advantage of simplicity in implementation, as the second is more secure and reliable. The MAC based authentication depends on a white list of allowed MAC addresses to communicate in the network. A module responsible for authentication can either keep updated flow rules in the switches for each host or create the flow rules when a host appears. The latter is more dynamical as the size of the flow rule tables in the switches are limited. Although, creating flow rules on demand requires more responsibility from the module to recognize the host when it connects to the network. However, the web server based is implemented with allowing traffic initially only between the authentication web server and the host. The web server will provide an authentication interface requiring host's identification. After successful identification, the web server notifies the module responsible for authentication to create flow rules for the connected host.

4.4 Design and Architecture

Broadcast Remover is designed according to the SDN principle: a network of switches which support OpenFlow, the controller which controls the switches while providing information and a managing interface for the applications, and the applications which handle the tasks related to the network. In the Broadcast Remover, the tasks are divided into logical ensembles which are indicated as modules. The Figure 14 illustrates the design of the implementation.

The **applications** of the Broadcast Remover can be implemented in several different ways: (1) one application which runs on the controller and where each of the module is implemented as a separate function, (2) each module implemented as its own application, or (3) a combination of modules in several applications. In the first case the simplicity of the inter-module interaction becomes as an advantage. In other cases, the controller has to provide an interface for interconnectivity between the applications. Nonetheless, multi-application solution can be easier to manage as a failure in one of the applications does not necessarily stop the other applications. The purpose and responsibilities of each module are described in following subsections.

The **controller** of the Broadcast Remover represents a basic SDN controller as described in Section 2.1. Only depending on how the application layer is implemented, the controller may have to provide an inter-application communication interface.

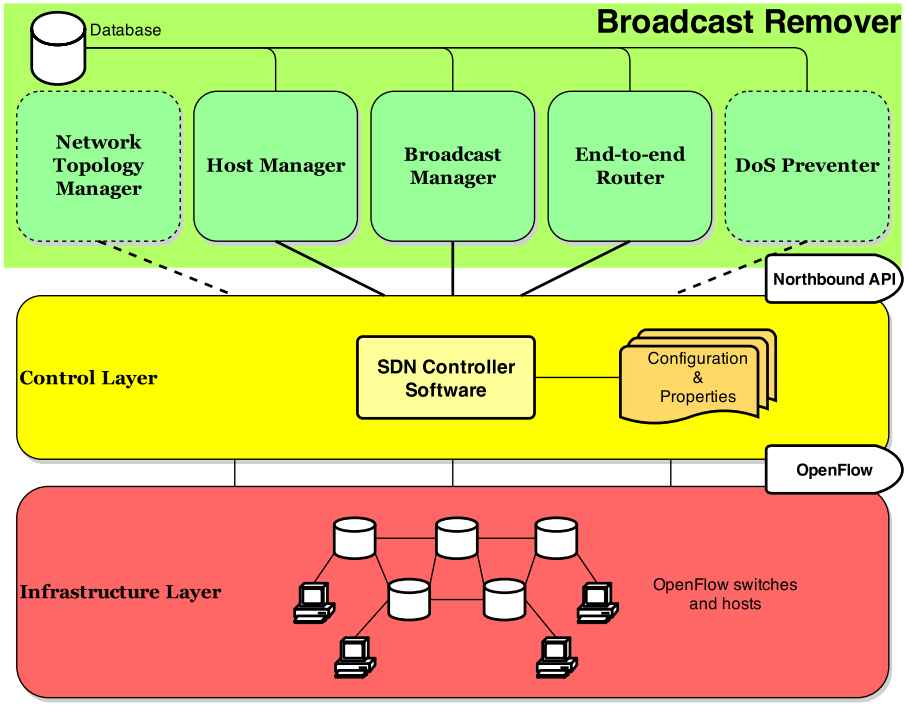

Figure 14: The design of the Broadcast Remover. The Network Manager module (see Section 4.4.1) might not be needed if the controller provides it. The DoS Preventer module (see Section 4.4.5) adds optional security and therefore not required for implementation.

Additionally, the controller can offer the construction of the network topology provided with the reliability of the switch identity. Optionally, if this is not provided by the controller it can be implemented as one of the applications or its modules.

As for the **switches**, the connection between the controller and the switches has to be secured both physically and logically. Meaning that the wiring and devices are located in secure domains and they use TLS to encrypt the management communication between the switches and the controller.

When a switch picks up traffic which did not match any flow rules, the switch can send the unidentified frames to the controller, which should trigger the controller to generate new flow rules concerning the unidentified traffic. The switch can alternatively just drop unidentified traffic and allow only the traffic specified by the controller. Both ways are possible, yet they differ in the way of handling new connections and traffic. To examine which way is more convenient, the process where a new host appears has to be studied.

Now, as a host connects to the network, the identity is unknown before the host communicates. Hence, it is possible that the connected host is vicious (e.g. passive

listener, misbehaving host) and therefore the traffic from the host must be filtered and allow only a certain type of traffic to initially verify its identity before more privileges is given. Alternatively, problems may arise if all traffic is directed towards the controller, such as the threat of a DoS attack.

One possible solution of a switch sending unidentified traffic to the controller would be that only the first frame is sent to the controller and the switch drops everything else. If the traffic comes from a trusted source, the controller should either react or not respond. This now requires that the first frame sent from the host should have at least the host's MAC address to be able to verify the host. To minimize spoofing attempts, the switch will allow only one frame per physical port connection per connection session, contrary to one frame per a MAC address.

Another solution of a switch having predefined flow rules would be to modify manually permissions for the statically identified hosts, while everything else is dropped, expect the DHCP request (for host identity check). Clearly, this way the network favors non-statically defined hosts, yet it can be very inefficient if the network has many statically identified hosts. Anyway, Broadcast Remover will take this solution as a default behavior at all the switches, as it is simpler to implement and it is less vulnerable to unidentified traffic. This also ensures that initially the network is not available for any kind of traffic until the controller permits it. This is considered as a security benefit while it prevents for example the passive eavesdropping when it is done by directly connecting to the network (see more about passive eavesdropping in Section 3.2).

The **hosts** will connect to the switches and they do not require any modification in their existing networking behavior. As a host connects to the network, it connects and performs connectivity procedures the same way as in a legacy Ethernet network.

4.4.1 Module: Network Topology Manager

The network consists of switches with OpenFlow support. Using OpenFlow, switches provide an interface to control them. Hence, the switches are recommended to be connected to the controller securely, so authentication can be used for reliability to have a trusted controller for the switches and vice versa. This way, the controller can generate a list of connected switches and provide an interface for the applications to communicate with them.

The Network Topology Manager module will collect periodically the list of the connected switches from the controller and it will requests every switch to provide its current connectivity information (port status). The module will then probe each port of each switch to see which ports are connected to another switch and construct the network topology accordingly. When there are changes in the network topology, the module updates its information. The switch probing can be done using

Link Layer Discovery Protocol (IEEE802.1ab LLDP [53]). As LLDP is a standard and used also by legacy switches, the switches which are not OpenFlow supported may respond to the probe but they are not included in the network. For security reasons these port should be blocked as they present a possibility to perform MITM attack (explained in Section 3.3). The network should consist only of OpenFlow supported devices which are connected to the controller securely. In case needed, the module can support manual addition by the network administrator.

The module then provides other modules the list of connected switches and their port statuses along with the topology information. Using these information, the other modules can give flow rules directly to a precise switch and route traffic.

This module should be in principle included in the controller as the network topology is commonly needed for many purposes. In case the controller has a topology discovery functionality provided with security mechanism needed, then this module becomes obsolete. Nevertheless this module is the first module to be initiated to make network functional.

4.4.2 Module: Host Manager

The task of this module is to provide hosts with an IP address, authenticate unidentified hosts, and to keep module's database updated. It cooperates with Broadcast Manager (Section 4.4.3) and End-to-end Router (see Section 4.4.4) modules in the case of host detection or flow rule creation. The Host Manager module maintains a database which consists of both connected and disconnected hosts in the network. Additionally, the module provides an interface for other modules to have access to the database. Furthermore, it is the responsibility of this module's to provide the authentication measurements according to the security level choices, either with a white list of allowed MAC addresses or with an authentication Web server.

As **a host connects** to the network, typically, an IP address for IP communication is required. The host either requests it from a DHCP server or it is spontaneously defined by the user or a stateless auto-configuration [54]. Primarily, the connection starts out on the physical level. The switch is able to recognize on the connected interface that a connection on the physical level is established, but on the link level the connection is not made before the host transmits any traffic indicating its identity. Usually, it is a broadcast message to prevent identity conflicts (i.e. ARP announcement [54]) or to locate a DHCP server with a DHCPDISCOVER message [16]. In either way, all broadcast messages are handled by the Broadcast Manager module (see Section 4.4.3), which triggers then this module in the case of an IP address request. This module will then perform as a DHCP server and answers to the host accordingly.

Host's which have defined spontaneously its own IP address can not use the network

unless the administration has added the host manually in the module's database. The administrator needs to provide the IP and MAC addresses, and additionally the port location where the host will connect. Also, multiple connection points can be defined to assure the mobility of the host. The module generates then flow rules for the switches where these hosts can appear to enable communication for the host (such as host-gateway for Internet connection). According to the need, End-to-end Router module can be triggered to enable intra-network communication. Nonetheless, these manual additions are compromising the security of the network, as an attacker can spoof the identity of these hosts and misuse the communication. Proving the identity of an host is without authentication almost impossible. Hence, the web based authentication can offer the reassurance of the identity of the host.

In the case of a host which is connected to the network but no traffic can be detected from its interface, can be in the worst case a passive listener trying to sniff the traffic. This threat is void as the initial state of the switches is not to push any traffic out of any port before the controller allows it. Alternatively, an authenticated host which has privileges to communicate can be disconnected while in the same port a spoofed host is reconnected with the same identity (MAC address) as the privileged one. This is a major MAC spoofing threat (see Section 3.2) which can be prevented if web based authentication is used. With web based authentication, every disconnection requires re-authentication, hence limiting spoofing possibilities. Although it might generate some frustration on the user side in case of accidental disconnection, but the security benefit makes it worth it.

As discussed earlier, the **authentication** can be implemented in two ways, either web based or MAC address based. In the case of MAC address authentication, the module holds a white list of allowed MAC addresses which will be allowed to communicate. The authentication happens automatically during IP address request as the IP address is only given to those in the list. Others who request an IP address will be banned by creating a flow rule to drop everything which comes from the port with its MAC address. This is clearly the less secure way to implement authentication as spoofing a MAC address is fairly simple. Also, a DoS attack to over exhaust the system or searching allowed MAC addresses is possible. With probing different MAC addresses the system will eventually allow a MAC address found in the list or keep banning every not allowed MAC address, which would exhaust the system. Nevertheless, to prevent over exhaustion, an attempt counter can be implemented. Here the module registers the port where the authentication request was issued and calculate the frequency of authentication requests per port. After a threshold, the module can quite down the port for a predefined time. This task of preventing the threat of DoS can be outsourced to be handled by the DoS Preventer module (see Section 4.4.5), but for simplifying purposes it might be more efficient to implement it directly into this module. Moreover, if the host has its own IP address already defined, this authentication method becomes obsolete. To increase secureness, the web based authentication method is recommended.

The procedure of authentication with the web based differs. After the host has gained an IP address, the module will first trigger End-to-end Router module (see Section 4.4.4) to create flow rules for the host to allow only frames from the port at the switch where the host is connected with the host's MAC and IP addresses to communicate with the authentication web server at protocol port 80 (HTTP). The same procedure is done to the hosts with predefined IP addresses. Any other traffic is set to be dropped. Now the only action what the host can perform is to authenticate with predefined user identification such as a login and a password. Security threats in this case are not too obvious. Spoofing is only possible if the identification is stolen in addition to the MAC address. If irrelevant traffic is sent extensively, it might over exhaust the switch. Nevertheless, the avoidance of this threat is handled by the DoS Preventer module (see Section 4.4.5).

Regardless of which authentication method is implemented, upon successful authentication, the module will trigger End-to-end Router module to create the flow rules to serve the hosts needs or privileges, according to the configuration choices (will be discussed more at Section 4.4.4).

At this stage, an additional host behavior situation must be considered: the changing of physical location in the network. As it was discussed, with every reconnection a re-authentication is required, regardless of the location in the network. Hence, if a duplicate host is detected, it might have two possible reasons: (1) the host has moved and the database has not been updated of its disconnection, or it is (2) a spoofing attempt. Regardless of the reason, a check up upon the original host has to be made. A sufficient way to check would be to see the state of the port where the host was previously connected. If the connection is still active, the connection duration meter is read to verify if the connection is older than the time of the last update in the database entry. Generally, the database should be as accurate as possible and the delay of updates as short as possible. Obviously, when the original host is found then the duplicate must be a spoofing host. If the host is no more connected in the original location, then the new connection can be accepted. Again, it might be a spoofing attempt, which can be hindered with the web based authentication, yet one more option is available to verify the reality of the situation. In the case where the new location is far from the initial location, it is assumable that the transition between the location can take time. Hence, an additional feature of a transition delay meter for port changing could prevent spoofing attempts made too hastily.

The **database** of the module contains the information of the connected hosts and if demanded a list of allowed and banned hosts. This database can be used for monitoring and managing. It has potential to be used in extended modules which are not presented in Broadcast Remover. The database is accurate in respect of the host addition as it is linear, yet the hosts which are leaving must be triggered externally. The module can perform periodically a check up upon all the switches and their port statuses in addition to the connection duration meter readings. Additionally, an OpenFlow switch can be configured to notify the controller upon port status

changes. The module can either poll the controller or the controller is set to trigger a database update function which will refresh the database entries. A downside with this can be a misuse or a malfunction of the physical connection, as every time the interface goes up or down it triggers the procedure which can eventually exhaust the system. A simple solution for this can be a counter to each port status changes. When the counter reaches a threshold, the administrator can be notified of a failure and the switch reconfigured to shut down the misbehaving interface.

It is also notable that this implementation requires that a host should have only one identity (MAC address) per port (the switch interface) per connection (on the physical layer). Inevitably, the restriction of single identity per port serves for the cause of preventing unauthorized network extensions. As discussed in Section 3.1, one of the vulnerabilities of the legacy LAN is its extensibility with additional switches. Simply by connecting a switch or a WLAN router to the network can add additional access points. Although, expanding the network with a NAT featured network device could hide unauthorized access to the network. The issue is a known downside, as detecting NAT devices can be complicated, yet not impossible. The module can be extended to support such a feature, either by implementing an existing solution, such as sFlow [55] or by creating an interface for a third party product. Nevertheless, in case of such an event, the right procedure is to ban the port (the switch interface) by silencing the port and additionally notify the administration.

Table 2: The suggested structure format of the Host Management module's database.

#	MAC address	IP address	Auth status	Last seen switch(port) time	connection start time	Additional info
1	AA:BB	10.2.1	true	sw12(22) 15:12.44 15.04.2014	10:23.33 01.01.2014	Example of authenticated host
2	BB:CC	10.2.2	false	sw03(04) 09:11.10 16.04.2014	23:22.13 02.03.2014	Example of unauthenticated host
3	CC:DD	N/A	false	sw10(15) 00:01.55 18.04.1986	N/A	Example of unidentified host

The database is structured as a list (see Table 2), where each row resembles a host and its networking status. Each host's information and status is stored to its respected column. Each host (row) should have at least the entries on its MAC address, the location of the interface connected, time of the initial connection, and

last seen packet transmission. Additionally information on each host's networking status is essential for check up and monitoring measurements, such as authentication status. In case of web based authentication, IP address becomes obligatory as the host gains an IP address before authentication. Each host on the list is a connected host. When a host disconnects it can be either removed or saved on a separate list. Having information of old connections is useful for logging and additional feature measurements. One example could be the use of port changing monitoring.

4.4.3 Module: Broadcast Manager

As the implementation's name indicates, this module plays the biggest role of preventing the major Ethernet LAN thread, the broadcasting messaging. As discussed earlier, the functionality of broadcasting can be useful and efficient, hence, removing the functionality would not be a solution. The solution this implementation thrives is to have more control over the broadcast messages and to handle them according to the need.

This module's task is to address all broadcast messages sent in the network. According to the situation, the module either prevents, allows, or modifies the broadcast attempt. Each broadcast message situation needs special attention to achieve the original functionality. At the switch, a broadcast frame can be recognized from its destination address (255.255.255.255 in IPv4). The purpose of the message can be separated into three different categories: (1) host management, (2) service advertisement, and (3) private usage. As a host sends out a broadcast message, the controller (the module) needs to identify the initial purpose of the message. To do this, the Broadcast Manager module can be implemented in two ways: (1) upon each broadcast message the switch sends it to the controller to get further instructions, or (2) the module generates rules for foreknown broadcast message types and situations, and when the switch receives a broadcast message which does not match any given rules it drops it. Inevitably, the second way is more efficient as it decreases the traffic amount at the controller.

Anyway, regardless of the broadcast situation, the initial rule is to drop all broadcast messages except those which matches one of the defined rules. Broadcast messages sent in the first category are related with host management. Specifically, ARP and DHCP messages. A switch can recognize an ARP message from the Ethernet frame's type field. Additionally, as there are several types of ARP messages, the only type supported by the implementation is the ARP request. Other types are ARP probe [56], ARP announcement [54], reverse ARP [57], and inverse ARP [58]. To distinguish between these types at the switch, the flow rule which accepts only ARP requests has to be defined. By examining the ARP packets fields, the switch can detect that: ARP probe has the sender IP address field set to zero, ARP announcement has the sender IP address equal to target IP address and the sender MAC address equal to target MAC address, reverse ARP has the opcode set to

number 3, and inverse ARP has the opcode set 8 or 9. Hence, to drop all unsupported ARP messages, the module sets flow rules to the switches to do the following comparisons: Ethernet type is 0x0806 (ARP [17]); ARP opcode is 1 (request); ARP sender MAC address is equal to the Ethernet frame sender MAC and is non zero and is not equal to the target MAC. This way only an ARP requests are sent to the controller (captured by the Broadcast Manager module). Similarly, DHCP message can be recognized from the Ethernets frame's type field to be 0x0806 (IPv4), and from IP packet the protocol is set to 17 (UDP [59]) with destination port 68 [16]. When such a message is recognized, it is sent to the controller to be further dealt with.

Now, as the controller receives an broadcast message sent from a switch, the Broadcast Manager module captures it. If the message is an ARP message, the module looks up from the Host Management module's database the MAC address of requested IP address and generates accordingly an ARP response message and sends it to the requesting host. Here it is trusted that the database is updated and reliable. Additionally, the module could request a database update either from the Host Management module or from the controller, but it can exhaust the controller if an update is frequently requested. Hence, relying on the Host Management module might be sufficient enough. However, if the captured message is a DHCP message it is sent to Host Manager module (see Section 4.4.2) who will then serve the host accordingly.

As for broadcast messages which belong to the second category, these are predefined broadcast messages, which purpose is to advertise its service through broadcasting throughout the network (i.e. network printer or network access storage). These hosts are manually added by the administrator to the Broadcast Manager module's allowed-broadcast list and when the host appears in the Host Management's modules database, the Broadcast Manager triggers End-to-end Router module (see Section 4.4.4) to create flow rules for the broadcasting host to be able to reach all connected other hosts in the network. Alternatively, if the broadcasting host can not authenticate, such as a printer, then the authentication value is manually set to true. Now to prevent spoofing attempts, upon each disconnection the value will be overwritten to false. This may cause frustration as the service is unavailable until the administrator enables it again. Hereupon, it is important to recognize the cause of disconnection and act against it.

Moreover, a periodical check of the connected host in the database must be applied in order to recognize whether new hosts have appeared, which either have broadcasting privileges or they can be included to receive broadcast messages. Other way to implement this is to make the Host Management module trigger the Broadcast Manager module after a host has authenticated. Additionally, a broadcast-to-multicast feature could be applied where the administrator can define a group of hosts which are allowed to receive broadcast messages from a host. Regardless, the broadcast privilege is set in a manner that only with the identity of the allowed host from a specific network location (port) is allowed to broadcast, hence minimizing connec-

tion misusing possibilities. Additionally, each host in the allowed-broadcast list has to have the flow rule configurations to specify which type of broadcast messages are allowed.

Lastly, the broadcast messages which belong to the third category are quite similar as to the second category. Both of these need to be manually configured. The purpose of these broadcast messages is a private manner, meaning that the implementation network can support any additional types of broadcasting attempts, but it requires a manual addition from the administrator and it can have more parameters to be configured such as the flow rule renewal interval, and allowed ports for broadcasting and listening. It is important that these broadcast features are configured carefully not to enable contradicting rules in the switch. After the configurations are enabled the module triggers End-to-end Router module to create and set the flow rules to the switches.

4.4.4 Module: End-to-end Router

The main task of the End-to-end Router module is to create flow rules at the switches in the network to enable communication between hosts. It is mainly used as a tool by the other modules to enable the desired communication paths between hosts. By default the module selects the shortest path between hosts, utilizing the network topology (provided by the Network Topology Manager module, see Section 4.4.1). The path is selected by calculating the distances using an algorithm, such as the Dijkstra's algorithm [60], or the Bellman-Ford algorithm [61], which are commonly used routing algorithms. The implementation does not have any requirements regarding the algorithm as the network is rather statical. Alternatively, if congestion avoidance is needed, a third party application can be used such as Plug-n-Serve [62] to select the paths.

For creating routes, the module needs the source host, the destination host, and the type of traffic which will be allowed. Therefore, when this module is triggered by the Host Management module to enable traffic between the host and web authentication server, the Host management module will have specify that only IP based traffic, with TCP and as destination port 80 is allowed towards the web server. Also, when the authentication was successful, the routes to the gateway and possibly other hosts in the network, the type of traffic which is allowed between the hosts and towards the gateway. The communication between the hosts is not a necessity, but it is an optional feature if it is known by the administrator that these hosts need to be initially connected. Otherwise, the communication between hosts is established when the Broadcast Manager module triggers this module upon a successful ARP request. Also, for broadcasting hosts to reach other hosts in the network, the Broadcast Manager module defines the source host along with destination hosts.

The established flow rules are valid for a predefined time. OpenFlow defines for

each flow rule two timeouts. A hard timeout for the maximum amount of time the flow rule is valid and an idle timeout for the maximum amount of time the rule is allowed to be idle. When one of these timeouts go on, the flow rule is deleted. The duration is recommended to be short to avoid expired flow rules allowing traffic in the network. Hence, periodically after the hard timeout, this module should check whether the host's connection is still active from the Host Management module's database. If the host is still connected, then the flow rule is renewed. Another way to manage the flow rules is to set the timeout to zero which means that the rules are permanent and when the host disconnects the rule is removed. This way might be more convenient as the selection of the optimal timeouts value can be difficult. Yet the recognizing the disconnected host in real time can be tricky. Either this module periodically checks the status from the Host Management module's data base or the Host Management notifies this module that a host is down and its flow rules should be deleted.

4.4.5 Module: DoS Preventer

This module is optional for Broadcast Remover. Its task is to increase secureness by preventing possible DoS attacks aimed at the network (see Section 3.4). At the switch, the traffic which satisfies a flow rule is considered generally harmless, yet in certain cases where the flow rule is set for the host management purposes, an extensive amount of traffic can have a harmful intention behind it. Specifically, as the host is allowed to request an IP address from the controller (managed by the Host Management module, Section 4.4.2), theoretically only one request is sufficient or in case of occasional packet loss some reattempts. However, if the host sends requests constantly, it can be considered as an attempt of a DoS attack. Other such cases are when the web based authentication is used and the host tries constantly multiple times to authenticate, or during ARP request the host sends constantly multiple ARP requests.

These attempts can be handled by quieting down the port for a predefined time. At the switch, each flow rule has a counter indicating the amount of times the flow rule was successfully matched. When this counter reaches a certain threshold, the module sends the switch a *quite port* command which prevents the port from interacting while keeping it connected. The previously mentioned cases need a threshold to be set and a period of time the port should be quite. This module then periodically checks the counters and acts when the thresholds have been exceeded.

Similarly, if the host sends constantly traffic which does not match any flow rule, it can be considered as an attempt of a DoS attack. Hence, not to ban every accidental traffic of a host, a threshold of a certain amount traffic from a host should be defined. The switches have also counters for traffic which did not match any flow rule, and these are also inspected during the periodical check.

4.5 Comparison with Existing Solutions

The Broadcast Remover is an implementation which particularly tries to prevent the thread of the broadcast messaging as explained earlier. Additionally other security measures were addressed, yet other than that is unadressed. Therefore, as presented in the Section 2.4, there are several SDN implementations which intent to increase the secureness of the network. Briefly, FRESCO (which at the time of writing this thesis evolved to SE-Floodlight, see Section 2.4.1) is a framework to create security network applications to control an OpenFlow network, and FLOVER (see Section 2.4.2) is a model checking application to verify that SDN applications run by the controller do not create contradicting flow rules or violate predefined security constrains. Thus, Broadcast Remover is not as generic as FRESCO/SE-Floodlight or FLOVER. In addition, Broadcast Remover is still at the theoretical level whereas FRESCO/SE-Floodlight and FLOVER are at the moment functioning implementations. Hence, a direct comparison of abilities can not be done, yet inspection of whether a similar implementation can be build on top of FRESCO or whether FLOVER could be used along the Broadcast Remover, is interesting.

As mentioned in Section 2.4.1, security applications in FRESCO are created by utilizing either the provided FRESCO modules or by creating own modules with FRESCO's scripting language. By combining these modules, functionalities can be created to process OpenFlow events. Hence, by implementing functionalities from the Broadcast Remover in the FRESCO manner, and by adding a database for the host managing purposes, it is possible to create an implementation similarly to the Broadcast Remover. In the end, all OpenFlow implementations consist of construction of certain flow rules upon a specific networking event. Naturally, this requires a proper study and testing. SE-Floodlight (see Section 2.4.1) is the continuation of FRESCO and does not primarily differ from the frameworks' aspect. Hence, building the Broadcast Remover upon SE-Floodlight follows similar pattern as with FRESCO.

FLOVER is an application operating along the controller. When a new flow rule is issued, FLOVER verifies that in the network there are no flow rules contradicting each other (e.g. one rule to allow and another to deny for the same traffic). Hence, FLOVER could run along with Broadcast Remover to verify the flow rule production. Whereas, FRESCO/SE-Floodlight includes only flow rule production prioritisation, FLOVER could be run even along FRESCO or SE-Floodlight to verify the flow rule production of SDN applications. Again, this needs more studying and testing, for proper research.

4.6 Evaluation and Summary

To address the issue of broadcast thread in Ethernet legacy LAN's, Broadcast Remover is designed. The presented implementation was planned carefully, using ex-

istent SDN properties and realistic operation models. As earlier stated, the implementation functions only on IPv4 excluding IPv6 and requires only OpenFlow supported switches. The implementation is still at the theoretical level and the next step is to produce a real implementation by programming the application in to a SDN controller. Testing in a simulation environment or in a physical network is important for benchmarking and analyzing of overall performance. Also it serves the detection of unnoticed short comings.

The Broadcast Remover addresses four types of Ethernet threats in addition to the broadcasting:

- **Man in the Middle attack**, by alternating broadcast behavior, ARP and DHCP poisoning are not more possible. Each host is directly connected to the DHCP server and address resolution is performed at the controller instead in the network. Also when connecting switches to the controller, only authenticated switches are used and if other switches are found the port to it is blocked.
- **Host spoofing**, if web based authentication provides additional protection to the MAC based authentication. If only MAC based is used the potential of spoofing is more likely, yet with web based authentication spoofing a host becomes difficult. Furthermore, spoofing becomes more possible if in the network statical IP addresses are used, the web based authentication is the right solution to provide additional protection, yet general carefulness should be applied when statical IP addresses are used.
- **Denial of Service**, the system can be loaded in multiple occasions with traffic or computation inducing events which will over exhaust the system. By following the flow rule counters, these attempts can be hindered.
- **Eavesdropping**, by preventing Man in the Middle attacks and hindering spoofing attempts, eavesdropping is also prevented. Additionally, by managing the port connections and restricting traffic flow from unauthenticated hosts, eavesdropping becomes almost impossible (possible if spoofing successful).

From the host point of view, the Broadcast Remover provides a secure way to gain an IP address, request an address resolution, reliability of traffic integrity. Whereas from the network point of view, it provides the assurance of trusted switches and authenticate hosts.

The design shows that the usage of web based authentication is highly recommended. It does not only reassure the identity of a host, but it provides protection from spoofing attempts. For example in broadcasting, when a host is on the allowed list, the broadcasts are allowed only from the specific port where the host is connected and only with its identity (MAC and IP address). Here, the spoofing possibility has been

reduced to a single point, which means that if one were to misuse this privilege, it requires only to spoof the privileged host and to reconnect if MAC based authentication is used.

Likewise, the database (managed by the Host manager module), is fragile towards the delays of the host disconnections. There is potential, that at the time of data reading the database is inaccurate. Hence, when the implementation is applied in a real environment, extensive tests should be performed for database related events.

To improve the Broadcast Remover ability to withstand the fail cases, the most important element of the network has to be secured. This is the controller, the *linchpin* of the network. It presents a point of single failure and if the controller is compromised, all the network is. Hence, the securing of the controller physically and with proper authentication is at most importance. For redundancy, a second controller can be applied for backup purposes if the initial controller fails. The second controller copies the state of the initial one and is ready to take immediate control.

The Broadcast Remover is designed to serve a network where the traffic integrity and information confidentiality are required. Users need authentication and devices are identified.

5 Discussion and Conclusions

Currently, it seems that improving security aspects in SDN has been favored by designing security services in the applications and the controller. Also OpenFlow specification has shown that improving security by its standardizing influence is favorable and desirable [63]. There has been speculation on the relationship of the controller with the applications, as to what extend of privilege sets a controller must apply for the applications in order to prevent security exploits [64]. The security development of the network hardware is decoupled from SDN. As SDN focuses only on the software level of the hardware, the security aspects of the network devices stays on the hands of the hardware suppliers.

SDN shows that it has the potential to improve current networks. SDN prevails mainly on the link and transport layer, and similarly its security utilities. Yet, it would be interesting to see how a high layer application could use SDN and add a security entity to it. This area needs clearly pioneering and innovation.

Nevertheless, from the security point of view, SDN has five security focus points and four attack targets, which are presented and introduced in the following subsections. The points are suggested as the improving factors which need development and the defined targets serve as guidelines when developing SDN applications.

5.1 Five Security Focus Points in Software-Defined Networking

As explained in Section 2.1, SDN architecture consists of three layers: **Application**, **Controller**, and **Infrastructure**. Between these layers are two interface layers: **Northbound API** (between Application and Controller) and **Southbound API** (between Controller and Infrastructure). Hence, for improving security in SDN, these are the five focus points to have the attention on. In the Table 3, possible security enforcing factors are introduced to each of the focus points.

In principle, the applications have the control of the network. Through the controller, they could issue any flow rules to manage the traffic of the network. Furthermore, SDN seems to follow the footsteps of computer science where the controller represents the operating system, which run applications on top of it. Hence, it is very likely that the applications are produced by a third party, making the application reliability critical. It would not be a surprise if in the future a popular controller platform got a SDN application store, where the users could buy and install a new feature in the network. This would probably result the emerging of the SDN anti-virus or anti-spyware applications. Existing security implementations such as SE-Floodlight (see Section 2.4.1) try to provide already these types of services. It seems that the verification of the application malevolence deserves proper attention and further study.

The most crucial weakness of SDN is the controller. As the central entity of the network, it represents a single point of failure, compromising the high availability and reliability of the network. The importance of controller's security, both from hardware and software point of view, deserves critical attention. Mechanism and principles should be developed for the case of controller malicious exposé or failure.

5.2 Four Attack Targets in Software-Defined Networking

When designing security services, the basic way to tackle the possible threats is to think from an attackers' perspective [65]. In SDN, four attack targets can be defined:

1 A **Host** or the **End-User**. As in generally, in the networking security, the end-users and host are probably the most desired targets. This is mainly because they have the information, service or function to steal, tamper or sabotage. If an attacker gets access to the target, the attacker will try to find any vulnerabilities to perform the attack.

2 The **Network**. The security enforcing factors presented in each of the five security focus points (see Table 3), can contain vulnerabilities and bugs when implemented. This target presents the ability to alter the network configurations and settings, and for example prevent end users from a service or alter it to a fake one. An attacker would try to expose any vulnerability to take control of the network or some part of it.

3 The **Traffic**. As hosts communicate, crucial information can travel through the network which might attract an attacker. This target presents the ability to eavesdrop traffic and steal its content. By getting access to the network, an attacker could eavesdrop traffic by intervening or mirroring.

4 The **SDN principle**. As previous targets are quite generic as they also apply to a legacy network, targets which are SDN specific are in the SDN principle. The controller, which is the central entity of the network; OpenFlow, which is the standardized communication method between the switches and the controller; SDN applications, which is have the ability to manage and monitor the network, these all can offer the means to access other targets or sabotaging these elements.

Generally, in information technology, software vulnerabilities are discovered either during development or through an expose from an external entity (either an attacker or a user). Hence, a substantial amount of resources are used for testing while developing software [66]. These targets provide the means for designing test cases for security testing, when developing SDN software and especially when designing SDN controller. These targets are not definitive, as each target can include several

sub targets. Therefore, a proper study or a survey of an optimal testing design for SDN paradigm is needed, especially designed for security aspects.

5.3 Conclusion

SDN can bring many advantages to improve Ethernet's security. Due to the entity of a central element in the network, the controller, SDN gives an extended control over the network. Hosts connected to network can be directly recognized and managed. Network traffic can be categorized by any parameter and routed precisely through any route, and new features can be implemented on the fly. Ultimately, the network can be shaped according to the demand. One could call this naively, the ultimate network power.

Nevertheless, as the saying goes *"With great power, comes great responsibility..."*, reflects perfectly the case of SDN. As seen from the Sections 3 and 4, SDN's ability to control the network, removes the need of many protocols along with their vulnerabilities, such as STP and VLAN. Other protocols (specifically link layer related) can still be used and their weaknesses overcome. For example, ARP broadcast messages, which has spoofing vulnerabilities, can be overcome by redirecting a traffic flow.

Then again, new vulnerabilities arise, which need meticulousness attention. For example, the threat of the compromised controller, is a serious problem demanding a complex solution. Compromising a controller can be achieved as easily as by slipping a malicious SDN application into the controller. This is a known drawback when the network transforms from distributed to a centralized network.

The existing SDN security implementation presented in Section 3, proves that the researching community has recognized the demand for secure SDN solutions and hence provided with a framework or a system checking solutions. There are also other implementations which were not introduced in this thesis, yet they have a similar purpose.

Summing up, SDN can offer more secure network if itself is properly secured. In the end, SDN has the potential to revolutionize the network industry.

Table 3: Some possible security enforcing factors on security focus points are listed below.

Application layer	– Verify the origin of the application – Let application participate on authentication towards the controller – Verify the reliability of the controller – State the required information needed to operate and give privileges to access only them – Require authorization of accessing to modify or monitor the application – Inter application communication should be done only through the controller to support the reliability of the application
Northbound API	– Limit access to API only for authorized applications – The inter application communication has to go through the controller
Controller layer	– Provide network information only to authenticated applications – Create different network information levels – Provide privileges to applications according to fore mentioned levels – Use secure communication with OpenFlow (TLS) when communicating with the switches – Check the flow rule requests from the applications, and prevent conflicts or security violations – Provide backup controller in case of primary fails – Provide switches with safety run mode to perform when connection to the controller fails – Require authorization for accessing to modify or monitor the controller
Southbound API	– Use secure communication of OpenFlow (encryption and authentication)
Infrastructure layer	– Authentication to and from the controller – Require safety mode (instruction how to perform when controller connection fails) – Require authorization for accessing to modify or monitor the switch – Network devices (switch and controller) placement in a safe environment with limited access – Secure network cabling with limited access

References

[1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, (Berkeley, CA, USA), pp. 1–6, USENIX Association, 2010.

[2] R. Chua, "Juniper's New SDN Strategy and Contrail's Starring Role," Posted on SDN central: Jan. 15, 2013, http://www.sdncentral.com/sdn-blog/juniper-new-sdn-strategy-contrail-role/2013/01/ Retrieved: May 28, 2014.

[3] H. Zimmermann, "OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection," *Communications, IEEE Transactions on*, vol. 28, pp. 425–432, Apr 1980.

[4] ONF, "Open Networking Foundation." https://www.opennetworking.org/ Retrieved May 28, 2014.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[6] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown, and S. Shenker, "SANE: A protection architecture for enterprise networks," in *USENIX Security Symposium*, 2006.

[7] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, (New York, NY, USA), pp. 1–12, ACM, 2007.

[8] B. Pfaff *et al.*, "Openflow switch specification v1.3.0," 2012.

[9] The Institute of Electrical and Electronics Engineers (IEEE), "IEEE 802.3 working group." http://www.ieee802.org/3/ Retrieved: May 28, 2014.

[10] R. Metcalfe and D. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, vol. 19, pp. 395–404, July 1976.

[11] J. Abbate, "The triumph of ethernet: Technological communities and the battle for the lan standard (review)," *Technology and Culture*, vol. 44, no. 3, pp. 635–637, 2003.

[12] "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. Amendment 2: Registration of Object Identifiers," *IEEE Std 802-2001 (Revision of IEEE Std 802-1990)*, pp. 19–27, 2002.

[13] "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment 4: Provider Bridges," *IEEE Std 802.1ad-2005 (Amendment to IEEE Std 802.1Q-2005)*, pp. 1–74, May 2006.

[14] "IEEE Standard for Local and Metropolitan Area Networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks," *IEEE Std 802.1Q-2011 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–1365, Aug 2011.

[15] "IEEE Standard for Local and Metropolitan Area Networks–Media Access Control (MAC) Bridges," *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, pp. 1–277, June 2004.

[16] R. Droms, "Dynamic Host Configuration Protocol." Internet Engineering Task Force, Request for Comments (RFC) 2131 (Draft Standard), Mar. 1997.

[17] D. Plummer, "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware." Internet Engineering Task Force, Request for Comments (RFC) 826 (INTERNET STANDARD), Nov. 1982.

[18] P. Mockapetris, "Domain names - concepts and facilities." Internet Engineering Task Force, Request for Comments (RFC) 1034 (INTERNET STANDARD), Nov. 1987.

[19] P. Mockapetris, "Domain names - implementation and specification." Internet Engineering Task Force, Request for Comments (RFC) 1035 (INTERNET STANDARD), Nov. 1987.

[20] K. Ingham and S. Forrest, "A history and survey of network firewalls," *University of New Mexico, Tech. Rep*, 2002.

[21] I. Dubrawsky, "Firewall evolution - Deep Packet Inspection," *Security Focus*, vol. 29, 2003.

[22] X. Zhang, C. Li, and W. Zheng, "Intrusion prevention system design," in *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*, pp. 386–390, 2004.

[23] P. Congdon, B. Aboba, A. Smith, G. Zorn, and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines." Internet Engineering Task Force, Request for Comments (RFC) 3580 (Informational), Sept. 2003.

[24] "IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control," *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)*, pp. C1–205, Feb 2010.

[25] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks," in *To appear in the ISOC Network and Distributed System Security Symposium*, 2013.

[26] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, pp. 263–297, Aug. 2000.

[27] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 121–126, ACM, 2012.

[28] S. Shin, P. Porras, V. Yegneswaran, and G. Gu, "A framework for integrating security services into software-defined networks,"

[29] "Security enhanced floodlight." Presentation at Open Network Summit, April 2013. Available at: http://sdrv.ms/XXyl4X Retrieved May 28, 2014.

[30] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model Checking Invariant Security Properties in OpenFlow," June, 2013.

[31] B. Dutertre and L. De Moura, "The yices SMT solver," *Tool paper at http://yices.csl.sri.com/tool-paper.pdf*, vol. 2, p. 2, 2006.

[32] R. Chua, "SDN Security An Oxymoron? New Interview with Phil Porras of SRI International." Posted on SDN central: Feb. 26, 2013 http://www.sdncentral.com/technology/sdn-security-oxymoron-phil-porras-sri/2013/02/ Retrieved: May 28, 2014.

[33] C. Matsumoto, "Startup Tackles OpenFlow Security." Posted on LightReading: Dec. 4, 2012, http://www.lightreading.com/document.asp?doc_id=695093 Retrieved: May 28, 2014.

[34] J. Dix, "Network world: The promise of software defined networking." Posted on Digital Spotlight, Network World: May 13, 2013, http://www.networkworld.com/news/2013/070113-sdn-271298.html Retrieved: May 28, 2014.

[35] M. Kassner, "Software-Defined Networking: How it affects network security." Posted on TechRepublic: Jul. 15, 2013, http://www.techrepublic.com/blog/it-security/software-defined-networking-how-it-affects-network-security/ Retrieved: May 28, 2014.

[36] N. MacDonald, "The impact of software-defined data centers." Published at Gartner: Oct. 16, 2012.

[37] J. Sommer, S. Gunreben, F. Feller, M. Kohn, A. Mifdaoui, D. Sass, and J. Scharf, "Ethernet - a survey on its fields of application," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 2, pp. 263–284, 2010.

[38] T. Kiravuo, M. Sarela, and J. Manner, "A survey of ethernet lan security," *IEEE Communications Surveys and Tutorials*, 2013. Article in Press.

[39] "Global Security Report 2011." A white paper published by Trustwave, 2011. http://www.secretservice.gov/Trustwave_WP_Global_Security_Report_2011.pdf Retrieved: May 28,2014.

[40] B. Aboba, D. Simon, and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework." RFC 5247 (Proposed Standard), Aug. 2008.

[41] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246 (Proposed Standard), Aug. 2008.

[42] S. HomChaudhuri and M. Foschiano, "Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment." RFC 5517 (Informational), Feb. 2010.

[43] D. Siegel, "On the new threats of social engineering exploiting social networks," bachelor's thesis, Technische Universität München, Aug 2009.

[44] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[45] O. Arkin and J. Anderson, "EtherLeak: Ethernet Frame Padding Information Leakage." http://www.rootsecure.net/content/downloads/pdf/atstake_etherleak_report.pdf, 2003. Retrieved Feb 3, 2011.

[46] C. Abad and R. Bonilla, "An analysis on the schemes for detecting and preventing ARP cache poisoning attacks," in *Distributed Computing Systems Workshops, 2007. ICDCSW '07. 27th International Conference on*, p. 60, IEEE, Jun 2007.

[47] D. Bruschi, A. Ornaghi, and E. Rosti, "S-ARP: a secure address resolution protocol," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pp. 66–74, IEEE, Dec 2003.

[48] I. Dubrawsky, "Safe layer 2 security in-depth." A white paper published at Cisco Inc, 2004. http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/sfblu_wp.htm Retrieved: May 28, 2014.

[49] O. Zheng, J. Poon, and K. Beznosov, "Application-based TCP hijacking," in *Proceedings of the Second European Workshop on System Security*, EUROSEC '09, (New York, NY, USA), pp. 9–15, ACM Press, 2009.

[50] J.-W. Lee, S.-H. Park, K.-H. Gum, and T.-M. Chung, "Design of secure arp on macsec(802.1AE)," in *Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference on*, pp. 1 –4, dec. 2010.

[51] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol." RFC 4252 (Proposed Standard), Jan. 2006.

[52] M. Marro, "Attacks at the data link layer," Master's thesis, University of California Davis, 2003.

[53] "IEEE Standard for Local and Metropolitan Area Networks– Station and Media Access Control Connectivity Discovery," *IEEE Std 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005)*, pp. 1–204, Sept 2009.

[54] S. Cheshire, B. Aboba, and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses." RFC 3927 (Proposed Standard), May 2005.

[55] P. Phaal, "Detecting NAT devices using sFlow," 2003. Published at sFlow http://www.sflow.org/detectNAT Retrieved: May 28, 2014.

[56] S. Cheshire, "IPv4 Address Conflict Detection." RFC 5227 (Proposed Standard), July 2008.

[57] R. Finlayson, T. Mann, J. Mogul, and M. Theimer, "A Reverse Address Resolution Protocol." RFC 903 (INTERNET STANDARD), June 1984.

[58] T. Bradley, C. Brown, and A. Malis, "Inverse Address Resolution Protocol." RFC 2390 (Draft Standard), Sept. 1998.

[59] J. Postel, "User Datagram Protocol." RFC 768 (INTERNET STANDARD), Aug. 1980.

[60] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[61] R. Bellman, "On a routing problem," tech. rep., DTIC Document, 1956.

[62] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-n-serve: Load-balancing web traffic using openflow," *ACM SIGCOMM Demo*, 2009.

[63] S. H. M. Wasserman, "Security Analysis of the Open Networking Foundation (ONF) OpenFlow Switch Specification." Internet-Draft for Internet Engineering Task Force: Apr. 17, 2013.

[64] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for openflow applications," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 171–172, ACM, 2013.

[65] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. New York, NY, USA: John Wiley & Sons, Inc., 1st ed., 2001.

[66] L. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *Software Engineering, IEEE Transactions on*, vol. SE-4, pp. 345–361, July 1978.