Hellenic Mediterranean University

Department of Electrical and Computer Engineering

MSc in Informatics Engineering

Master Thesis

“Cyber Security Analysis on a Production Environment featuring Software Defined Networks”

Panagiotis Aspradakis

mp154

Supervisor: Prof. Spyridon Panagiotakis

Περίληψη

Οι συνεχώς μεταβαλλόμενες απειλές στον κυβερνοχώρο αποτελούν καθοριστικό χαρακτηριστικό του σημερινού παγκόσμιου τοπίου. Η ασφάλεια στον κυβερνοχώρο είναι απαραίτητη για οργανισμούς, επιχειρήσεις και πολλούς άλλους ενδιαφερόμενους και ένα από τα σημαντικότερα ζητήματα για τη βιομηχανία 4.0. Με μια ποικιλία ατόμων που συμμετέχουν σε οργανισμούς, δίκτυα επιχειρήσεων, βιομηχανίες, περιήγηση στον Ιστό, σχετικά με τις επιχειρηματικές τους εργασίες ή άλλες διαδικτυακές δραστηριότητες, πρέπει να ληφθούν μέτρα για τη διασφάλιση της εποπτείας κάθε χρήστη. Το πρόβλημα είναι ότι με το συμβατικό δίκτυο που χρησιμοποιείται αυτήν τη στιγμή, ένας διαχειριστής δικτύου ή ένας επαγγελματίας πληροφορικής δεν μπορεί να έχει ακριβή εικόνα σε πραγματικό χρόνο για το πόσες συσκευές βρίσκονται στον οργανισμό, την επιχείρηση ή τη βιομηχανία και δεν έχει την ευελιξία να αλλάξει το δίκτυο τοπολογία εν κινήσει, για την αντιμετώπιση των απαιτήσεων χρήσης σε πραγματικό χρόνο ή συμβάντων ασφαλείας.

Το Δίκτυο Καθορισμένο από Λογισμικό (SDN - Software Defined Network) είναι πολύ σημαντικό για την οργάνωση του δικτύου στη Βιομηχανία 4 επειδή επιτρέπει τον διαχωρισμό του ελέγχου από τις ροές δεδομένων. Ως εκ τούτου, είναι σύμφωνο με τις άλλες αρχές του λογισμικού και της νεφοποίησης της παραγωγής. Ο ελεγκτής ενορχηστρώνει τις περιόδους σύνδεσης δικτύου μεταξύ υπηρεσιών σε κοντέινερ και τελικών χρηστών ή τελικών συσκευών. Ένα τείχος προστασίας στην είσοδο ενός εταιρικού δικτύου παρακολουθεί την κυκλοφορία δεδομένων και επιτρέπει την επιβολή κανόνων σε επίπεδο ροής. Το Σύστημα Ανίχνευσης Εισβολής / Σύστημα Πρόληψης Εισβολής (IDS/IPS - Intrusion Detection System/Intrusion Prevention System) επιτρέπει την εφαρμογή κανόνων σχετικά με την προστασία της υποδομής από κακόβουλους εισβολείς και ενορχηστρώνει

την προστασία της ασφάλειας στον κυβερνοχώρο του συνολικού οικοσυστήματος από τους εισβολείς.

Με κίνητρο τα παραπάνω, αυτή η μεταπτυχιακή διατριβή θα αξιοποιήσει την ευελιξία των Δικτύων Καθορισμένων από Λογισμικό, της τεχνολογίας Βαθιάς Επιθεώρησης Πακέτων (DPI - Deep Packet Inspection) και του Δικτύου Εικονικής Λειτουργίας (NFV - Network Function Virtualization), αξιοποιώντας εικονικές μηχανές για να μιμηθούν ένα πραγματικό περιβάλλον παραγωγής που αποτελείται από διάφορα συστήματα παραγωγής και λειτουργίες ασφαλείας. Εστιάζοντας στις πτυχές της ασφάλειας στον κυβερνοχώρο, θα εκτιμηθεί τόσο η εκτεταμένη επιφάνεια επίθεσης που θα εισαχθούν αυτές οι νέες τεχνολογίες, όσο και οι ενισχυμένοι μηχανισμοί ασφαλείας που επιτρέπουν (π.χ. αντιδραστικές ή κινούμενες τεχνικές άμυνας).

Για να μάθουμε περισσότερα σχετικά με το πώς λειτουργούν τα Δίκτυα Καθορισμένα από Λογισμικό και οι σχετικές τεχνολογίες και πώς κάποιος με τη διοικητική θέση πάνω του μπορεί να το κάνει πιο ασφαλές, καταλήξαμε στην ιδέα να δημιουργήσουμε ένα εικονικό Δίκτυο Καθορισμένο από Λογισμικό και να εφαρμόσουμε λύσεις ασφαλείας σε αυτό, βλέποντας πώς θα συμπεριφερόταν σημειώνοντας ταυτόχρονα στατιστικά στοιχεία κίνησης και ροής πακέτων δεδομένου ότι η πλατφόρμα εικονικοποίησης που χρησιμοποιήσαμε δεν συνοδεύεται από συγκεκριμένα εργαλεία ή εφαρμογές ασφαλείας Δικτύων Καθορισμένων από Λογισμικό, ενσωματώσαμε μια παραδοσιακή λειτουργία ασφαλείας στην υλοποίηση της υποδομής μου. Τα παλαιά δίκτυα είναι επίσης ξεπερασμένα, παρέχοντας ένα ευρύ φάσμα πόρων ασφαλείας παλαιού τύπου που μπορεί να χρησιμοποιήσει κάποιος. Το Δίκτυο Καθορισμένο από Λογισμικό λειτουργεί πολύ παρόμοια με ένα τυπικό δίκτυο και όταν πρόκειται για πακέτα κυκλοφορίας και πρωτόκολλα, είναι δυνατή η χρήση αυτών των πόρων σε υποδομή Δικτύου Καθορισμένο από Λογισμικό. Στη συνέχεια, ολόκληρο το έργο άρχισε να ασχολείται με αυτό, ελέγχοντας την αποτελεσματική ενσωμάτωση υπαρχόντων εργαλείων ασφαλείας δικτύου στις λειτουργίες μας για να κάνουμε το SDN ασφαλέστερο και πιο ανθεκτικό στις επιθέσεις.

Abstract

The ever-changing cyber threats are a defining feature of today's global landscape. Cyber Security is essential for organizations, businesses and many other stakeholders and one of the major issues for Industry 4.0. With a variety of people participating in organizations, businesses networks, industries, browsing the web, going about their business tasks or other online activities, steps must be taken to ensure the supervision of each user. The problem is that with the conventional network currently in use, a network administrator or an IT practitioner cannot have an accurate real-time view of how many devices are in the organization, business or industry, and does not have the flexibility to change the network topology on the fly, to address the real-time usage requirements or security incidents.

SDN is very important for network organisation in Industry 4 because it enables separation of control from data flows. Hence it is inline with the other principles of softwarization and cloudification of the production. The controller orchestrates the network sessions between containerised services and end users or end devices. A firewall at the entrance of an enterprise network monitors the data traffic and enables rules enforcement at flow level. IDS/IPS enables the application of rules concerning protection of the infrastructure from malicious intruders and orchestrates the cybersecurity protection of the overall ecosystem from attackers.

Motivated by the above, this Master's Thesis will leverage the flexibility of Software Defined Networks (SDN), Deep Packet Inspection (DPI) technology and Network Function Virtualization (NFV), exploiting virtual machines to emulate a real production environment consisting of various production systems and security functions. Focusing on cybersecurity aspects, both the expanded attack surface that these new technologies introduced will be

assessed, as well as the enhanced security mechanisms that they enable (e.g. reactive or moving defense techniques).

To learn more about how SDN and related technologies work, and how someone with the administrative position over it can make it safer, we came up with the idea of setting up a virtual SDN network and implementing security solutions on it, seeing how it would behave while noting traffic and packet flow statistics. Since the virtualization platform that we used does not come with any SDN specific security tools or applications, we incorporated a traditional security feature into my implementation of the infrastructure. Legacy networks are also obsolete, providing a wide array of legacy security resources that one might use. SDN operates very similarly to a standard network, and when it comes to packet traffic and protocols, use of these resources on an SDN infrastructure is and is possible. The entire project then started to address around this, checking the efficient incorporation of existing network security tools into our functions to make SDN safer and more attack-proof.

Table of contents

1. Introduction

The rapid advances in Information and Communication Technologies (ICT) [74] are drastically affecting most industries. However, this rapid development requires a proper understanding and definition, both by researchers and professionals, in order to achieve a broad spread of technological progress. From 2008 onwards, as internationally recognized, we are going through the 4th industrial revolution, ie the rapid evolution of digital technology with more advanced quality features. In other words, it is digital technology and its advanced forms, such as Internet of Things (IoT) [71][72] and Internet of Services (IoS), artificial intelligence, nanotechnology, soon the quantum computer and the combination of all the above forms and types.

The 4th Industrial Revolution is primarily a vision of change and a hopefully integrated approach, based on the integration and holistic management of the processes of modern business. For the successful transition to Industry 4.0, the digital transformation is considered necessary, which is a continuous evolutionary process, which creates opportunities, but at the same time requires modern infrastructure and training in specialized skills.

Industry 4.0 is not just about automation, but calls for a new model of intelligent automation using decentralized control systems, advanced connectivity and interoperability based largely on the continuous flow of information through the Internet of Things (IoT). The reorganization of traditional automation systems with intelligent systems that self-improve and self-regulate, through Machine Learning and Data Analytics algorithms, effectively contributes to the optimal adaptation of business processes to changing consumer needs. Data is in the spotlight. The exchange of information and big data in real time, which are subject to extensive and targeted statistical analysis to be useful business guidelines, contributes significantly to critical decision making. Also Industry 4.0 affects all enterprise networks, all productive sectors, forcing them to follow new trends. Some examples of this trend are smart factories, smart cities, smart grids, smart agriculture, smart, farming, etc.

Today, the trend towards a digital revolution in industrial production is known as Industry 4.0 [73]. Industry 4.0 as a concept is widely used in Europe and mainly in the German industrial sector. One of the first times the term Industry 4.0 was mentioned was in a "memo" of the German government, issued in 2013. German Chancellor Angela Merkel herself in another case (January 2015 at the World Economic Forum in Davos) referred with particular emphasis to Industry 4.0 as "the way we will be able to achieve the 'fusion' of the online world and the world of industrial production". In the USA and in other English-speaking countries some commentators use alternative terms Internet of Things (IoT) [75], Internet of Everything (IoE), or Industrial Internet. The common denominator of all these concepts is the recognition that traditional methods of industrial production are under the influence of a digital transformation. The real innovation comes not so much from the existence of a new technology as from the combination of available technologies but in a different way, thus providing a plethora of new possibilities. So under Industry 4.0, components, products, machines, spare parts and other physical entities acquire their own network identity. They can negotiate with each other or be interconnected and their interaction can be simulated. The various systems can be virtually completed, tested and optimized.

We are now experiencing the second wave of the fourth Industrial Revolution. The first wave of Industry 4.0 (2009 - 2016), focused on the concepts of Digitalization, Internet of Things, Robotics, Advance Analytics and Cloud Computing. The second wave of Industry 4.0 (2016 - 2025) focuses on the following topics: Artificial Intelligence, Blockchain, Smart Automation, 6G Communication (1 Terabyte upload / download) and new energy sources. The third wave

(2025 - …) will be related to Quantum Computing, Cybersecurity, Neuro-technology, Nano-technology and Bio-informatics. The Fourth Industrial Revolution sets the new limits of prosperity as well as the future of labor, while redefining the policies that form the basis for the competitiveness of trade in the future.

The Internet of Things (IoT), now known to all of us, is the communication network of a variety of devices, home appliances, cars as well as any object that incorporates electronic means, such as software, sensors and network connectivity, to allow connection and data exchange. Simply put, the philosophy of the IoT is to connect all electronic devices to each other (local area network) or with the ability to connect to the Internet (World Wide Web).

The term "Things" is not strictly associated with certain products. It refers to a wide variety of devices completely different from each other, such as cars with integrated sensors, cameras, air conditioners, lights, security systems, smartwatches and even cars whose complex sensors detect objects in their path. They are some of the many technology products. A key feature of all is the connection between them with the ultimate goal of the user being able to control them from a computer or mobile. Internet of Things is one of the top three technological developments of the next decade and is the next big step in technology.

Next-generation IoT systems require scalability, and integration of heterogeneous devices and protocols. To meet these challenges, an advanced architecture was proposed based on microservices technology, consisting of services, i.e. specific functions derived from the IoT level, and data. The Edge server supports the processing of data from various sensors. This step saves bandwidth for communicating with the cloud, and helps to decongest the kernel. The vast proliferation of mobile devices and content, the virtualization of servers, and the advent of cloud services are among the trends that are driving the networking industry to rethink traditional network architectures.

The design of traditional networks makes extremely difficult to meet the current requirements of users, companies, industries and suppliers. Mobile devices, virtualization network equipment [9] or cloud services are becoming increasingly important as the various factors force the telecommunications manufacturer to reconsider traditional network structures.

In order to provide an answer to this situation, the Software Defined Networks (SDN) [7] arises. The SDN networks have an architecture dynamic, manageable and adaptable network, which makes them suitable for the dynamic nature of applications nowadays. The main difference between these networks and traditional is the separation between control functions and forwarding, allowing programmed directly on the control plane and data plane abstracting. This feature provides a number of advantages to SDN networks such independence from manufacturers, vision and global and centralized control networks, programming and service automation. In short, makes it possible to configure and freely manage all network resources, simplifying deployment and reducing costs.

The variety of possibilities and promises offered by SDN networks have made major companies in the field of telecommunications, such as Google [11], they have opted for this technology, migrating traditional SDN networks in order to improve performance and reduce costs networks. However, with the SDNs a relatively new concept, there is still a long way to go in important areas such as network security [3].

All this convergence of the internet world with the world of production has made the latter vulnerable to cybersecurity issues, with the result that the issue of shielding enterprise networks against cyberattacks is very critical. Particularly useful at this level is the issue of

DPI as it helps with intrusion detection issues. So in this context in this thesis we examine the weaknesses of modern enterprise networks and propose a protection methodology that uses DPI for application in enterprise networks that use SDN.

Many conventional networks are hierarchical, based on rows of Ethernet switches arranged in a tree structure. This topology made sense when the client-server structure was dominant, but such a static architecture no longer meets the dynamic computing and storage needs of today's data centers. SDNs come to meet the current challenges, meeting the current and future needs of modern networking.

SDNs play an important role in Industry 4.0, allowing various users around the world to connect, scale up new services and transform industries. It will also allow solutions such as the Internet of Things (IoT) and Artificial Intelligence (AI) to become global by removing barriers to connectivity. In essence, SDNs collect and remove control and automation of workflows through network virtualization (NFV), making it ideal for optimizing computing, distributing computing power in remote locations, moving data center functions, and so on support for Internet of Things (IoT) environments.

Many companies even today rely on the management and production of systems that are unconnected or closed. However, with the increase in connectivity and the use of standard communication protocols, which are created in the context of Industry 4.0, there is a strong need to protect critical industrial systems from cyber security threats. Once Cyber-Phisical Systems are developed in organizations, malware can affect and spread from one machine to another through communication systems. Viruses that spread through this malicious software may be intended to modify the production process or destroy data through the system, leading to product quality errors or a complete shutdown. CPS security issues in the form of cyber-attacks and data theft are critical and need to be controlled to improve system reliability and acceptance. This requires secure, reliable communication and advanced authentication and access management systems for both machines and users. Industrial data is very sensitive as it incorporates many aspects of industrial operation including information on product business strategies and the company itself.

Connected devices and the Internet of Things are increasingly used for large-scale attacks. Various DDoS attacks have been reported during 2016, including the DDoS attack up to 620 Gbps, which keeps well-known security journalist Bryan Krebs's website off operation since the end of September 2016. The fears are high that soon such attacks will become even more intense and will become the norm. And these are not just DDoS attacks. Ransomware is also circulating on the Internet of Things, and security experts warn of the pervasive impact of exploited vulnerabilities on the IoT.

Many of the applications required by businesses, such as Skype, WebEx, etc., use proprietary signatures and web ports for communication, but this carries risks that may result in data loss as there is virtually no control over applications and protection against misuse, and also cannot properly configure traffic (QoS). Through DPI technologies there can be uninterrupted bandwidth availability even during saturation periods, giving users fully controlled access without compromising on performance.

To learn more about how SDN and related technologies work, and how someone with the administrative position over it can make it safer, we came up with the idea of setting up a virtual SDN network and implementing security solutions on it, seeing how it would behave while noting traffic and packet flow statistics.

Since the virtualization platform that we used doesn't come with any SDN specific security tools or applications, we incorporated a traditional security feature into my implementation of the infrastructure. Legacy networks are also obsolete, providing a wide array of legacy security resources that one might use. SDN operates very similarly to a standard network, and when it comes to packet traffic and protocols, use of these resources on an SDN infrastructure is and is possible.

The entire project then started to address around this, checking the efficient incorporation of existing network security tools into our functions to make SDN safer and more attack-proof.

Thesis is structured in separate sections as follows:

- In the second chapter, we will describe the related technologies that we will use for our implementation, such as Network Defined Networking (SDN) technology, Deep Packet Inspection (DPI) [8] technology and Network Function Virtualization (NFV) [4] as well as how they coexist with each other.
- In the third chapter we will talk about network security and above all we will focus on what we are most interested in which is nothing but security in SDN where we will refer to the threats [1] that govern them and the defense mechanisms [2] to protect them.
- In the fourth chapter we talk about Industry 4.0 and the relationship with SDN and also we will analyze the tools that we will use to implement the experimental part of this thesis, as well as what their role will be in the whole implementation.
- In the fifth chapter we will carry out the first part of our experimental part of this thesis where we will simulate a virtual environment of enterprise network which will include a lot of the technologies that we have analyze at previous chapters and we will make cyber attacks on the infrastructure [6] that we will have implemented and see how it reacts.

- In the sixth chapter we will also carry out an experimental part of this thesis where now we will simulate the Detect and Prevention system for the cyber attacks on the infrastructure that we have implemented at chapter five.

- Finally, seventh chapter presents the conclusions and directions of the future research that emerged during the preparation of this thesis.

2. Related Technologies

In this chapter we will try to analyze the technology of Software Defined Networks (SDN), Network Function Virtualization (NFV) technology, Deep Packet Inspection (DPI) technology and how these technologies can be interconnected.

2.1 Software Defined Networks (SDN)

2.1.1 Definition

On the web we can find quite a few different definitions of SDN but everyone eventually comes to the same interpretation. The Open Network Foundation (ONF) **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** describes SDN as follows: SDN is an evolving architecture that is scalable, accessible, cost-effective and adaptable, makes it perfect for today's applications with a high bandwidth, dynamic character. This architecture decouples network management and forwarding functions allowing direct programmable network control and abstracting the underlying infrastructure for applications and network services. The OpenFlow protocol is a cornerstone for developing SDN solutions.

From Wikipedia [13] we take this definition: «SDN is a network management methodology which allows energetic, programmatically proficient organize setup to boost network efficiency and surveillance to make it further like clouding in addition to conventional network management. [14] SDN is intended to address the question that conventional network inactive design is decentralized and complicated while modern networks want greater adaptability and simpler troubleshooting. SDN aims the centralization of network intelligence in a single network part differentiating the network packet forwarding process (data plane) from routing (control plane). The control plane composed by a controller or more that are known to be the SDN network brain where the entire information is developed into. Nevertheless, when it comes to security, intelligence centralization has its own drawbacks,[14] scalability and elasticity [14] which is the major issue of SDN.

Since the latter's introduction in 2011, SDN has been commonly related with the OpenFlow protocol (decide the route of the network packets via network switches for remote communication with network plane components). But since 2012 [15] [16] OpenFlow is no longer an exclusive option for many businesses, they have introduced proprietary techniques to it. These include the Open Network Environment of Cisco Systems, and the virtualization platform of Nicira's network. SD-WAN relate innovation comparable to Wide Area Network (WAN). [17]

We may conclude that Software-Defined Networking (SDN) is an architecture aimed at making networks scalable and agile. SDN's goal is to enhance network management, allowing businesses and service providers to respond rapidly to evolving business needs.

A network technician or network manager can configure traffic from a central control console within a Software-Defined Network without needing to handle specific switches on the network. The SDN host controller manages the switches so can provide network services whenever necessary, irrespective of the server's unique connections to the devices. This method is a step away from common network structure where, based on their configured routing tables, individual network devices make traffic decisions 0. In the other hand if we see our traditional network where traffic is moving between various switches & routers in order to reach its final destination.

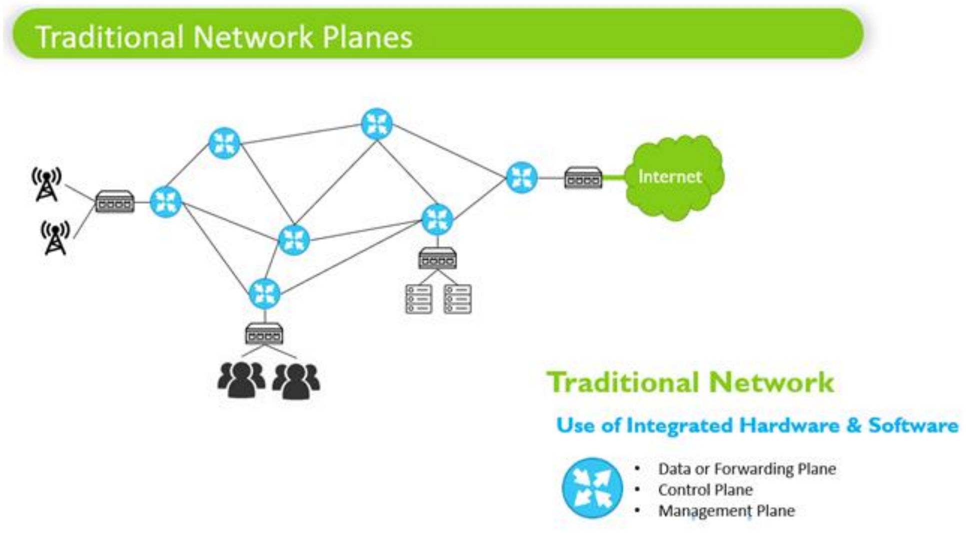

Traditional Network Planes
Internet
Traditional Network
Use of Integrated Hardware & Software
• Data or Forwarding Plane
• Control Plane
• Management Plane

Figure 2.1 – Traditional Network Planes

SDN changes how networking is fundamentally done. Instead of having network intelligence distributed across every device, SDN aims to centralize command and control in a master device (or a few of them for redundancy) and to split networking into three planes as we can see in the next paragraph.

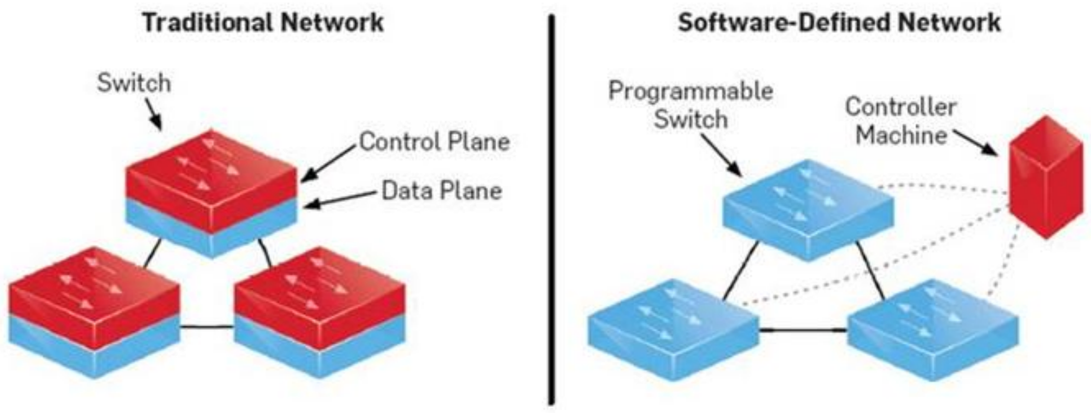


Figure 2.2 – Traditional Network to SDN Network

2.1.2 SDN structure

A typical presentation of SDN structure consists of three layers: operation, control and infrastructure as shown in figure 2.3. The SDN structure consists of decoupling the management and control tasks from the network (the control plane) and the packet transmission task (the data plane).

As shown in Figure 2.3, the controller communicates with the physical devices of the network (the switches) through a standard communication protocol called OpenFlow (the most used with SDN technology), while using APIs to interact with the network application layer.

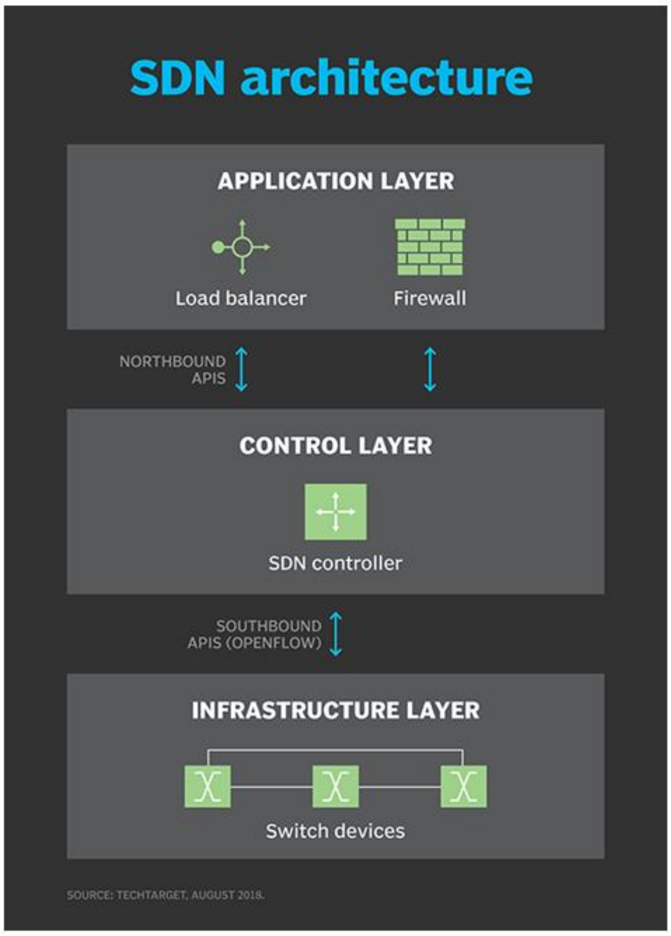


Figure 2.3 – SDN structure

The configuration of the SDN splits the network to three different layers, linked through the Northbound and Southbound APIs.

Heart of this structure is Control layer. It is responsible for the configuration, management and maintenance of the network. As the name implies, this layer controls the data plane by establishing the rules it will have to follow (for example, routing tables and switch tables). Among the protocols that participate in this plane there may be mentioned for example OSPF, STP, ARP, or BGP. This layer uses a logical entity called an SDN controller that is the network “brain”. This controller is responsible for discovering and maintaining the topology, informing the devices of the transmission plane on how to manage the packets in the network, switching the load of paths and communicating with the services and applications offered by the network. Typically, the controller is centralized and is intended to ensure that the network as a whole is operating optimally by communicating with the application plane via APIs and the data plane using a communication protocol (OpenFlow, ForCes [19], IRS [20], NetConf [21], etc.).

The infrastructure layer designates all the equipment (switches) constituting the network. It represents the Data Plane in the SDN architecture. The switches are responsible only for the transmission of packets from the source to the destination. These switches route streams based on information contained in tables (for example FIB or CAM/TCAM). The transmission plane provides the switching and packet filtering functions.

The third layer presents the application plane of the SDN architecture. This layer contains programs and services that perform functions specific to the SDN controller via the API application interfaces such as REST. These applications may create an abstract view of the network by gathering controller information for decision making purposes. These SDN applications could include network monitoring, topology discovery, path reservation, statistics and network analysis, network security, access control, Network Function Virtualization (VNF) etc. Also, application level, not surprisingly, contains traditional network apps or functions used by organizations, which can contain IDS, firewalls or load balancing. When a conventional network uses a functional system, such load balancer or firewall, an SDN substitute the system with an app which utilizes the controller to track data plane actions.

These three layers interact via northbound and southbound APIs, respectively. Via its northbound interface apps can talk to the controller while controllers and switches talk each other through southbound interfaces, like OpenFlow-though there are additional protocols.

The north controller API currently does not have a standard design matching OpenFlow as a main southbound interface. It's attainable that the OpenDaylight controller's northbound API can appear over time as a prototype provide extensive assistance from its vendor.

Although the ONF is constantly changing in terms of terminology, the most common terms for network architecture components are:

- SDN App (SDN App): SDN applications are programs that directly communicate the needs and desired behaviors of your network to the SDN controller through the NBIs (Northbound Interfaces). They consist of an application logic and one or more NBIs.

- SDN Controller (SDN Controller): Logical control entity responsible for converting SDN application requests into data paths, providing an abstract network view to the application layer through statistics and potential acts. From one NBIs or further, the SDN control logic and the CDPI (Control Data Plane Interface) driver, an SDN controller is composed.

- SDN Datapath: A logical aspect which exhibits visibility and audit of its capabilities for forwarding and processing. It consists of a CDPI agent, a collection of forwarding engines and processing functions which include easy forwarding between external interfaces and internal processing of traffic. Single (physical) network dimension may contain data paths.

- SDN interface from the control plane to the data plane: Interface between controller and data path, which provides programmability at the time of forwarding, capacity announcement, statistical reporting and notifications events

- “Northbound” SDN (NBI) interfaces: These are interfaces that provide abstract views of network activity and its specifications between SDN applications and controllers.

- "Drivers" and "agents" interface (Interface Drivers & Agents): Each interface is implemented by a pair of this type, which represents the background (related to the infrastructure) and the top (related to the application).

- Management and administration (Management & Admin): The management plane covers static tasks, better managed outside the application, control and data planes, such as the allocation of resources to customers, the configuration of physical equipment and the agreement between reachability and credentials between physical and logical entities.

The following figure 2.4 shows the graphic scheme of the architecture of the SDN networks and all their components.

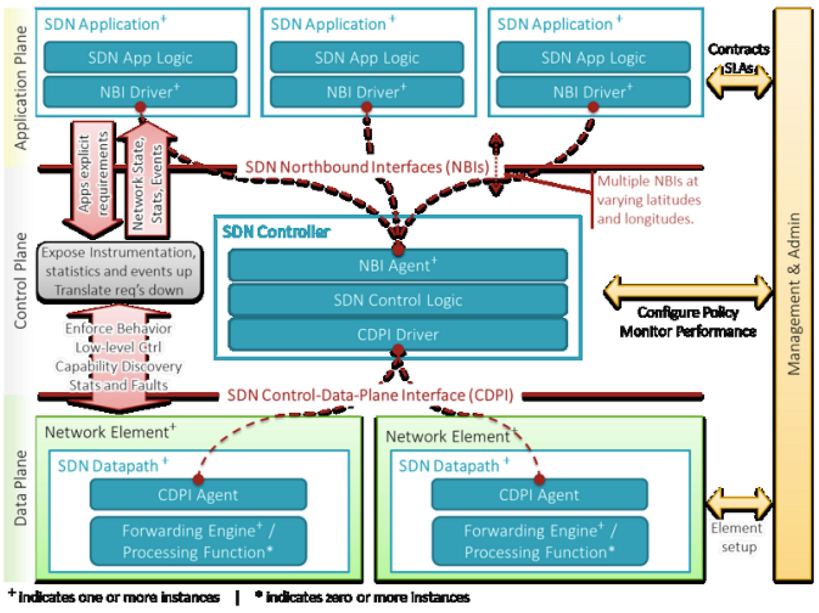

Application Plane
SDN Application+
SDN App Logic
NBI Driver+
Contracts SLAs
Apps explicit requirements
Network State, Stats, Events
SDN Northbound Interfaces (NBIs)
Multiple NBIs at varying latitudes and longitudes.
Control Plane
SDN Controller
NBI Agent+
SDN Control Logic
CDPI Driver
Expose Instrumentation, statistics and events up Translate req's down
Enforce Behavior
Low-level Ctrl
Capability Discovery
Stats and Faults
Configure Policy
Monitor Performance
Management & Admin
SDN Control-Data-Plane Interface (CDPI)
Data Plane
Network Element+
SDN Datapath+
CDPI Agent
Forwarding Engine+ / Processing Function*
Element setup
+ indicates one or more instances | * indicates zero or more instances

Figure 2.4 - SDN structure with all components

2.1.3 How SDN works

SDN incorporates a range of innovations, as well as functional isolation, virtualization of the network, and programming automation. SDN initially concentrate primarily on separating network control level from data plane. As long as the control plane defines how the packets flow over the network, the data plane transfers packets from one area to other.

A packet hits a network switch in a typical SDN case, and the rules that are implemented in the proprietary firmware switch state the switch to forward the packet. Some rules for packet management are sent by the central controller to the switch.

The data plane tool as we know switch, asks the controller for feedback when appropriate, and supply the controller with details for the traffic it manages. Switch transmits each packet to follow the same path to similar destination and handles all packets just the same method.

SDN uses a style also called versatile or fluid, where a switch makes an order for a packet which has no defined route to a controller. This method is distinct from adaptive routing, which increases routing requests through network topology-based routers and algorithms, not through a controller.

SDN's virtualization feature comes into action via a virtual overlay that is a relatively separate physical network on the network. Users can add end-to-end overlays to remove and classify network traffic. That micro-division is particularly practical for service providers and operators of multi-tenant cloud environments and cloud services, from the moment they can have specific policies for each tenant with a separate virtual network.

2.1.4 Benefits of SDN

An administrator can use SDN to adjust any switching rule if required-prioritize, override or even bypass forms of packets with an elementary level of control and protection [5]. For young people it is very useful in a multi-dwelling environment, as it helps the manager to handle traffic loads more flexibly and efficiently. This essentially allows administrator to use cheaper freight switches and have greater manage of network traffic than ever before.

There are other advantages as well end-to-end visibility and network management. Another benefit is that a network administrator has to assign policies only to the master controller that switches connected that to config a big number of devices. This feature offers a security benefit, as the controller is able to control traffic and assign security rules. If controller finds something suspicious in the traffic, it may retrieve or drop packages. SDN also virtualizes previously made equipment and facilities, resulting in reduced network footprint and lower operating costs.

SDN has also participated in the development of SD-WAN technology. SD-WAN uses the virtual cover of SDN technology, eliminates the connections of an enterprise across the WAN and generates a virtual network which can use any link it finds suitable for transmitting traffic.

2.1.5 SDN challenges

Security is a benefit to SDN technology and is a concern. The SDN Central Controller has only one failure point, and can be detrimental to the network if an attacker is targeted. Ironically, another problem with SDN is that the networking industry does not have a clear concept of "Software-Defined Networking". Various providers offer distinct approaches to SDN, from hardware-based models and virtualization systems to extremely comparable networking architectures and controller-free methods.

Many networking programs, like white-box networking, network isolation, network control, and programmable networking, are often unclear about SDNs. Although SDN will benefit from those technologies and processes and operate with them, it remains a separate technology.

SDN technology gained prominence around 2011, when implemented in tandem with the OpenFlow protocol. Acceptance has since been fairly sluggish, especially for businesses with smaller networks and fewer resources. Many companies also report that the cost of developing SDN is deterrent.

SDN's main adopters include service providers, network operators, telecommunications and carriers, as well as major companies like Facebook and Google, which have all the necessary means to tackle and contribute to emerging technologies.

2.1.6 SDN impact

SDN has profoundly affected IT system management and network implementation. As SDN develops, it does not only rebuild network infrastructure architecture, it also differentiates the way in which IT plays its role, as IT management becomes more involved in decision making and redesigns the entire IT infrastructure.

SDN architectures often use OpenFlow protocols to program network access. Because of this, businesses can deploy worldwide awareness software to access switches and network routers at the end of their networks, instead of proprietary firmware commonly used for configuration, control, protect and optimize network resources.

Although SDN implementations are spotted in all industries, technology impacts are greater in technology-related areas and in financial services. SDN affects the way telecommunications companies work. Verizon uses SDN to merge all new Ethernet and IP-based precision access routers at one network.

The aim is to make the edge architecture simpler so that Verizon can increase flexibility and operating performance in order to encourage new technologies and facilities. SDN will help Verizon handle the network better and eventually give its customers better coverage.

Progress in the financial services sector depends on joining up with a large number of trading participants, low latency and the extremely stable grid infrastructure that feeds the global financial markets.

Almost all financial market participants rely on older networks which are, in part, unpredictable, difficult to manage, slow down, and have enormous security issues. However, financial services organizations may build forecasting networks with SDN technology to make systems more efficient and effective for applications involving financial transactions.

2.1.7 SDN Controller

SDN controller is an application to a network architecture that controls the flow control to enhance network control and application rendering. The SDN controller platform usually runs on a server and uses protocols to warn switches that need to send packets.

SDN controllers' direct traffic in compliance with the promotion policies implemented by a network administrator, thereby reducing manual settings for individual network devices. By eliminating the control level from the network hardware and running it as a program, the central controller enables the automated network management and the integration and management of business applications. The SDN controller actually acts as a kind of network Operating System (OS).

The controller is the nucleus of a given network of software. It is located at one end of the network between the network devices and the applications at the other end. The controller must switch through any communication between applications and network devices.

The controller interacts with applications via northbound interfaces-such as firewalls or load balancing. In 2013 the ONF formed a working group specifically focused on deployed APIs and their creation. Nevertheless, the industry was not built on a standardized collection, mostly because the specifications for implementation varied so widely.

The controller uses a southbound interface to speak to individual network devices, historically similar to OpenFlow protocol. These south protocols allow the controller to configure network devices and to select the optimal network path for traffic in applications. In 2011 ONF developed OpenFlow.

2.1.8 Advantages and disadvantages of SDN Controllers

SDN Controllers have an important advantage in that the host controller knows all available network paths and can handle packets based on traffic requirements. It can automatically alter traffic flows and warn network operators about congested links due to the visibility of the controller in the network.

Companies must be using more than one controller to provide redundancy backups. Three tend to be a common number between open source and commercial SDN options. This redundancy will allow the network to continue to operate in the event of a loss of connectivity or sensitivity to the controller.

The controller acts as a failure point so its protection is essential for any software-defined network. Anyone owning the controller has access to the whole network. This means that network operators need to establish protection and authentication policies to ensure access is accessible only to the right people.

2.1.9 OpenFlow

In SDN Architecture, OpenFlow (OF) protocol is the key to entire topology. The group of few intellects began working on the specification of OpenFlow at Stanford University, and the ONF was formed in 2011 to support SDN and OpenFlow. There are several workgroups created which started standardizing this OpenFlow protocol.

Most of OEMs & vendors are supporting this OpenFlow protocol now. Here SDN controller may belong to an OEM where data / forwarding plane may belong to a different vendor. SDN controller with OpenFlow protocol can also interact with many types of network elements from different vendors.

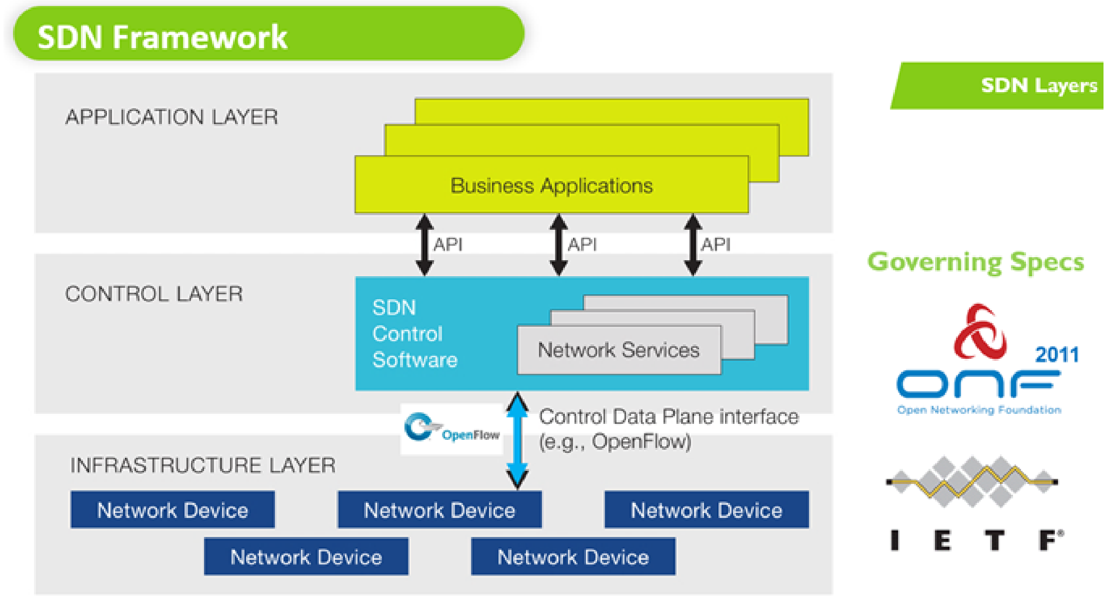


Figure 2.5 – OpenFlow on SDN

In our days there are several communication protocols amid the control plane and the data plane such as ForCES, IRS and NetConf. However, the OpenFlow communication protocol remains the most used on SDN network equipment.

OpenFlow is a multivendor standard defined by the ONF that acts as a relay between the controller and the network equipment of the transmission plane. It provides an open protocol for scheduling flow tables on different switches and routers. This is a standard protocol used by the SDN to provide switches with instructions for programming their data plane and obtaining information from these switches so that the controller can have a logical global view of the physical network. This view is used for all decisions that the control plane must make (for example, routing, traffic filtering, load sharing and address translation) [22]. This is because the controller can add, delete, or modify feeds by obtaining port and flow statistics and other information using the OpenFlow protocol.

OpenFlow protocol provides the control plane with a single system or southbound interface for the communication with the data plane. This is followed by all vendors such as Cisco, Juniper, HP etc. OpenFlow works between control and data or forwarding plane on standard

API specified. With help of OpenFlow the brain or controller can manipulate or change the routing tables, routing algorithm used by forwarding plane. This allows remote configuration of packet forwarding tables by adding, changing and removing the rules and behavior that suit packets. This OpenFlow needs to be supported by both controller and forwarding routers used in network. The existing devices can be also evolved to support OpenFlow, for example, all existing traditional switches routers etc can become data forwarding plane can receive instructions from 3rd party controller. Here we can transform our existing network to support SDN.

Since controller is exposed to business applications with help of APIs, this gives us opportunity to innovate multiple features & functionalities in the network as per customer requirements.

OpenFlow controller can communicate with more than one OpenFlow switches. The communications between controller and switch are secured by the Transport Layer Security (TLS) [25] based asymmetrical encryption and unencrypted TCP connections. As we can see in Figure 2.6, the links may be in-band or out-of-band.

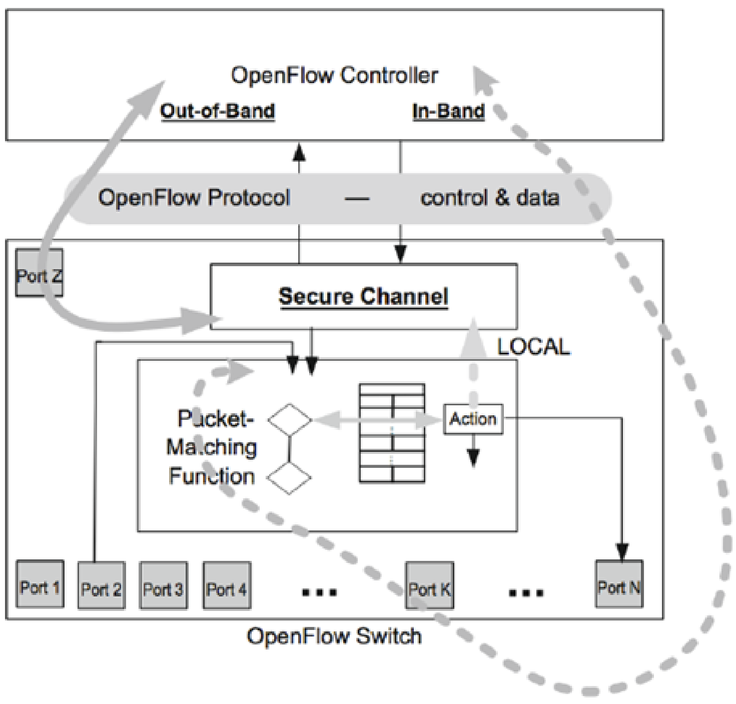


Figure 2.6 – OpenFlow Connections

In-band:

From a part of the OpenFlow data plane which is port K, the controller's OpenFlow messages arrives. The OpenFlow packet-matching logic can manage such packets. The flow tables will have been built to forward this OpenFlow traffic to the virtual port of LOCAL (results in the messages being transferred to the mechanism of the secure channel).

Out-of-band:

The secure channel connection goes into the switch via port Z. Some legacy network stack must transmit the OpenFlow messages to the secure channel phase in the switch via the secure channel. Protected out-of-band channel is only appropriate for an OpenFlow-hybrid switch. The OpenFlow protocol allows programmers/network administrators/researches to run networking experiments on real hardware in real networks. The main idea behind OpenFlow is to provide an abstraction of hardware functionality that would be compatible with most of the hardware available.

To summarize OpenFlow gives remote access to the control plane in switches and routers, which connect to a controller on TCP port 6653 or with earlier versions, port 6633. The controller is then able to add, modify and remove packet matching rules and actions, which are executed on the network equipment at "wire speed".

The OpenFlow switch

An OpenFlow switch consists of three sections or more:

- A **Flow Table** to tell the switch how to process the flow with an action associated with each flow entry.
- A **Secure Channel** that links the switch to a remote control mechanism (known as the controller) that allows commands and packets to be sent between the controller and the switch.
- The **OpenFlow Protocol**, which provides the controller with an open and standard way to communicate with a switch.

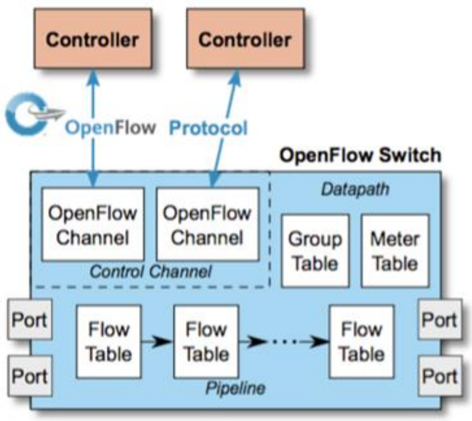


Figure 2.7 – Main components of an OpenFlow switch

The most modern Ethernet switches and routers contain flow tables that are used to perform layer 2, 3 and 4 transfer functions, indicated in the packet headers. Although each vendor has different flow tables, there is a common set of functions for a wide range of switches and routers. This common set of functions is leveraged by OpenFlow, the protocol between the OpenFlow Central Controller and the OpenFlow switch that, as shown in Figure 2.7, can be used to program the switch transfer or routing logic [22].

An OpenFlow switch is called any network device that supports the OpenFlow protocol, such as a switch, router, or access point. Each OF device maintains a flow table (that is, a switch table or routing table), constructed from TCAM (Ternary Content Addressable Memory), which indicates the processing applied to any packet of a certain flow [23].

OpenFlow is supposed to be a way to test new routing or transfer methods to build these flow tables. The way it does this is to establish a secure SSH tunnel between the OpenFlow switch and the controller (Figure 2.8). The controller will work, regardless of this new routing process. When a new stream arrives, the OpenFlow switch sends the first packets to the controller. The controller then constructs an entry in the flow table (a flow or switch rule) to handle the remainder of that connection [24].

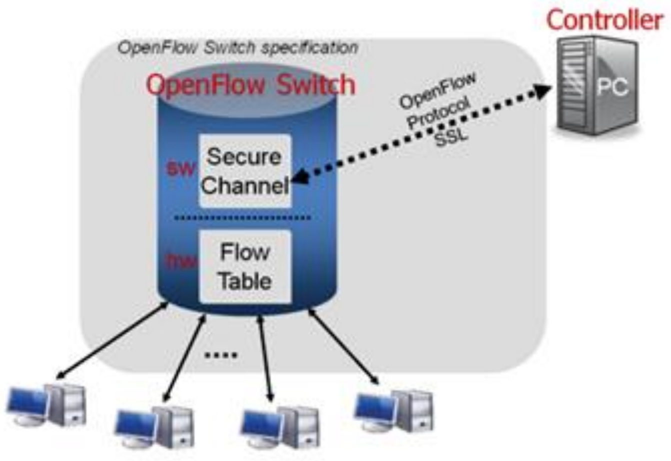


Figure 2.8 - OpenFlow Switch

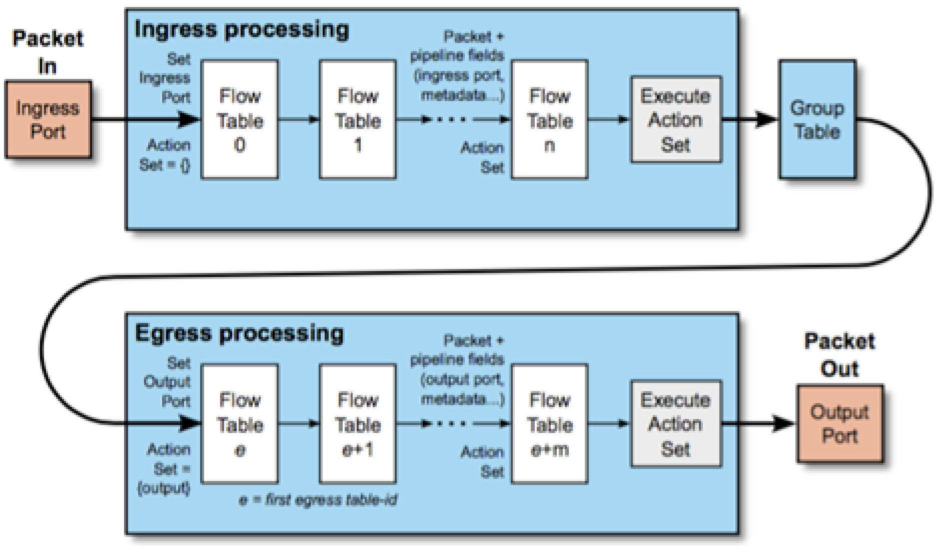


Figure 2.9 - Packet flow through the processing pipeline

OpenFlow Security Issues

OpenFlow security is an important aspect that poses a threat to SDN. *"OpenFlow protocol does not implement security by itself, therefore the transport protocol or a protocol underneath it must provide security if it is needed" [36]*. In SDN, security remains the responsibility of administrators, as there is an option: *"The OpenFlow channel is usually encrypted using TLS, but may be run directly over TCP" [36]*. The above specification recommends as using TLS 1.2 or higher. Controller, switch and channel protection are the key advantages of OpenFlow security as they consider that the main threatened objects must be protected from various attacks. In the below table we present the security challenges based on these key parts of OpenFlow.

OpenFlow Part	Attacks
Switch	Scanning, spoofing, hijacking, DoS, tampering, replaying attacks, etc.
Controller	Scanning, spoofing, hijacking, DoS, tampering, replaying attacks, etc.
Channel	Man-in-the-middle attacks (MITM), hijacking, repudiation, network monitoring, etc.

The security issues mainly related to the vulnerabilities of the switch, where intruders can use it to conflicting the SDN. Such OpenFlow switch and flow table provide network management, routing and access control information are probably an important target. Specifically, intruders can target network components inside the same. Initially, intruder can attempt to achieve forbidden physical or virtual network access, or to take the control of a server which is already connected to the SDN, and then attempt to launch attacks to destabilize network components.

At the switch, attackers will compromise SDN with malicious behavior and inputs like scanning, spoofing and so on.

- Intruders can use scanning as first move to knowing the basic data of the entire network and to collect sensitive information.
- Through Spoofing, intruders disguise themselves effectively as something else by falsifying data and thereby obtaining an illicit profit.
- The intruders intend to take control of a communication segment through Hijacking, which is a lot more effective from Spoofing (because intruders be able to manipulate an item entirely through spoofing). If the intruders manage the switch successfully in OpenFlow they will learn all the flow rules and contact data.
- Tampering refers to unauthorized alteration of network details (modification of streams in flowcharts).
- Denial of Service (DoS) attack attempts to make the intended users inaccessible a network utility that involves Flooding, Smurf and DNS enhancement.
- Besides the typical attacks above, attackers may use weakness to reach the same targets.

Unlike switches, intruders may perform different malicious behaviors and intrusions at the controller, like scanning, hallucinations, hijacking etc.

- Intruders can use scanning as first move to knowing the basic data of the entire network and to collect sensitive information.
- If the intruders successfully carry out a Spoofing attack, they can imitate the controllers.
- At control level, if attackers handle controller successfully with Hijacking, they can freely extract sensitive information, such as passwords, contact details etc. They may also infringe or alter whatever data, and redirect traffic to every shipment. OpenFlow will be completely abused in this case.
- Like Tampering attacks on data plane, intruder with the method of spoofing on northbound API messages or southbound messages to network devices they want to introduce new flows to the controller. If an intruder moves through the legislative auditor successfully, they will be able to allow traffic via the SDN at their own will and potentially circumvent security policies.
- An attacker can attempt to execute a DoS attack or use other methods Leading the controller to crash.
- Other attacks may also be used to add to the confidentiality, credibility and availability of the SDN, like recurring attacks, privileges etc. These attacks will take advantage of the relative vulnerabilities to accomplish the same aim at controller level.

Intruders may make diverse attacks on the channel to export sensitive information, like MITM, monitoring, repudiation, etc.

- MITM attacks are feasible and successful, without strong confidence in the mechanism. This type of attack will jeopardize all facets of the CIA (confidentiality, integrity, and data availability also known as the CIA triad) [37].
- With the network monitoring assault, intruders will attempt to learn the data between switches and controllers by watching the OpenFlow channel.
- Repudiation means a person will refuse to take part in a conversation. As a result, non-denial helps to ensure no rejection occurs. This problem can also be triggered by attacks by MITM, by occupying the space between two parties and thinking they are the other party.
- A lot of attacks will promote the effectiveness of MITM attacks, such as repeated attacks. OpenFlow applications may be harnessed to launch attacks with potential vulnerabilities. Spoofing attacks and Hijacking to accomplish this.

Often, with OpenFlow evolving, attacks become more complex, leveraging vulnerabilities at different levels. Such security issues could seriously threaten the protection of such networks. For example, by targeting OpenFlow switches and controller, hijacking and spoofing attacks can endanger more than one part of the CIA triad. It looks like this trend will continue and attackers might in the future be built to invade and compromise OpenFlow by attackers. Successful countermeasures therefore need to be built to provide security in various layers.

SDN is an evolving architecture tailored to the fluid design of the devices today. It can help companies speed up production and delivery of applications, and significantly reduce IT costs through automated policy process automation. It also allows for cloud architectures, offering automated on-demand delivery and mobility on scale.
We must secure flows at switches and protect various vulnerabilities at switches. In

OpenFlow the controller is a key element, access control and enforcement of policies are our main aims to defend against various intrusions.

Channel is a very important part that links the controller and the switches to OpenFlow. Many channel levels and security upgrades will actually benefit both controller and switches. We need to build a safe and clean channel of communication.

2.2 Network Function Virtualization (NFV)

Network Functions Virtualization (NFV) [28] is a network architecture concept that uses virtualization IT technologies to virtualize entire categories of network node functions into blocks that can be connected or joined together to create communication services.

In 2012, the European Telecommunications Standards Institute (ETSI) NFV Industry Specification Group (NFV ISG) launched NFV Technology to allow customers to migrate networking features from dedicated vendors and proprietary hardware to applications hosted on commercial off-the-shelf (COTS) platforms [29]. The key concept is to provide Cloud-based Virtual Machine Network (VM) services where each VM can perform various network functions (firewall, intrusion detection, deep packet inspection, load balancing, etc.). The key advantages of deploying network services as virtual functions are:

- Flexibility in the allocation of network functions into general-purpose hardware
- Rapid implementation and development of new network services
- Support for multiple versions of service and multi-tenancy scenarios
- Reduction in capital expenditure (CAPEX) costs by managing energy usage efficiently

- Automation of operating processes, thus improving efficiency and reducing operational expenditure (OPEX) costs.

We take the description below for NFV from ETSI white paper 0 “Network Functions Virtualization seeks to transform the way network operators design networks by implementing modern IT virtualization technologies to combine multiple forms of network equipment on industry standard high-volume servers, switches and storage located in data centers, network nodes and end-user premises, as shown in Figure 2.10. It includes implementing network functions in software that can run on a variety of industry standard server hardware and can be transferred to, or installed in, different network locations as needed without the need for new equipment installation.”

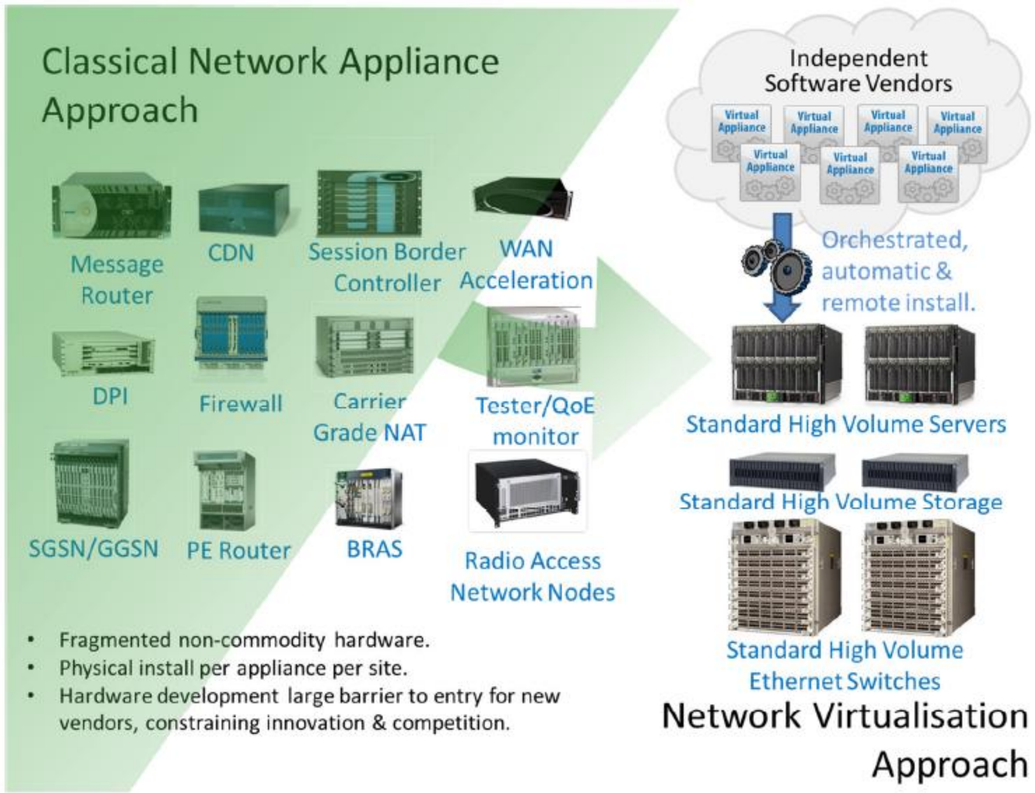
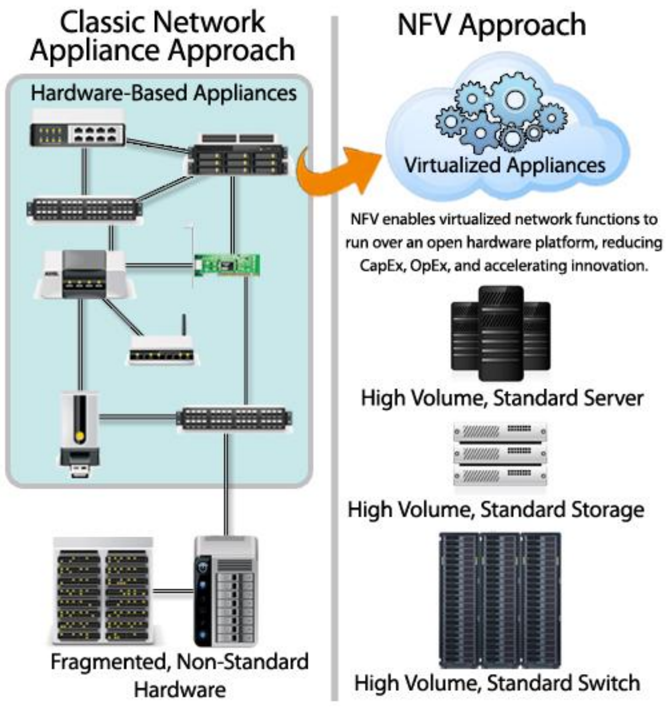

Classical Network Appliance Approach
Message Router
CDN
Session Border Controller
WAN Acceleration
DPI
Firewall
Carrier Grade NAT
Tester/QoE monitor
SGSN/GGSN
PE Router
BRAS
Radio Access Network Nodes
Fragmented non-commodity hardware.
Physical install per appliance per site.
Hardware development large barrier to entry for new vendors, constraining innovation & competition.
Independent Software Vendors
Virtual Appliance
Orchestrated, automatic & remote install.
Standard High Volume Servers
Standard High Volume Storage
Standard High Volume Ethernet Switches
Network Virtualisation Approach

Figure 2.10 - Vision for NFV

NFV is a way to minimize costs and speed up the service delivery for network operators by decoupling functions such as a firewall or dedicated hardware encryption and transferring them to virtual servers. Rather than buying costly proprietary equipment, service providers can buy cheap switches, storage, and servers for virtual machine operation, performing network functions. It breaks down several functions into one physical server and reduces costs. The service provider will actually create a new virtual machine to perform the task if a customer wants to add a new network function.

For example, instead of installing a new hardware appliance across the network to allow network encryption, it is possible to install encryption software on a standardized server or on a switch that is already on the network.

This virtualization of network functions eliminates reliance for network operators on dedicated hardware equipment, and allows increased scalability and flexibility across the whole network. Unlike a virtualized network, NFV aims to discharge network functions only, rather than the whole network.


Classic Network Appliance Approach
Hardware-Based Appliances
Fragmented, Non-Standard Hardware
NFV Approach
Virtualized Appliances
NFV enables virtualized network functions to run over an open hardware platform, reducing CapEx, OpEx, and accelerating innovation.
High Volume, Standard Server
High Volume, Standard Storage
High Volume, Standard Switch

NFV eliminates the need for dedicated hardware for network deployment and management by unloading network operations into software that can function on industry-standard hardware and which it can operated from anywhere in the network of the operator.

Separating network functions from hardware offers the network operator numerous benefits, which include:

- Reduced room available for network hardware
- Reduce power demand on the network
- Reduced maintenance costs to the network
- Make network updates faster
- Lengthy network hardware lifecycles
- Reduced costs for the repairs and equipment

NFV is based on but not the same as conventional server-virtualization techniques, as those utilized in business IT. In addition to standard high-volume servers, switching and storage devices, or even cloud computing infrastructure, a VNF can involve more than one virtual machine running different software and processes, rather than having specific hardware devices for every network mode.

A virtual boundary controller for example can installed to secure a network without the usual cost and complexity of purchasing and implementing physical network security modules. Virtualized load balancers, firewalls, tools for intrusion detection, and WAN accelerators may be included in NFV examples.

Till now, we have covered NFV which helps us building the agile and flexible compute resources. Since NFV is all about three main concepts: Softwarization, Virtualization &

Orchestration. SDN is all about networking control functions for routing & policy definition in automated way for ensuring IP/Network reachability well in time. All systems in SDN & NFV utilize network complexity, and do so differently.

SDN provides a new level of programmability and abstraction to network layer which is playing phenomenal role in automated networks of future. Let’s take deep dive on SDN & it’s working. For instance, we can see the building blocks which represent the NFV & host multiple applications. Just like we can remove, replace or keep new tenant in build, similarly we can host applications in NFV as well.

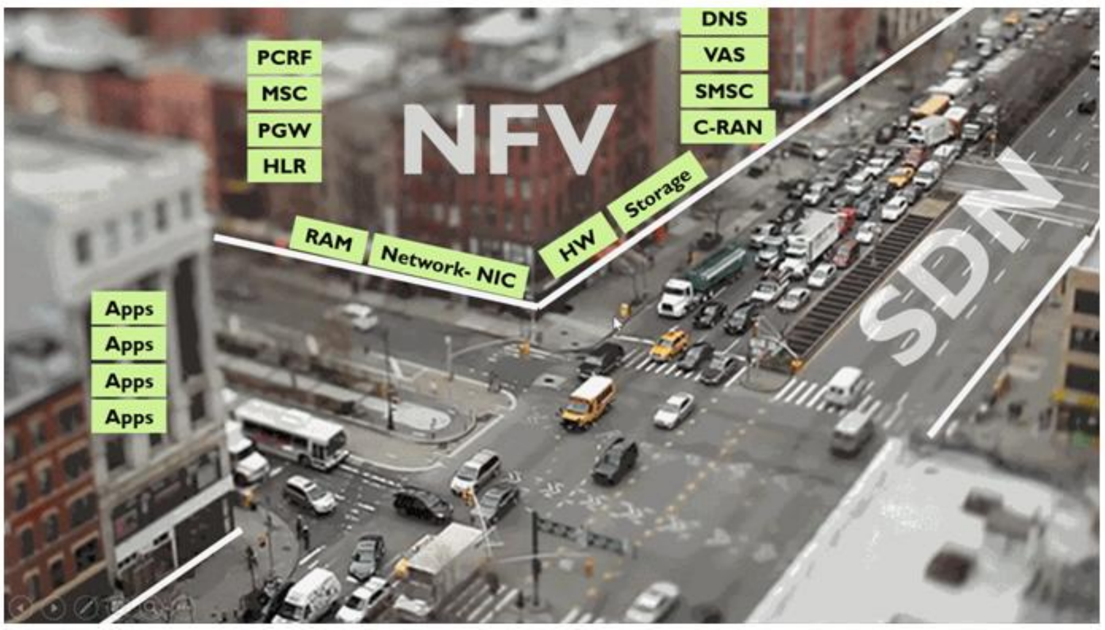


Figure 2.11 – SDN & NFV

We can see traffic on road representing SDN whose main responsibility is to carry traffic to right destination in automated way.

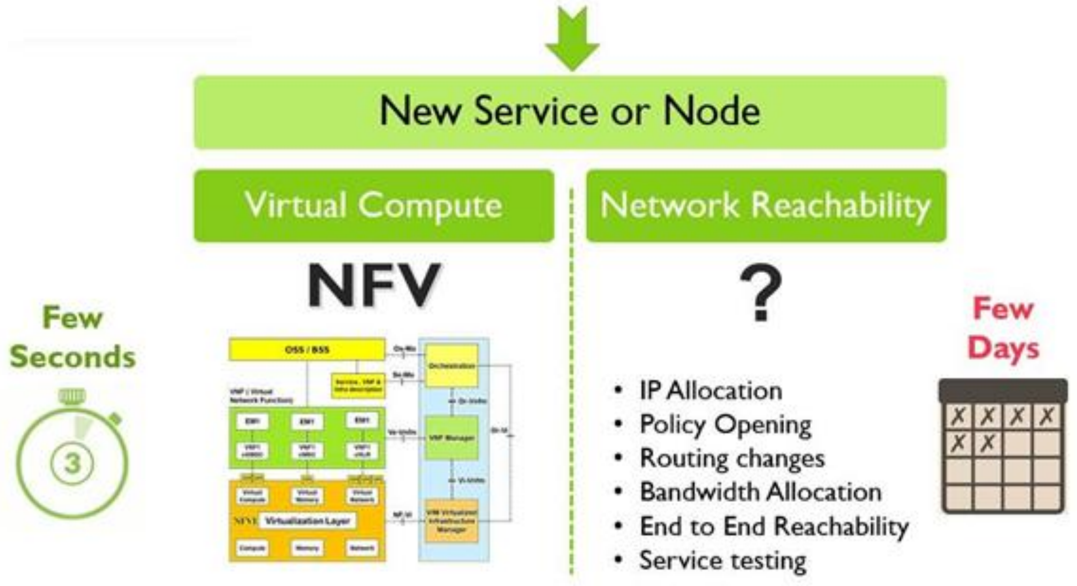


Figure 2.12 – NFV or?

Let's understand why we need SDN & how it is helping. Okay, we see NFV aid in virtualizing networks that allow us to launch rapidly and eventually reduce the time to create new service / new node to a few seconds. For example, with the help of orchestrator & automation, we can create vMSC or vMME in a few seconds & make it ready.

Yet, another problem is access to the network. We need multiple things for any new node or device, such as IP allocation, bandwidth allocation, policy opening, routing changes to achieve end-to-end connectivity, and service testing. It's not all automatic so it takes a lot of time to plan design, make adjustments in every router/switch and do it.

For typical cases, this will take a couple of days or weeks to complete IP routing & enable all necessary links to end reachability. SDN supports us here by making this network versatile and programmable to route / switch.

Benefits & Features of SDN based networks:

- Directly Programmable
- Flexible, Dynamic & Agile

- Centrally Managed
- Abstracts Network
- De-Couples Control & Forwarding plane
- Based on Open Standards & Vendor Neutral

NFV, as shown in Figure 2.13, is interconnected with SDN, but not contingent on it. NFV can be applied without the need for an SDN, although it is possible to combine the two principles and technologies to potentially add greater value.

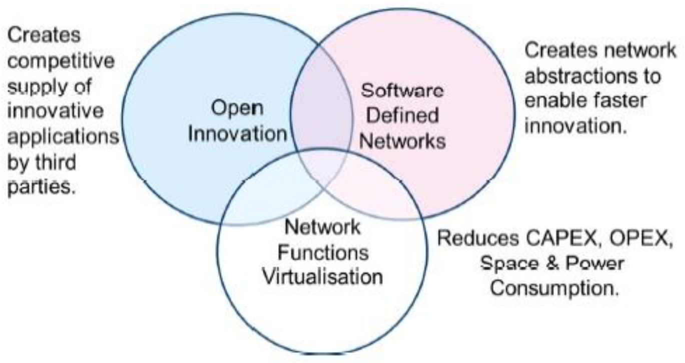

Creates competitive supply of innovative applications by third parties.
Open Innovation
Software Defined Networks
Creates network abstractions to enable faster innovation.
Network Functions Virtualisation
Reduces CAPEX, OPEX, Space & Power Consumption.

Figure 2.13 – NFV interaction with SDN

NFV targets can be accomplished through non-SDN methods, building on the current techniques used in many datacenters. But solutions that focus on SDN's suggested separation of control and data forwarding planes will improve performance, ease compliance with current installations, and promote operational and maintenance procedures.

NFV will enable SDN by having the hardware to operate the SDN applications on. In comparison, NFV strongly aligns with the SDN goals for using generic servers and switches. It plans to work closely with SDN progressive organisations, such as ONF, whose work will be explicitly recognized.

Differential of NFV and SDN

NFV and SDN are similarly linked but not the same. Those words are sometimes used synonymously incorrectly. The main points of each are listed below, in order to evaluate both SDN and NFV for their similarities and differences.

- SDN struggles with the substitution of direct management of structured network protocols. As a result, SDN aims to reduce the complexities of distributed network control protocols by designing an overall device with the simplicity. As such, this significantly improves flexibility, as only one instance needs to be updated to reflect the change.

  Thus, SDN separates the planes for network control and forwarding and offers a central view for more effective network service implementation and operation.

- NFV substitutes proprietary NEs network elements for software running on standard servers. NFV is focused on improving the network services themselves, in other words.

  This strategy decouples the network functions from proprietary hardware, putting them on more common servers or computers so that these functions can run in software to provide greater flexibility for service, modifications and updates.

While the two technologies are similar in as much as they both use software, they are very different in reality, pursuing different goals. The similarities between NFV and SDN mean both strategies can be used in a mutually beneficial way on the same network.

2.3 Microservices / Containers

As we know NFV moves traditional network functions such as routing, firewalls and load balancing, out of physical hardware to run as Virtual Network Functions (VNFs). Ideally, the

NFV can pack some VNFs together and streamline the process of providing and upgrading network or application features. NFVs and microservices share a similar modular architecture and VNFs can even be developed in microservices architecture.

Microservice or the architecture of microservices is an application development approach in which a broad app is constructed as a suite of modules or services. Every module provides support to a particular task or business objective and uses a simple, well specified interface to communicate with other service sets, such as an Application Programming Interface (API).

The application is divided up into components in the microservice architecture. Every service usually routes a specific process and maintains its own database. A service may produce notifications, log data, prop up User Interfaces (UIs), manage user certification and carry out specific other activities.

The model of microservice offers a more collaborative approach to software development for development teams. Microservices allow the isolation, reconstruction, redistribution and autonomous management of each service. For example, if a system fails to produce reports correctly, the problem with that particular service might be easier to identify. This specific service, independent of other services, could then be tested, restarted, corrected and replicated as needed.

Microservices Pros and Cons

Pros:

- easily distributed
- need less development time
- be able to scale rapidly

- be reused in various projects
- provide stronger insulation of faults
- can used in very small teams
- function good with containers

Cons:

- potentially too much granularity
- extra effort tailored for service contact
- lag during heavy use
- dynamic testing

Microservices and container systems

A container is an independent and functional package of software, including all the dependencies required for independent operation. These are cut off from the rest of the underlying software, and a lot of them may be used in the same environment. Growing service is individually packaged in a microservice architecture under the same setting as itself or the relevant servers.

A VM may be used to create small businesses as a different solution to containers. A VM simulates computer systems designed to generate physical computer functions. Probably, every service could use a VM to host a target feature. However, for small businesses, they are avoided because of the separate Operating System (OS) and other requirements required for each VM. Containers are far more resource productive, as the application needs only the underlying code and related dependencies.

Container technology has evolved rapidly in recent years, making the Kubernetes [31] ecosystem the most viable set of solutions that can simplify, accelerate, and reduce the cost of deploying and operating cloud networks. For telecom operators, this is like discovering the missing link that will enable them to achieve cost savings and service innovation promised all those years ago by the early vision for the NFV.

Kubernetes is an orchestration system for automating application development, scalability and management. Within the ETSI's NFV architecture, it performs the roles of Virtual Infrastructure Manager (VIM) and VNF Manager (VNFM). But beyond the application lifecycle management.

Operators who have built NFV VM strategies and ETSI's architecture may discourage the idea of introducing another new concept into an already complex NFVI. But the beauty of Kubernetes is that it can provide a common environment for all types of cloud infrastructure.

In the Kubernetes ecosystem, there is a gem of a project called KubeVirt [32] that provides a way to get an existing VM and run it inside a container. With KubeVirt, we can deploy and manage applications that include any arbitrary mix of native container workloads and VM workloads using Kubernetes. The underlying infrastructure can be OpenStack [33], VMware [34], Bare Metal [35] or any of the major public clouds, such as Azure, AWS or Google. In this way, Kubernetes allows us to achieve a unified strategy for NFV that is absolutely exciting.

Security of microservices

Microservice architecture will mitigate some of the security issues resulting from monolithic apps. These simplify security monitoring as the various parts of an app are separated. In one department a security breach may occur without affecting other areas of the project. When used in containers, microservices provide protection to Distributed Denial of Service (DDoS) attacks, mitigating the deployment of infrastructure with too many server requests.

Nonetheless, issues remain when it comes to securing microservices apps such as:

- More parts of network are exposed to vulnerabilities
- We have more security issues when there is no follow-up to application updates
- There is more area of attack, through multiple ports and APIs
- Third-party software is not easy to controlled
- For each service protection needs to be maintained

Microservice developers have developed techniques to mitigate security concerns. Using a security scanner to be proactive, using access control restrictions, safe internal networks, like Docker environments, and run silently, interacting with all parts of the operation.

Dockers

The Linux kernel provides a containerization mechanism through which the user can create many structured containers such as VMs at a level above the Linux operating system, known as Linux Based Containers (LXC) [39]. The LXD [40] is essentially a revamped version of the LXC, offering more flexibility and features. In essence, it uses the LXC library to provide containers at a low operating cost. Docker [41], written in the GoLang programming language, has become synonymous with container technology because it was the most successful tool for their dissemination. However, container technology is not new, as it has been integrated into Linux in the form of LXCs for over 10 years, with the latter being a standard tool for workflow and cloud deployment.

The most common free and open source container management system is Kubernetes, which is a software project from Google. The Kubernetes we've been talking about provide mechanisms for developing, maintaining, and scaling container applications.

Linux Containers and Docker in particular are an essential path to virtualization on embedded devices, as they can bring advanced software development capabilities to these devices. VMs have greatly contributed to automation and productivity over cloud development in recent years, but due to their inevitable disadvantages they are not suitable for use in embedded systems. Virtualization, however, does not present any disadvantages at the operating system level provided by Docker (and Linux Containers) and thus can be used for IoT [70] interconnected devices. A Docker container has its own unique Namespace (PID, UID, filesystems, network) so that the processes running within it can be completely isolated from the operating system on which it runs, and share the resources made available to them, without contacting the available resources of other containers. In essence, Docker is a software or tool that offers containerization.

2.4 Deep Packet Inspection (DPI)

DPI is an advanced packet filtering tool, running on the Open Systems Interconnection (OSI) reference model's seventh layer (application layer) [26]. Using DPI effectively enables its users to detect, identify, categorize, move or interrupt unwanted code or data packets.

Generally, this tool is more efficient than standard packet filtering which only controls the headers of packets. It checks the packet data segment when it comes to a checkpoint trying to find non-compliance, intrusions, spam, viruses or further predefined factors to decide if the packet will go on or has to be routed to another location. Companies with this tool be able to enhance their network management systems without deeply investing in core network technologies.

DPI is also known as full packet inspection and extraction of information. DPI combines the features of an IPS and a traditional firewall IDS. Communications engineers and service providers often use this to control network traffic. With deploying valuable network resources in high priority data packets and messages, they can manage network traffic. IDS can detect intrusions, but is unable to prevent an attack. IDS can use Deep Packet Intrusion to support:

- Gather more details about the attack.
- Identify such Attack signatures and patterns.
- Controls network traffic, such as FTP root access, Telnet or basic content to HTTP.

IDS which based on DPI can inspect package contents and provide more information. With this technology IDS is able to detect and even monitor an attack faster. IDS compare the attack to a database to suit the attack signatures and allow DPI to act in accordance with its policies.

In contrast, IPS can detect and block attacks in real-time. Some IPS solutions also use DPI technology to prevent attacks. Could:

- Prevent signatures of certain attacks.
- Safeguard ourselves from certain bugs and exploits.

In most cases the security component of traditional solutions can be enhanced by a DPI. DPI combines the IDS / IPS capabilities with a conventional state-of - the-art firewall, allowing it to identify attack variants that cannot be detected by these tools.

Typically, DPI comes as a security feature or as a virtual DPI that is deployed to server. While using a dedicated security system / DPI is the correct application, you may also want to incorporate DPI as a service or software. Despite its many advantages, the following drawbacks apply to DPI:

- In addition to defending against existing types it is capable of creating new vulnerabilities. While it is effective against attacks like DoS, buffer overflow and types of malware, DPI may be used to cause attacks like these.
- The difficulty and encumbrance of firewalls and other security-based software are grown up.
- Demand updates and modifications to continue to be perfectly effective.
- When we run DPI the cpu of our computer it works at high rates so we can't run other applications. So, this greatly affects the operation of our machine.

Governments use DPI to track and defend the cyber geographic frontiers. DPI was also used to monitor user actions, to protect the security of LAN and WAN, and to remove malware and software that is suspicious. Additionally, service providers use DPI to monitor their Web browsing habits. And companies which focus on targeted ads use these customer data.

Network Intrusion Detection Systems (NIDS) run DPI to detect malicious content within the payload packets routed through the network. Snort [27] is a well-known NIDS open source that supports packet search and mapping. The patterns searched for during the DPI process usually consist of either strings or regular expressions.

String matching is a key element in most DPI machines. In many applications (such as Snort), string matching is used during the recognition process of both pattern types. When patterns are regular expressions, the string matching is first performed as a pre-filter and constitutes the bulk of the work performed by the engine. This procedure is commonly used, since regular expression machines are ineffective in comparison to string matching engines.

DPI provides application-aware networking, while SDN provides network-aware applications. Although SDN would change the generic network architectures fundamentally, it will have to work with conventional network designers to ensure high ability. The new SDN-based network structure will take into account all functionality currently available on separate devices or applications except for the main propulsion devices (routers and switches) including DPI, protection devices. DPI is so critical because it can help big businesses make their safety noticeable by shaping their traffic.

Not every router (or firewall) can manage inspection of the deep packets, however. Technology requires considerable resources to operate, so it is not common in Small to Medium Enterprises and can be a real traffic jam if not properly regulated. A DPI router needs to be powerful enough to be able to open, inspect, wrap and resend each packet. Only large companies, governments and suppliers of telecommunications services have the means to make this technology work.

Ntop [38], a network monitoring kit, is a tool for performing DPI and analyzing traffic to our network. It has a number of packet collection tools, traffic control, network detector, traffic processing, and DPI. NTop DPI is carried out by nDPI with the help of NTopng.

NDPI is an open source and extensible, OpenDPI based library. Using this tool, our network administrator may block unique traffic flows, hosts or network protocols. But since nDPI is only a library, it must be used to implement the rules with other programs, such as ntopng and nProbe cento. NTopng is a web interface monitoring tool that gathers traffic passively from a network interface to show its status and performance.

We may enforce L7 policies with ntopng and ndpi, such as configuring the release of specific applications within a network subnet.

3. Security in SDN-Based Enterprise Networks

In the previous chapter we tried to analyze the general elements that make up SDN networks and how they interact with some other modern technologies such as NFV and DPI.

The interconnection of various technologies as we saw in Chapter 2 is quite important and promising for the future of these networks but one of the main issues and we would say the most important thing that prevails is security in SDN networks.

So on this occasion in the following paragraphs we will try to give an analysis on the security of SDN networks.

3.1 Cyber attack

Cyber attack is any malicious activity that takes place via computer or network and is intended to modify, destroy, steal, intercept or even unauthorized access to the information of the rightful owner. A cyber attack target can be a computer information system, a computer network or a shared personal computer. A cyber-attack can come from a state, a group, a society, an organization or even an anonymous source.

3.2 Most important cyber attack techniques

Different types of attacks can be categorized according to the way they affect a system and the methodology followed with the best-known being Malware software, Phishing, Man-in-the-Middle, Denial of Service (DoS), SQL injection, Zero-Day exploit, Cross Site Scripting, Advanced Persistent Threat and Credential Reuse.

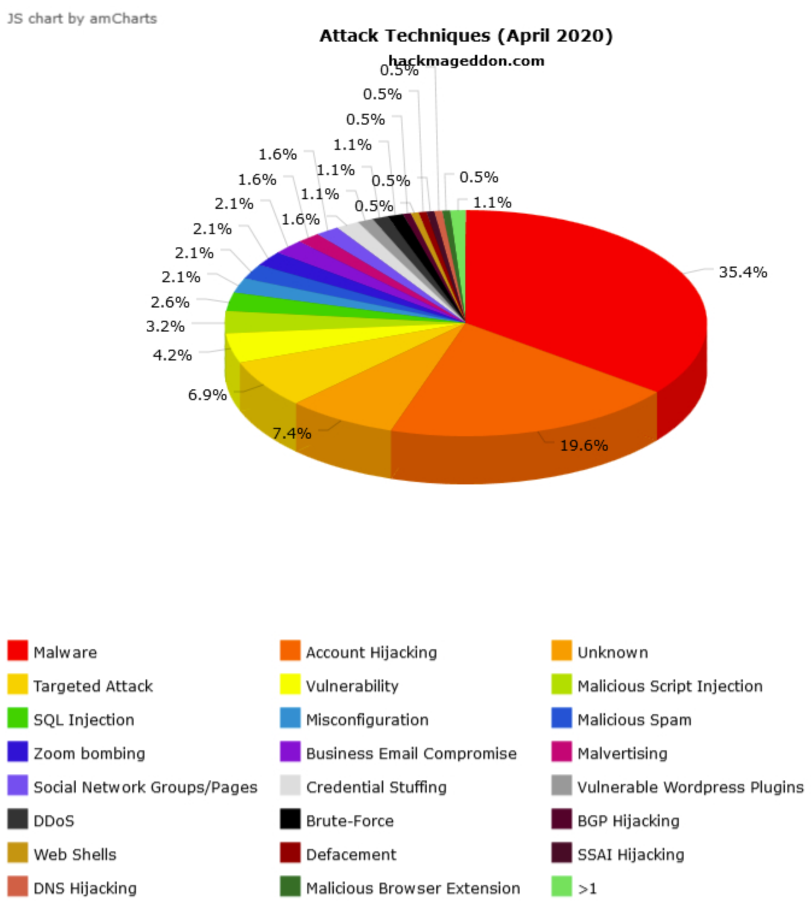


Figure 3.1 – Cyber Attack Techniques [61]

As shown in the chart above, hackers mainly use Malware to carry out their attacks, which is a set of malwares that includes software such as: trojan horses, worms, rootkits, ransomware and Potentially unwanted Programs (PuP). These software exploits potential security vulnerabilities and infiltrates systems, creating many serious problems as they operate behind the scenes. Two important points to note, from the statistics of technical attacks, are the large percentage of Account Hijacking amounting to 19.6% which are due to account violations due to COVID-19 as well as the percentage of hitherto unknown methods of attack that amounts to 7.4%.

3.3 SDN Security

Security is a major challenge for SDN networks. To date, the business and academic community have had limited discussion on this area. Therefore, in order for SDN technology to become more widely accepted in its use, more focus is needed on the security part, and for this reason the ONF organization has formed a corresponding team to coordinate and study all the issues related to it. The most important security issue concerns the central control of the network, which can be exposed to attacks, given its value for the whole system, the integration of old-style protocols, connections between sectors, etc. In general, there are vulnerabilities on SDN platforms. For example, security questions have been raised about the mechanisms for identifying and authorizing auditors-applications, whether they have the ability to allow multiple organizations to access network resources while providing them with appropriate protection.

As SDN philosophy is based on software, vulnerabilities in the code also have a significant impact on SDN security. In addition, SDN offers plenty of options for implementing security checks through the applications of an SDN controller. Such software solutions can allow more flexible security checks in dynamic and virtual network environments, thus providing a more practical means of controlling software security.

3.3.1 SDN architecture security issues

Each layer of SDN architecture has its own security issues which are:

Application layer

Network control using software is the basic policy of SDN. Most applications use network components and have access to all of its components.

- Enclosed applications: An application with a chain execution could enter a state of endless loop or gain unauthorized access to another application.
- Audit data manipulation: Applications if they have access to the auditor's resources could use them for malicious activities.

- Manipulation of controller-transmitter messages: Using control messages, connections to the transmitters could be terminated. It would also be possible to rewrite the flow rules.
- Trust: Trust between the auditor and the applications are essential and are a field that has not been fully researched.
- Third Party Applications: Developing third-party applications can lead to conflict control by the controller due to the non-normalization of the field.
- Authentication, Access, Responsibility: These three fields are a major challenge in SDN networks due to third-party applications. Malicious applications can gain access to network resources by affecting the level of applications and control.
- Resource depletion: Applications can target the depletion of controller resources such as memory and processor.

Northbound interface

Connecting the application field to the controller can have many effects on SDN architecture. That is why the northern interface can be the target of an attack.

- Lack of normalization of interface: At the moment there is no normalization of the northern interface and its independent development by the auditors can put them at great risk of security from attacks.
- Poor interface design: An incorrectly developed interface can be manipulated by applications to terminate an application that should work, putting it in a state of disrepair. This combined with the potential lack of authentication of applications puts the network at great risk.

Control layer

What makes SDN stand out is its centrally controlled architecture that has many advantages. However, it is a unique point that, if exposed to an attack, puts the entire network at risk.

- Packet-in message tampering attack: Authorized switches can send fake packet-in messages to harm the controller through the following attacks:
  - Targeted denial-of-service attacks: The attacker can put the controller in a state of inactivity through a packet-in flood. DoS attacks can be performed in many ways. This can also be done by overloading the transmitter by sending many packets that will lead to not finding flow rules. The transmitter will then send all the questions to the controller and there is a possibility that due to the memory of the transmitter it will reject the voluntary movement.
  - Poisoning of the auditor's face (spoofing): Through false ARP packages, the auditor's view of the network could be affected. The process by which the controller learns network devices is called host tracking service and is subject to spoofing attacks where false MAC, IP or VLAN tag addresses are declared.
  - Network topology poisoning: False LLDP (Link Layer Discovery Protocol) 0 messages could create false topology for the network and block traffic.
  - Side attacks: The impact of packet-in messages could be used to retrieve valuable information.
- Illegal access: Illegal access to the auditor is one of the biggest threats. If it manages to be manipulated, an entire network of attack bots could be set up to expand the attack.

- Illegal access to data: The data used by the auditor is quite valuable and needs security to avoid any configuration or interception. Any change to a variable can simply disconnect the switches from the controller.

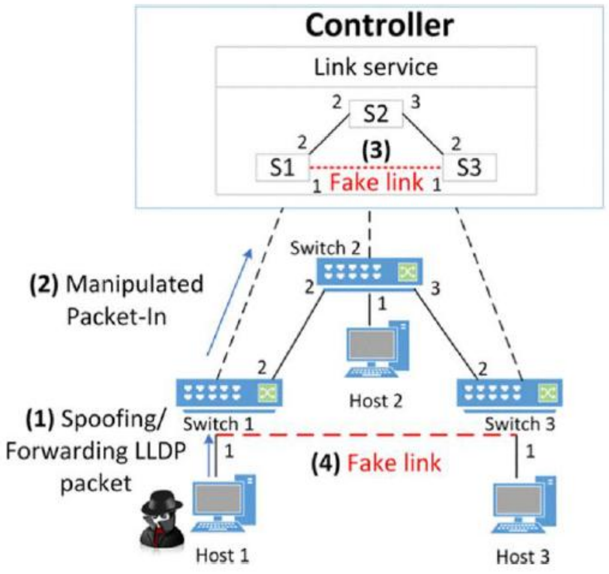


Figure 3.2 – LLDP Spoofing

Southbound interface

Separating the data plane from the control plane comes with many threats. The connection between the switches and the controller with secure TLS is optional for the new versions of OpenFlow. Also, TLS is not secure against TCP level attacks and is unreliable, so the following attacks can occur:

- Man-in-the-middle attack: If the attacker manages to control the channel then he can reshape the messages exchanged to corrupt the network.

- Eavesdropping: The attacker could either actively or passively steal valuable information by listening to the channel.

- TCP level attack: The TLS channel does not support security against attacks on the TCP protocol and could fall victim to such an attack.

- Service Denial Attacks (DoS/DDoS): The southern interface can be targeted by DDoS attack by overloading the communication channel.

<u>Data plane</u>

The controllers rely on the correct operation of the switches and routers. Then the biggest threats in the data field develop.

- False Flow Introduction: A major challenge is the possibility of identifying the controller as to whether the flow rules are authentic or not. An exposed transmitter could create false flow rules that would bring the data field inactive.
- Flow register entry limit on switches: Switches have a higher limit on the number of rules that can be entered. This could be the target of an attack that would deplete the positions on the board, effectively negating the service (DoS).
- Exposed hosts or switches: An exposed device can lead to a plethora of attacks. Except in the case of a traffic mission target boundary, the intruder may create dynamic attacks such as traffic retrieval, traffic hijacking, or service denial (DoS).

Recover after an error: After an error on a link or switch, the controller receives new information about the status of the network from the switches and recalculates the route. This requires a secure connection to the controller. If the switches cannot communicate with it then network recovery is very difficult.

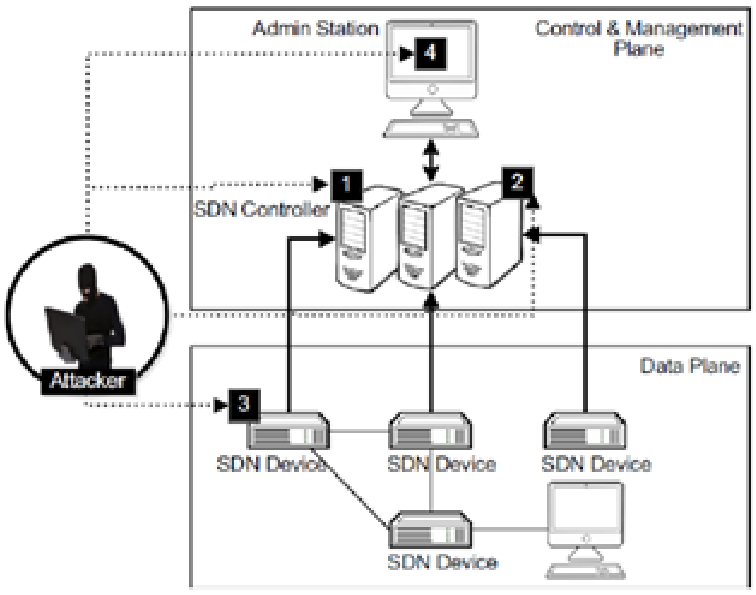


Figure 3.3 – Possible attacks on SDN

3.3.2 SDN Threats

In the above paragraph we have seen the security issues that occur in the architectural layers of SDN networks. We will now look at some of the major threats that may affect their operation.

- Denial of Service. The attacker can push the controller a large volume of traffic, which will constantly and randomly change the characteristics of the flows. Thus, the auditor receives new and unknown traffic flows which he does not know how to handle. The result is network outages and denial of service.

- Switch vulnerabilities. A malicious switch can flood the controller with malicious packets. The result will be a slowdown in network traffic or even the controller will not respond to new routing requests from other switches.

- Man-in-the-middle Attacks. If the TLS protocol for communication between the controller and the switch is not used, the attacker may exploit the vulnerabilities of the control level and launch attacks from there.

- Authentication Issues. In an SDN network, multiple nodes have access to one controller, thus increasing the possibility of unauthorized access and unlawful configuration.

- Open Interfaces. Since SDN supports open interfaces and known protocols to simplify networking, an attacker can take advantage of it and gain full control of the network.

3.3.3 SDN security research

Many research have been conducted on security in SDN networks in several security areas such as secure SDN design, satisfactory control, enforcement of security policy, increased security, security analysis or error tolerance. Here are some of them:

- A project for monitoring and analyzing SDN network is AVANT-GUARD [62]. It makes two important additions to the SDN architecture. First, an expansion of the data field, the transfer of connections that drastically reduces communication between the data field and control during a DoS attack. Second, triggers for collecting statistics. While it works well in the TCP protocol, it does not touch UDP or application level.

- FLOWGUARD [63] is a security structure that is responsible for detecting and dealing with political breaches in SDN protection walls. It works in real time.

- An important contribution in the field of security enhancement is the NICE [64]. It is an OpenFlow-based security structure for detecting vulnerabilities installed on virtual machines. These applications could be used to attack virtual machines in the cloud. It works on the basis of detailed models, programmable virtual transmitters and an interface for detecting and dealing with threats.

- FRESCO [65] is a security structure for OpenFlow networks based on exporting scripts to interfaces. This helps security experts develop a network monitoring logic as programming libraries. It is made in modules and its operation is done through NOX. Security is controlled by FortNox [66]. FortNox imposes a kernel that helps as an active defense against threats. It is a machine that prevents the contrast between the safety rules. This is done by prioritizing rules and detecting contrasts.

- Another effective effort is SnortFlow [67]. SnortFlow is a cloud computing system. It uses Snort and OpenFlow as a base.

3.3.4. SDN security mechanisms

Security mechanisms are intended to provide access to legitimate users, protect systems from attacks, and provide mitigation and countermeasures when attacks occur. Key control tasks can generally include detection, recording, protection, and countermeasures.

- Firewalls

The firewall is one of the most popular security mechanisms. They are responsible for monitoring network traffic and the rules of the firewall are those set to prevent or allow traffic based on IP addresses, ports, protocols and MAC addresses specified by users or network administrators.

The most important impact that SDN has on firewalls is that the SDN controller itself makes decisions about flows, unlike traditional networks, where this was the main role of the firewall. In SDN technology, the controller acts as a protection wall. Auditors constantly evaluate or know current topology using a link detection unit, as they transmit packets to ordinary switches and based on their response, the controller can often predict the current network topology. Also, SDN controllers have the ability to dynamically add rules to switching flow tables. This connection (between the firewall and the switches) does not exist in traditional networks. As a result, the rules of the firewall on traditional networks are static and unrelated to network traffic. It is therefore possible that some rules in traditional firewalls are outdated or inapplicable. On the other hand, the rules of the flow table in SDN are very dynamic and the outdated rules are eventually removed from the flow tables.

In SDN, a module of protection wall can usually be added as a northbound (REST) API to the controller. The REST API is an additional standard interface for interacting with most SDN controllers. Allows applications that developed from user to communicate with the controller. The rules of the firewall are different from the rules of the flow chart, although they may seem similar.

Much research has been done on how SDN-based firewalls can be implemented. Indicatively, FlowGuard, which we mentioned above, is an SDN-based monitoring framework for detecting possible conflicts between the rules and flows of the firewall. In addition, SDN-based firewalls for P2P networks are recommended. The use of P2P networks can benefit from SDN, because P2P networks have very dynamic users who often register and leave. The flexibility of SDN technology over traditional networks and its ability to dynamically tailor user requirements are appropriate for most P2P use cases.

- Hybrid firewalls

Hybrid protection walls are protective walls that operate in SDN technology environment along with traditional network technologies. An example is the FlowAdapter [69] application for managing streams on OpenFlow heterogeneous switches. Flow tables in OpenFlow converters should be able to match the material of traditional networks. In addition, there are certain types of fields in flow tables that do not correspond to traditional switches. In fact, the OpenFlow protocol itself is evolving as older versions had 12 properties and now the new versions have 40. There is always the need to support compatibility with older versions and at the same time the need to ensure that valuable information is not ignored because of this transformation.

- IDS / IPS

Intrusion Detection / Protection systems (IDS / IPS) stop or allow packets based on thorough packet research, data mining, template recognition, matching signatures with existing threat reserves, etc. Unlike traditional IDS, SDN IDS can use a huge amount of real-time information flow. SDN can change the way security mechanisms are distributed. For example, an IDS exists in a position in traditional networks (usually in network facilities). In SDN, IDS tasks can be distributed through network switches or factors, with the controller or one of its units being responsible for the process.

One of the best-known IDS is called Snort. Snort's integration into SDN faces many challenges. The SDN controller typically receives samples and not full flows as is the case with Snort mode. One way to balance this situation is for the controller to take the first packet or the first few packets of a given flow. Once it receives these, the controller installs rules on the switches that will handle the rest of the packets in this flow. But because the Snort system recognizes each packet as flowing, it would not be useful to be placed inside the controller as it will significantly affect the performance and structure of the OpenFlow network. The solution to this problem would be to create a service in the controller to manage a set of machines that run Snort and install rules that redirect traffic to the machines that run Snort.

Snort has its own limitations when it comes to the type of attacks it can detect. While it's a good open source IPS / IDS code (with a rule-based language that combines signature control, protocol, and specialty detection), Snort still relies on regular signature updates. It has no way of detecting higher-level exploits such as web exploits (eg malicious Java Scripts).

SDN and Snort also differ in how they are collected, retrofitted, and tracked. Traditional networks use network expansion ports to redirect traffic to monitoring or security applications. In SDN, data can be extracted from the controller via northbound APIs. Filters can also be applied, which will drive traffic based on specific criteria and the command controller will rewrite the traffic according to these criteria.

- Monitoring and Control

Monitoring and control are very important tools for many security checks. An important feature of SDNs is related to all the details that can be gathered in the flow and even in the level of the packages, which was difficult to achieve in traditional IP networks.

3.4 DoS/DDoS attack on SDN

Attacks of denial of service have become a real threat to the networks, the digital world and the network infrastructure. The reasons for such an attack can be economic, political or just a disturbance. They can paralyze the system and its services by overloading the servers, switches and connectors with malicious traffic. They bring a reduction in the quality of the service or a universal denial of it. The growing dependence on information technology infrastructure leads to the need to safely deal with these attacks. Attacks of denial of service are carried out by a computer and a network and are intended to consume the resources of the network so that they become inaccessible to the users of this network. There are many solutions for detecting service denial attacks and they are categorized according to the method. Also, each infrastructure network has a number of applications that have a different level of tolerance based on their purpose and scope e.g. organization, smart city.

3.4.1 DDoS atack

The purpose of a DDoS is to use various resources that are distributed across multiple areas against a single target and to disable its services based on packet flooding. Usually bot networks are used for successful attacks that are scattered around the world. This is the main difference with DoS attacks. There are 3 categories of DDoS attack based on the method used:

- Reflection-based flood attacks:

  a. Smurf Attack: In this attack a large number of ICMP packets with the IP address of the target computer as a source are retransmitted to the network using a broadcast IP. So, a huge number of malicious packages are causing congestion on the network.

  b. Fraggle attack: The fraggle attack is similar to smurf but uses UDP motion instead of ICMP.

- Flood attacks based on protocol exploitation.

  a. Flood TCP SYN (SYN Flooding): In a TCP SYN attack the attacker sends the message that starts the three-way start in the TCP protocol with the SYN field completed. The victim responds with SYN ACK flags but the attacker does not respond to the completion of the tripartite transaction usually because the IP of the source of the first message is false. So the queues of the victim for the TCP protocol are filled and it is impossible to create new connections.

  b. UDP Fragmentation: In this case the attacker sends large packets of UDP, over 1500 bytes each to consume more bandwidth with fewer packets. The victim's resources are being used to reunite the fake packages.

- Reflection and amplification based:

  a. DNS-enhanced flooding: In such an attack, many malicious bots generate DNS questions with a fake IP source that creates a lot of traffic to the victim.

b. NTP Enhanced Flood: Similar to DNS Enhancement Attack but uses NTP (Network Time Protocol) servers instead of DNS.

3.4.2 DDoS threats based on SDN architecture

- DoS attack on the flow chart. OpenFlow rule design makes the converter vulnerable to service denial attacks (DoS). Since packets with an unknown destination address will cause a new rule to be entered in the switch, an intruder can create large amounts of packets for unknown network computers in a long time, thus quickly filling the limited storage capacity of the panel. . When the flow chart is saturated with illegal traffic, legal traffic will not be properly promoted, as there is no longer any capacity available to introduce new rules.

- DoS attack on the buffer cache. As described above, the packages are stored in the temporary buffer memory waiting for the search results or the introduction of a new rule before they are forwarded. The buffer packages are marked for deletion based on the First In First Out (FIFO) process to free up storage space. As in the case of the flow chart, the storage capacity of the buffer memory is also limited. Attackers can flood large ports on the switch that belongs to a different flow than the one normally encountered by the switch. These large packages are then transferred to the buffer and this leads to saturation and malfunction. Once the legal packages are received, the buffer will not have enough space to store these packages and these new packages will have to be dropped.

- DoS / Distributed DoS (DDoS) attacks the controller. DoS / DDoS attacks attempt to make the controller functions inaccessible to legitimize users by depleting computer or memory resources, as shown in below Figure.

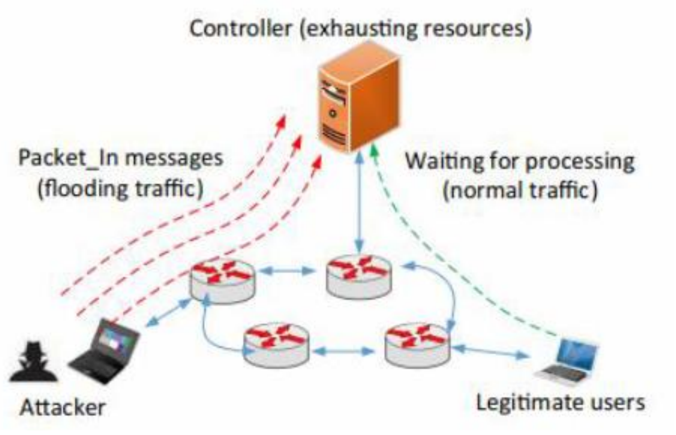

Controller (exhausting resources)
Packet_In messages (flooding traffic)
Waiting for processing (normal traffic)
Attacker
Legitimate users

Figure 3.4 – DoS/DDoS attack on SDN controller

An intruder could generate huge flood traffic in a short period of time on a network with SDN capability using its own server or by controlling other distributed 32zombie servers. This circulation is mixed with normal and it is difficult to distinguish between the two types. According to OpenFlow, if a converter does not know how to handle a new packet, it will first store that packet in its buffer and then send a message to the controller asking for instructions. Therefore, in the event of a DoS attack, the controller will have to deal with a huge amount of such messages generated by flood traffic in a short period of time, which can lead to depletion of resources to process normal traffic. At the same time, the bandwidth between the controller and the switches can be completely occupied by the aggressive traffic and this significantly reduces the performance of the entire network.

3.4.3 DDoS Detection in SDN

SDNs have a lot of data that can help with DDoS attacks. These elements are the separation of data field with control field, the central controller, the ability to program the network, the analysis of network statistics and the possibility of dynamic renewal of flow rules. Detection is categorized according to the method below:

- Entropy: Entropy is used to measure the randomness of a variable. Entropy techniques have been developed to measure a set of data. High entropy value gives a distribution with equal probabilities while low indicates a concentration of values in the distribution. For this reason, similar techniques are used to detect malfunctions in IDS (Intrusion Detection Systems). Entropy can be measured in several fields such as flows, IP addresses, number of packets or ports. Due to the success of traditional networks, they have been successfully transferred to SDN.

- Machine learning: Mechanical learning has been applied to several traditional IDS. Artificial neural networks, Bayes networks, Self-Organizing Map, fuzzy logic have been widely used. This type of technique differentiates flows and categorizes them as malicious or benign.

- Traffic Pattern Analysis: This technique is used in the case that infected computers show similar behavioral patterns and are different from those of goodwill.

- Connection Rate: The ratio of successful connections and their number are two possible types of measurement for this category. The ratio as the first case takes into account that the probability of successful connection of good-natured hosts is higher than that of malicious ones. It is used for the TCP protocol and counts the connections that the tripartite greeting does not complete. In the second case, it takes into account that malicious hosts will have much more connection effort than goodwill. It uses a time limit to detect them.

- SNORT: Snort is a widespread IDS and open source intrusion protection system (IPS). Many security structures have been built with SNORT as a base.

4. Implementation of an Industry 4.0 – Ready Enterprise Network

4.1 Introduction

In Industry 4.0 the basic idea behind enterprise networks is digitalization and softwarization. In this context, cloudification and virtualization of the resources is critical since it enables resusability and efficient service management. Containers and virtual machines are the main enablers towards this direction.

In industry, research has been conducted to test the capabilities of a software containers-based system in the world of real-time industrial controllers (such as PLC and SCADA systems). The aim was to study the main challenges observed in conventional systems, namely the simulation of the old type and the development of flexible functions, proving that it is finally possible to introduce such a system in industry. In addition, it is considered that there is a benefit to using some technology for Containers such as Docker in a factory environment, supporting the development and control of control software on distributed computing hardware, as well as offering more flexibility to factories by providing rapid changes to control software strengthening the trend for mass customization in the industry.

Software containers are an interesting alternative to virtual machines for better performance and implementation of systems in the "cloud" and beyond. Although many Virtual Private Server service providers had originally adopted software containers in the past, they eventually used virtual machines to achieve a consistency in performance. Needs for managing the consumption of resources, however, led to their reuse. The Linux operating system, with the use of the kernel where the operation of the containers is based, but also the zero cost was the basis for the expansion and reuse of the software containers. The transformation of a computer system into smaller independent virtual environments (instances) so that the system administrator of a company-organization can create and manage them according to his needs, as well as the ability on a computer to run different instances from different operating systems have led in recent years to the gradual introduction of the use of virtualization technologies. The technologies are divided into 2 categories, Container – based – Virtualization and Hypervisor – based – Virtualization with the former providing a lighter and more efficient virtual environment, as using the server kernel it executes multiple and different virtual environments, called software containers.

The general architecture of software Containers compared to virtual machines is shown in the graph below.

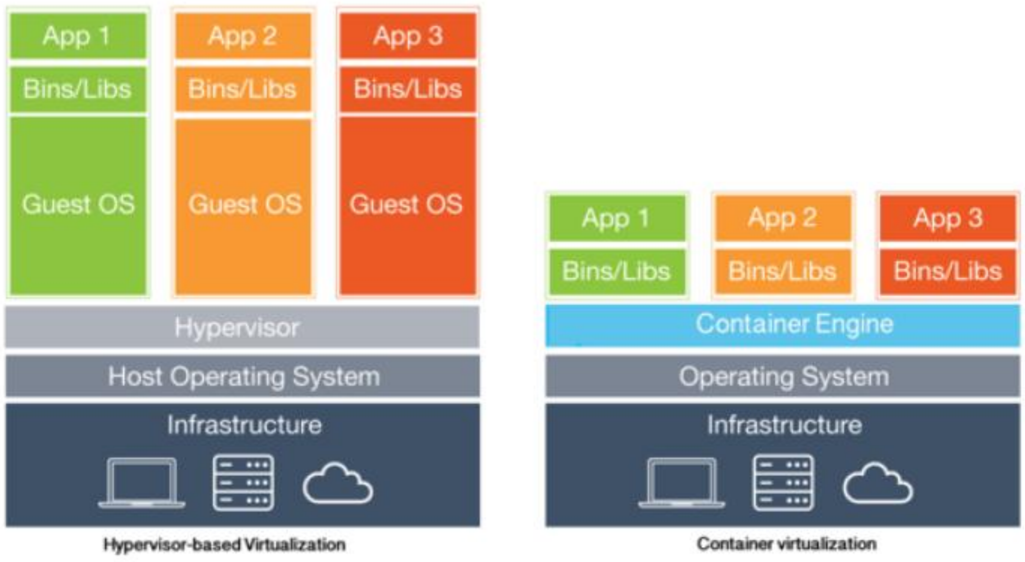


Figure 4.1 – Hypervisor & Container virtualization architecture

In both technologies there is an intermediate level that manages the system resources required to run applications. In the case of hardware virtualization, this layer is called the hypervisor and its role is to share the resources (RAM, processor, Disk) that each virtual machine requires to operate the operating system installed on it. Each virtual machine is an independent system with a virtual hardware, and the necessary libraries for its operation. In the second case the layer is called a kernel and manages the needs of an application in software (libraries) to run properly. More specifically, the term Container – based – Virtualization or Application Containerization defines the operating system-level virtualization method for the use and execution of distributed applications. Without using virtual machines, for each application but by running them on a central system using the kernel. The main difference between the above two categories is that in Container – based – Virtualization a container provides virtualization at the software level and not hardware, it is also a standalone environment with resources that can support the execution of applications without using a virtual machine installed on the existing system (hostmachine) which is the basic idea of the operation of hypervisor-based – virtualization technology. Containers can operate with the least amount of resources (software libraries, RAM, processor) to perform the process for which they are designed so that the technology of software containers allows 10 times application of virtual environments on a physical server against the hypervisor, without, however, offering solutions to security problems in recent years.

The new demands of emerging technologies are the driving force behind the development of information technology. The Internet of Things is an ever-evolving industry that requires more efficient ways of managing data transmission and processing. Two of the approaches that can meet the requirements of an ever-increasing number of connected devices are Cloud computing and Fog computing.

Cloud computing and IoT are prime examples of next-generation services and applications. Furthermore, Fog computing technology is expanding the Cloud to the edge of the network, giving flexibility to both the devices and the network itself. Its basic idea is to bring network resources such as memory and processing power close to the devices that produce the data.

A complete Enterprise Cloud integrates all the data volume created within the company. However, the corresponding transformation of the IT infrastructure can take years, as it affects many aspects of a company's organization, existing roles, processes, policies and services. The vast proliferation of mobile devices and content, the virtualization of servers, and the advent of cloud services are among the trends that are driving the networking industry to rethink traditional network architectures.

SDN is very important for network organisation in Industry 4 because it enables separation of control from data flows. Hence it is inline with the other principles of softwarization and cloudification of the production. The controller orchestrates the network sessions between containerised services and end users or end devices. A firewall at the entrance of an enterprise network monitors the data traffic and enables rules enforcement at flow level. IDS/IPS enables the application of rules concerning protection of the infrastructure from malicious intruders and orchestrates the cybersecurity protection of the overall ecosystem from attackers.

When a few years ago, everyone started talking about SDN, they thought it was one thing: separating network control from handling data packets moving through it. Traditional networks had already begun to adopt this philosophy, with the addition of controllers within switches and the use of various data center infrastructure technologies. SDNs took advantage of this idea and went even further by eliminating the need for the controller and package manager to be inside the device itself or even from the same manufacturer.

SDN, then, is one level above the hardware - but not dependent on it, that is, from the manufacturer - offering advanced management and monitoring capabilities through a unique cloud console. With SDNs, cloud computing takes on a real dimension, giving the administrator unlimited management and intervention capabilities where needed, without having to deal with issues such as how switches and various other network devices are connected within the network. With an SDN, the deploy of settings, devices, policies, etc., is done quickly without the need for the physical presence of the administrator. This on the one hand means ease of management and on the other hand a reduction of resources on the part of the company.

The key issues facing SDNs are security, scalability and flexibility. The result is a modern infrastructure that can offer new applications and services in minutes, instead of days or even weeks required in the past. In addition, this approach enables partners offering management services to tailor support contracts to the needs of their clients. With a SDN, the outsourcing of network administration becomes a reality.

In our work we have implemented all the above functionality over a virtualised infrastructure so we prove the viability of such an enterprise network organization. In particular we have used four vm over one pc. The vm 1 includes our firewall/router pfSense, the vm 2 includes Mininet which helps us to make our SDN topology, the vm 3 includes the SDN Controller and the vm 4 includes a Xubuntu Desktop operation system which run the graphic interfaces of Controller and services like sFlow-RT and Wireshark.

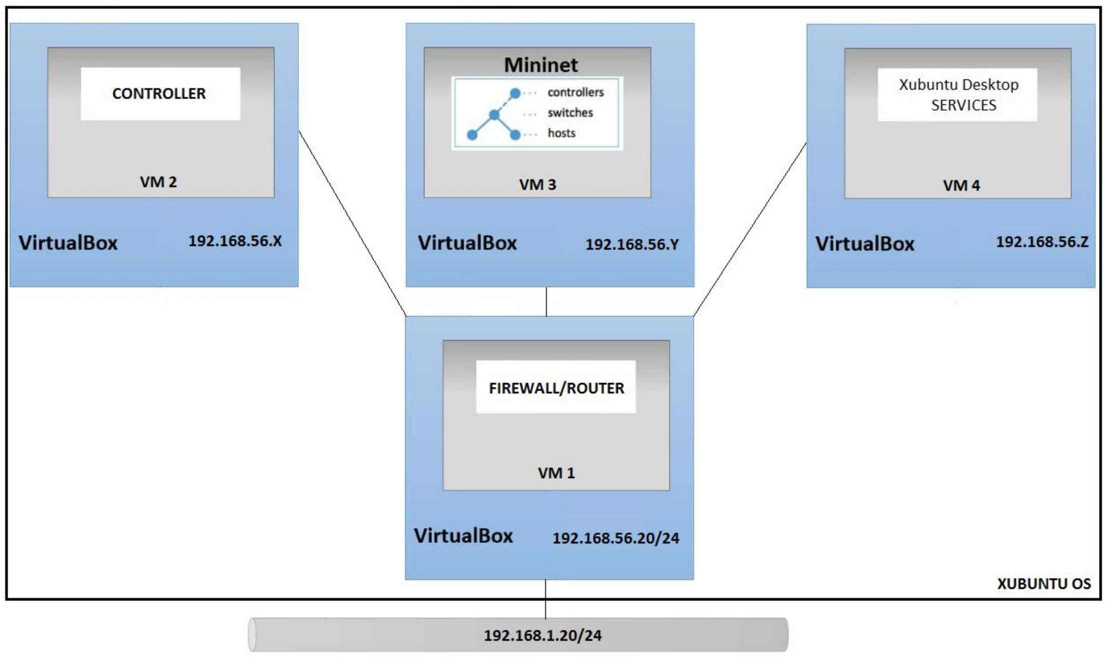


Figure 4.3 - Implementation

As well as with those mentioned in the previous chapters in which we have analyzed the most important parts that make up SDN networks, but also the most important issue that concerns them and will continue to concern them which is none other than security that also mentioned to paper [73], we decided to conduct an experimental implementation on the security of SDN networks. To this end, in this chapter we will try to analyze the general structure of the experimental part of this dissertation, as well as the individual tools we used to implement it.

So for this purpose we used our fixed sublink which was connected to the internet and also through the VirtualBox tool through which we create virtual machines we implemented a Firewall as well as the infrastructure of the SDN network.

First of all, we will introduce our Firewall which will handle all the incoming and outgoing traffic between our SDN infrastructure and real network. Then we will introduce our SDN infrastructure with the major part of this is controller which is responsible for network managing that we build through Mininet. Mininet is not a real part of our SDN infrastructure but is a network emulation tool that will facilitate the emulation of Software-Defined Network topologies, so it plays a very important role in the SDN infrastructure. After the above also we will talk for other tools that we used for Detect and Prevent Cyber-attacks such software tool Snort which help us via DPI technology to implement our IDS/IPS system.

4.2 Software tools for the implementation of a modern enterprise network

4.2.1 Firewall

In computer science, the term firewall is used to denote a device or program that is configured to allow or reject data packets that pass from one computer network to another.

The main function of a firewall is to configure data traffic between two computer networks. Usually these two networks are the Internet and the local / corporate network.

The purpose of installing a firewall is to prevent attacks on the local network and deal with them.

pfSense

pfSense [53][48] is an open source customized firewall/router computer software distribution based on FreeBSD. We can install it on a physical computer or a virtual machine to make a dedicated firewall/router for our network. It can entirely managed through a web-based interface. In addition to the robust, flexible firewall and routing platform, it includes a long list of features and a packet system that allows for further scalability without adding irritating and potential security vulnerabilities to the core distribution.

4.2.2 SDN infrastructure

4.2.2.1 Controller

SDN controllers are the network brains in SDN. They are essentially controlling the network by managing the flow control on "southbound" switches / routers via southbound APIs and on applications and business logic "northbound" via northbound APIs in order to develop new improved networks. Due to the large development of SDN in recent years, auditors have been making joint applications, such as OpenFlow.

OpenDaylight

OpenDaylight (ODL) [45] is an open source controller based in Java which was developed and driven by the Linux Foundation. OpenDaylight was designed with a modular framework that allows developers versatility to add any applications via the northbound API. ODL and other standard Internet Engineering Task Force (IETF) protocols support OpenFlow, as well as southbound communication. Additionally, vendors may add their own protocols to ODL as new elements for interaction between the north and southbound. Given this, ODL should only be running the OpenFlow protocol at the east and westbound connection.

LF Networking (LFN) [46], which promotes collaboration and operational excellence through networking initiatives, today announced OpenDaylight (ODL), the most omnipresent open source SDN device.

The first networking project of the Linux Foundation and now part of LFN, OpenDaylight was created in 2013 with the release Hydrogen 0 as an open source platform to speed up adoption, foster innovation and establish a more open and transparent approach to SDN. Today with 10th release Neon, ODL has become the most omnipresent open source SDN controller that helps power over 1B global subscribers to the network 0. In our thesis we will focus on sixth release "Carbon" as it was the most stable at our testbed.

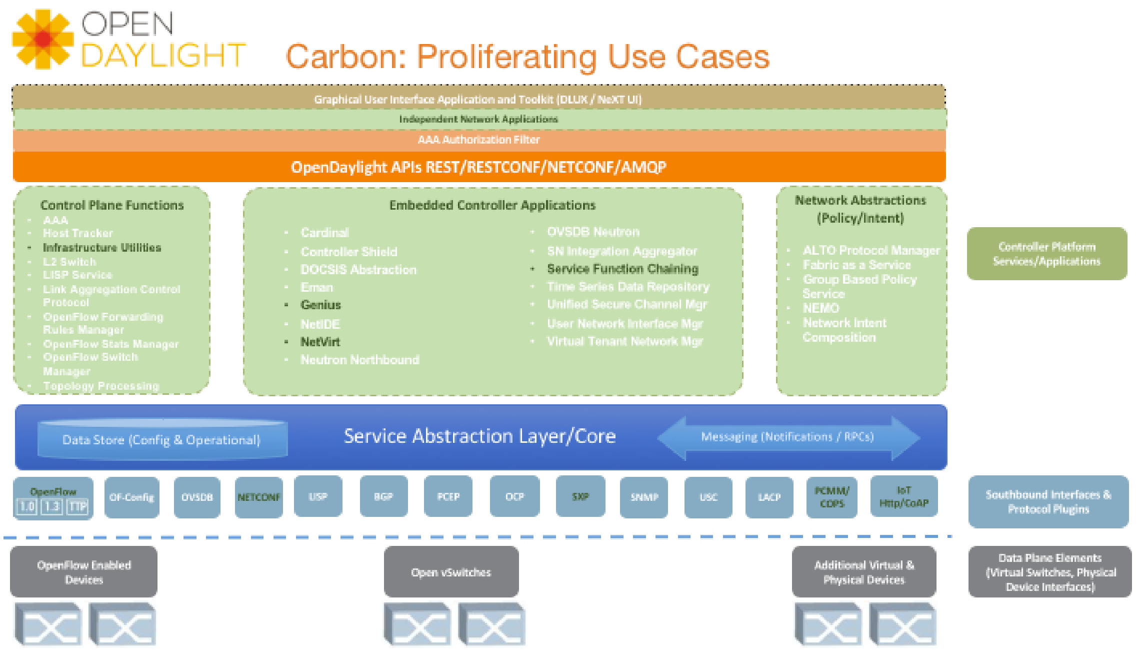

OPEN DAYLIGHT
Carbon: Proliferating Use Cases
Graphical User Interface Application and Toolkit (DLUX / NeXT UI)
Independent Network Applications
AAA Authorization Filter
OpenDaylight APIs REST/RESTCONF/NETCONF/AMQP
Control Plane Functions
AAA
Host Tracker
Infrastructure Utilities
L2 Switch
LISP Service
Link Aggregation Control Protocol
OpenFlow Forwarding Rules Manager
OpenFlow Stats Manager
OpenFlow Switch Manager
Topology Processing
Embedded Controller Applications
Cardinal
Controller Shield
DOCSIS Abstraction
Eman
Genius
NetIDE
NetVirt
Neutron Northbound
OVSDB Neutron
SN Integration Aggregator
Service Function Chaining
Time Series Data Repository
Unified Secure Channel Mgr
User Network Interface Mgr
Virtual Tenant Network Mgr
Network Abstractions (Policy/Intent)
ALTO Protocol Manager
Fabric as a Service
Group Based Policy Service
NEMO
Network Intent Composition
Controller Platform Services/Applications
Data Store (Config & Operational)
Service Abstraction Layer/Core
Messaging (Notifications / RPCs)
OpenFlow
1.0
1.3
TTP
OF-Config
OVSDB
NETCONF
LISP
BGP
PCEP
OCP
SXP
SNMP
USC
LACP
PCMM/COPS
IoT Http/CoAP
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Figure 4.4 - OpenDaylight Carbon

Ryu Controller

The word Ryu in Japanese (竜) refers to a dragon and a school of thought and means "flow" [49]. With a dragon in its logo, Ryu 0 is commonly referred to as an open source software-based segment, defined by a network framework (SDN). As already mentioned, SDN Controller is the brain of this type of networking, which communicates a wealth of information to the switches and routers of the network's main network, via Southbound

APIs, and to applications and business logic in the network's overhead section, via Northbound APIs. Ryu is supported by NTT (Nipon Telegraph & Telephon Co) 0 and is implemented in Data Centers of this company.

Ryu provides software elements, with well-defined application programming interfaces, that make it easy to create new web applications, control and management by network developers. This software-based approach helps Ryu personalize to meet specific needs, enabling developers too easily and quickly modify existing software or implement their own to ensure that the underlying network is able to respond to changes in demand from applications.

Ryu is written entirely in the high-end, general-purpose Python programming language, and all codes are available and open to anyone from the licensed Apache 2.0 version. It supports multiple protocols for device management such as NETCONF (Network Configuration Protocol) and OF-config (OpenFlow Management and Configuration Protocol), while the latter is noted to be one of the first and most widely used communication standards in networks. It even supports Nicira extensions 0.

Like any SDN network controller, Ryu can create and send OpenFlow messages, listen to asynchronous events such as the flow of a stream, analyze and manage incoming packets. The following Figure shows the architecture of the frame of a standard Ryu controller.

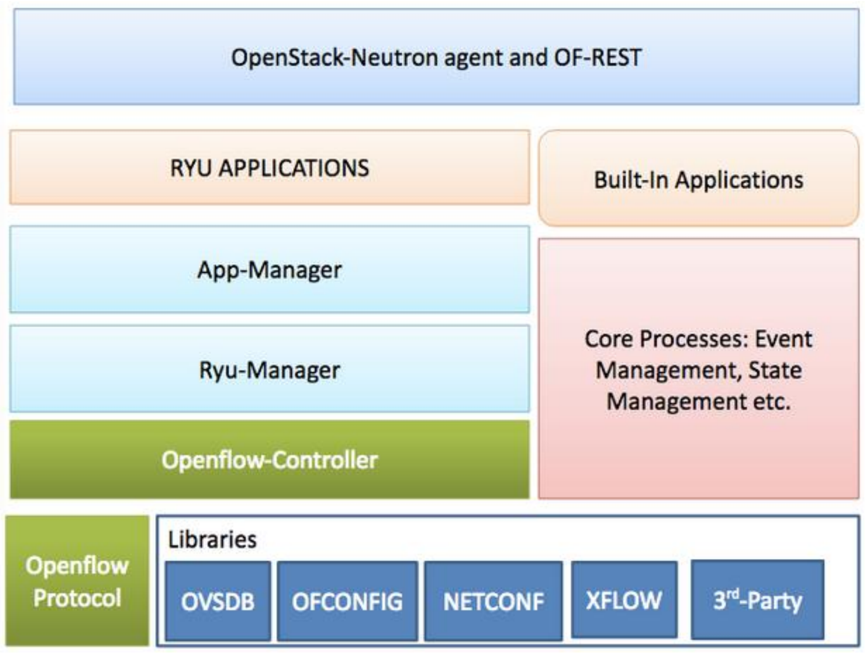


Figure 4.5 - Ryu Architecture

The main elements of Ryu's architecture are:

- Ryu Libraries
- OpenFlow Protocol and Controller
- Managers and Core-Processes
- RYU Northbound
- RYU Applications

4.2.2.2 Mininet

Mininet is a program which simulates network or, even, a network orchestration system. With this tool we can manage end-to-end servers, switches (OpenvSwitch (OvS)) 0, routers, and links on a single Linux kernel. With the usage of a lightweight virtualization transform a simple system into a full network, running the same kernel, system and user code. It is designed to create SDN networks easily, using an OpenFlow controller, several switches OpenFlow connected on an Ethernet network and multiple hosts connected to them. It also has features to support different types of switches and controllers, and Python API to create more complex scenarios. A host at Mininet acts like a real machine, we can ssh in it and execute random programs. If we execute these programs, they will send packets over a real Ethernet network, with the speed and delay of a given connection. The packets process what appears to be real Ethernet switch or router, with a certain degree of anticipation when two programs, such as a server and an iperf client, communicate via the Mininet, the measured output shall be the same as for the two manual machines.

The real thing is Mininet virtual hosts, switches, connections, and controllers, they are generated utilizing software instead of hardware alone and their behavior is identical to discrete hardware components for the most part. A Mininet network that look like a hardware network or a hardware network that look like a Mininet network and executes the same binary code and applications on either platform can generally be built. Moreover, Mininet networks running real code of Linux. Thanks to this feature, the code developed Mininet (for a controller, a switch or host) it is easily reproducible in a real environment by making minimal changes. Therefore, a design that works correctly in Mininet can be used almost directly to deploy, test and evaluate the performance of a real SDN network. In short, Mininet is an instant virtual network. It creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds, with a single command.

Its website [42] offers a great deal of information, from how to install using four different options to an extensive tutorial to familiarize our self with the Mininet environment in a simple way.

Why use Mininet [44]

- It takes only a few seconds to quickly start a simple network.
- We can build customized topologies.

- We can run real programs, everything running on Linux is available for us to run.
- We can customize packet forwarding: using the OpenFlow protocol, the Mininet switches are programmable.
- We may operate Mininet on our desktop computer, a VM, a native Linux box (Mininet is included with Ubuntu), or in the cloud (Amazon EC2 for example).
- We can distribute and repeat results. Once we have packed it up, anyone with a computer can run our code.
- It's easy to use, we can create and run on Mininet examples by creating Python scripts.
- Such as Mininet is open source, we can change the source code on https://github.com/mininet.
- If Mininet fails, or doesn't function for other reasons, we can address to the Mininet consumer and developer community via mininet-discuss and they will try to explain, repair, or support us to fix it. We can send also patches If we find bugs.

The other thing we must remember for network-limited tests is that we will possibly need to use slower connections. We must ensure that we will be carefully restricting the CPU bandwidth of our Mininet hosts. If we're mainly concerned with functional correctness, we can execute Mininet with no special restrictions on bandwidth, this is the fast and easy way to execute Mininet and it also delivers the highest output under load, at the cost of timing accuracy.

4.2.3 sFlow-RT

sFlow-RT [54] is an extensively used tool that offers real-time monitoring ability into Software Defined Networks. It sits in the control plane of the SDN stack. It changes the datagram received into summary statistics or actionable metrics on the flows as specified by the user. SFlow-RT collects a continuous stream of telemetry from standard sFlow agents embedded into network devices, hosts, and applications and transforms them to active metrics accessible via the REST API.

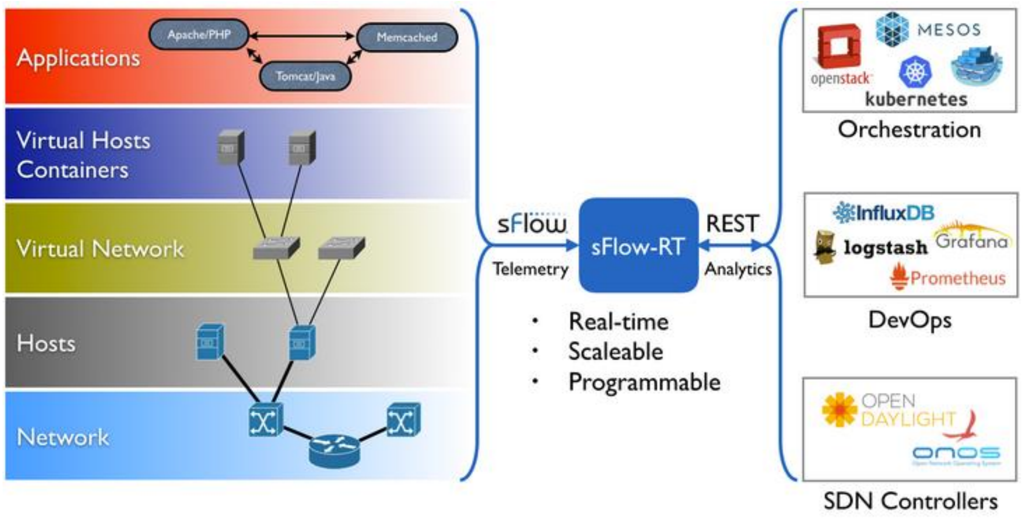


Figure 4.6 - sFlow-RT

Measurements can be easily integrated into a wide range of Internet and Cloud tools, Orchestration, DevOps and Software Defined Networking (SDN).

4.2.4 Wireshark

Wireshark [55] is a free and open source network protocol analyzer. It is used for network analysis, network monitoring, network troubleshooting, communications protocol development, education and training. The original name of the program was Ethereal, and in May 2006 it was changed to Wireshark for trademark reasons.

It is available for all major operating systems.

4.2.5 Snort

Snort [56] is a “lightweight” Network Intrusion Detection System (NIDS), as well as a packet analysis tool based on the libpcap library. The creator of Snort is Martin Roesch [57], who started developing the code in C programming language, and today a large number of people are involved in this process, with the aim of adding new functions to Snort and improving its capabilities. An important role in this is that Snort is Open Source software, available under the GNU General Public License (GPL) that makes free use and development of its code by anyone. The term "lightweight" has two meanings. One has to do with its application in relatively small networks, while the other has to do with its small size and ease of application and use.

Although Snort is known as a NIDS, in fact it is not the only one. Snort is characterized by four modes of operation, which are usually executed in combination with each other. The modes in which Snort is executed are defined by setting user parameters when running the program. Snort's four operating modes are:

- Sniffer: Read packets and display them on the console.
- Packet Logger: Save the packets it reads in future test files.
- NIDS: Monitoring the network and analyzing traffic based on user-defined rules and performing specific actions when a specific action is detected.
- NIPS (Network Intrusion Prevention System): Similar to NIDS with the ability to reject packets.

Snort is an example of signature-based NIDS software as a detection technique. Therefore, Snort determines each dangerous package by comparing the package with rules that have been defined by the administrator. Each Snort rule has two parts, the header and the option as we can see in Figure 4.6.

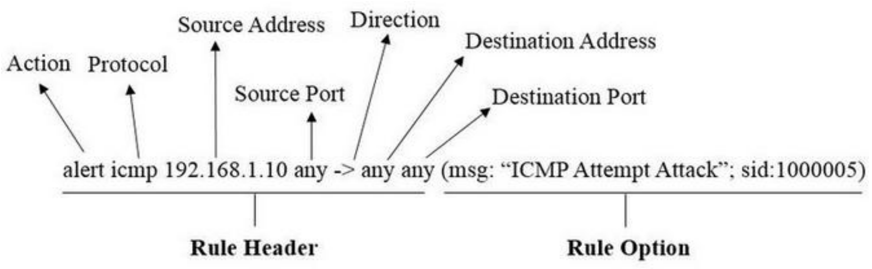


Figure 4.7 – Snort Rule Format

5. Attempting cyberattacks over a production-based enterprise SDN network

In this chapter we will implement an attack scenario in our SDN infrastructure and specifically we will pretend that an invader managed to break into our internal network by setting aside our Protection Wall. After he managed to break into our internal network, he managed to hack into the computer of one of our employees, where he is trying to attack an HTTP server.

To simulate the above scenario in VirtualBox we create these four VM's: pfSense, Mininet and sFlow-RT, OpenDaylight controller, and Xubuntu Desktop to run the graphic interfaces of ODL, sFlow-RT and Wireshark.

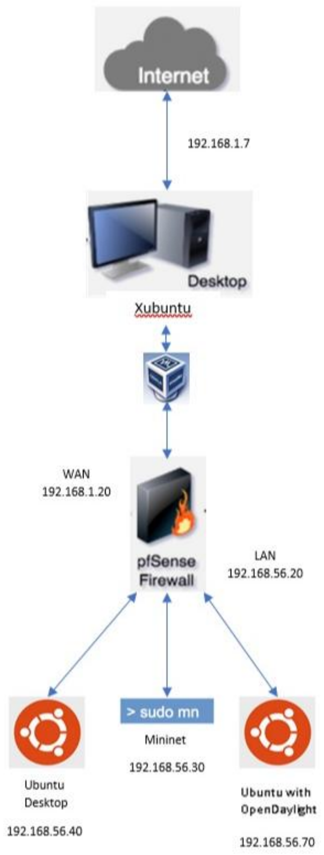
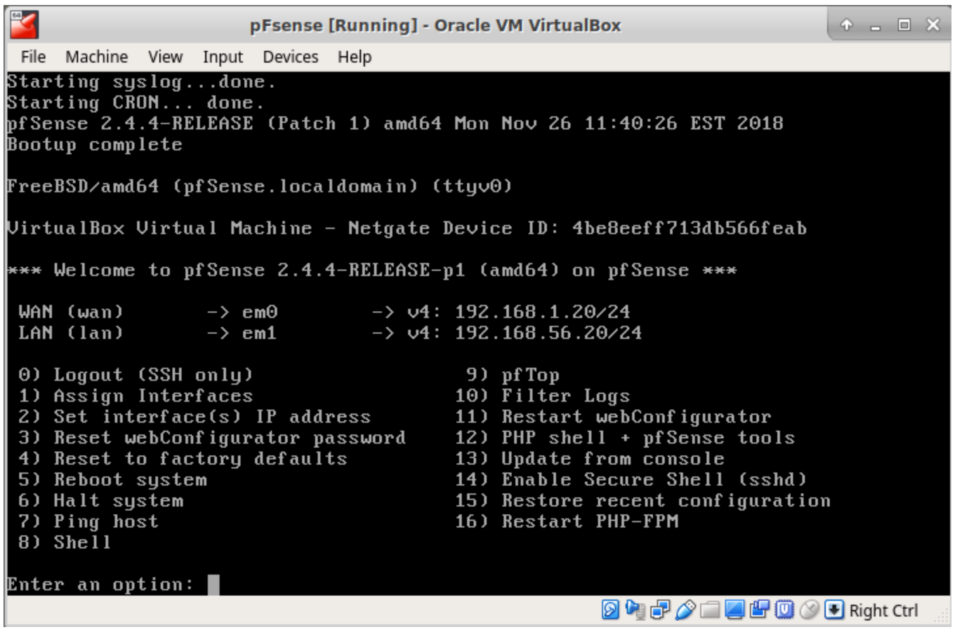


Figure 5.1 – Topology

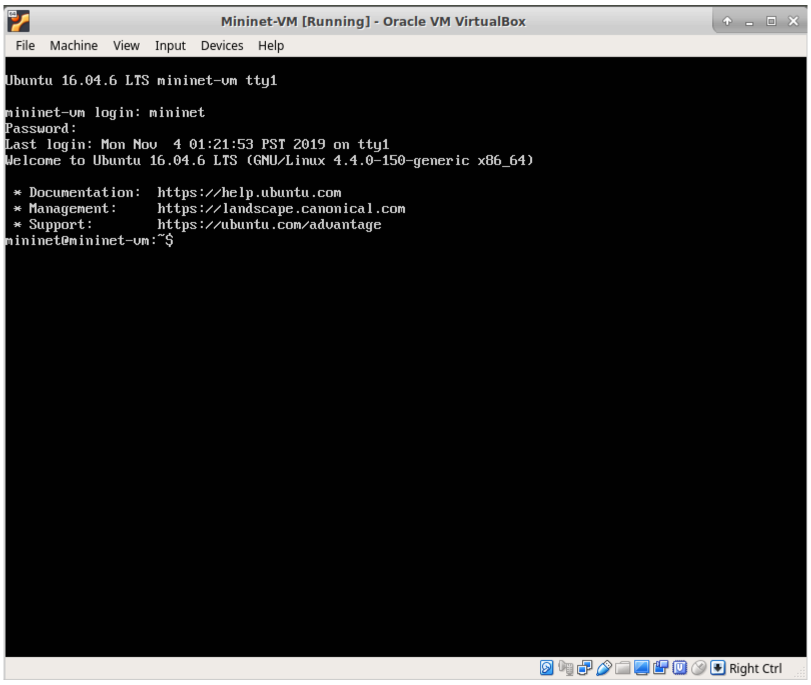
The first VM that we use for this implementation was pfSense, a firewall/router as we can see in Figure 5.1 and we use it to control WAN and LAN traffic.


pFsense [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Starting syslog...done.
Starting CRON... done.
pfSense 2.4.4-RELEASE (Patch 1) amd64 Mon Nov 26 11:40:26 EST 2018
Bootup complete

FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

VirtualBox Virtual Machine - Netgate Device ID: 4be8eeff713db566feab

*** Welcome to pfSense 2.4.4-RELEASE-p1 (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4: 192.168.1.20/24
 LAN (lan)       -> em1        -> v4: 192.168.56.20/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
 4) Reset to factory defaults         13) Update from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell

Enter an option:
```

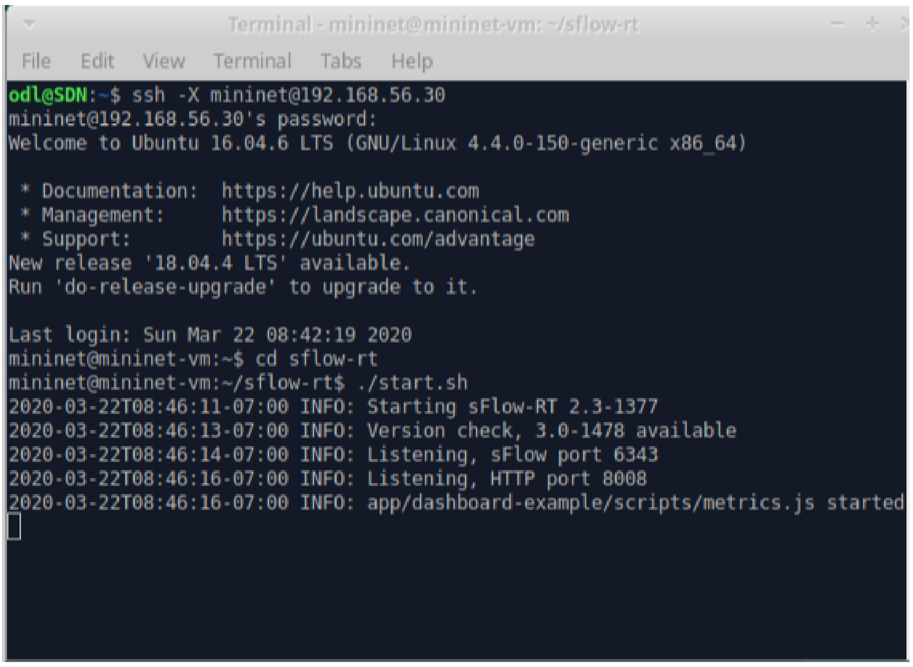
Figure 5.2 – pfSense

The second VM was Mininet that was used to create our topology and also in this VM we run sFlow-RT a tool for real-time monitoring.


Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Ubuntu 16.04.6 LTS mininet-vm tty1

mininet-vm login: mininet
Password:
Last login: Mon Nov  4 01:21:53 PST 2019 on tty1
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-150-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
mininet@mininet-vm:~$
```

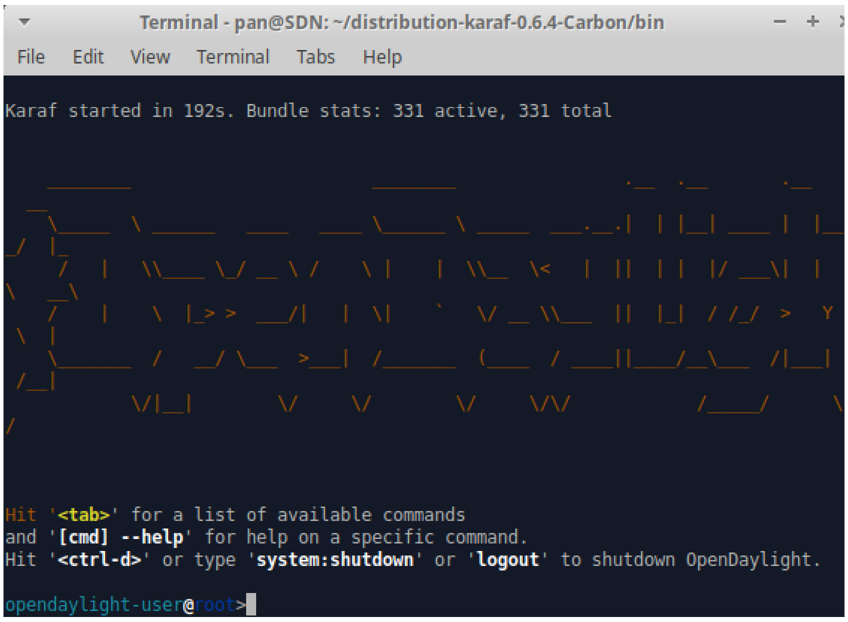
Figure 5.3 – Mininet


Terminal - mininet@mininet-vm: ~/sflow-rt
File Edit View Terminal Tabs Help

```
odl@SDN:~$ ssh -X mininet@192.168.56.30
mininet@192.168.56.30's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-150-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
New release '18.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Mar 22 08:42:19 2020
mininet@mininet-vm:~$ cd sflow-rt
mininet@mininet-vm:~/sflow-rt$ ./start.sh
2020-03-22T08:46:11-07:00 INFO: Starting sFlow-RT 2.3-1377
2020-03-22T08:46:13-07:00 INFO: Version check, 3.0-1478 available
2020-03-22T08:46:14-07:00 INFO: Listening, sFlow port 6343
2020-03-22T08:46:16-07:00 INFO: Listening, HTTP port 8008
2020-03-22T08:46:16-07:00 INFO: app/dashboard-example/scripts/metrics.js started
```

Figure 5.4 – sFlow-RT

In the third VM we install our controller OpenDayLight.


Terminal - pan@SDN: ~/distribution-karaf-0.6.4-Carbon/bin
File Edit View Terminal Tabs Help

```
Karaf started in 192s. Bundle stats: 331 active, 331 total


    ________                       ________                .__  .__       .__
  __
    \_____  \ ______   ____   ____ \______ \ _____  ___.__.|  | |__| ____ |  |__
_/  |_
     /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |
\   __\
    /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y
 \  |
    \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|
 /__|
            \/|__|        \/     \/        \/     \/\/            /_____/      \
/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

Figure 5.5 – OpenDayLight

And at our last fourth VM we install Xubuntu Desktop to monitor ODL, sFlow-RT and Wireshark.

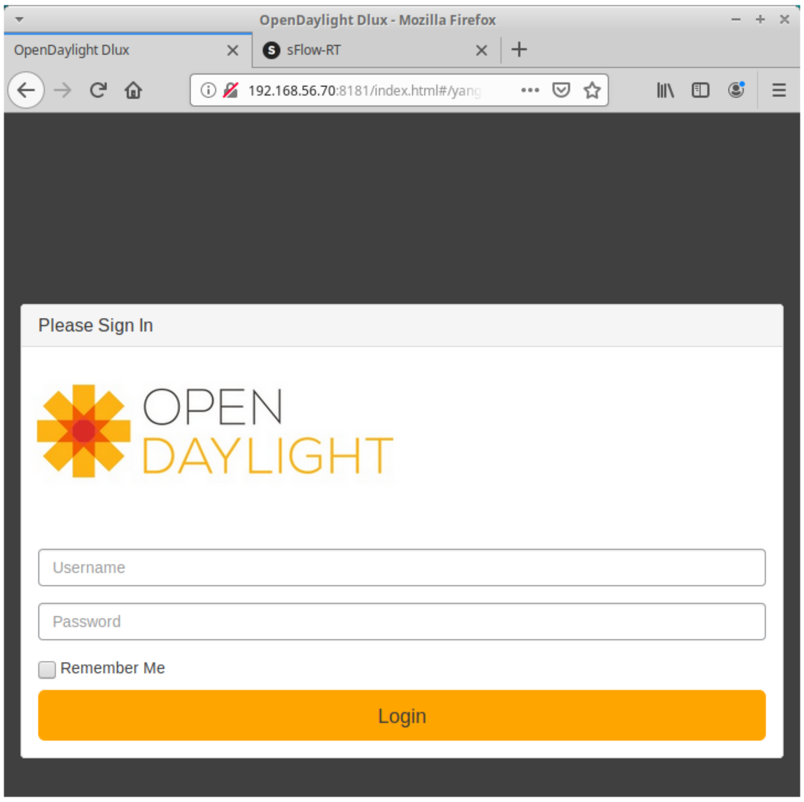
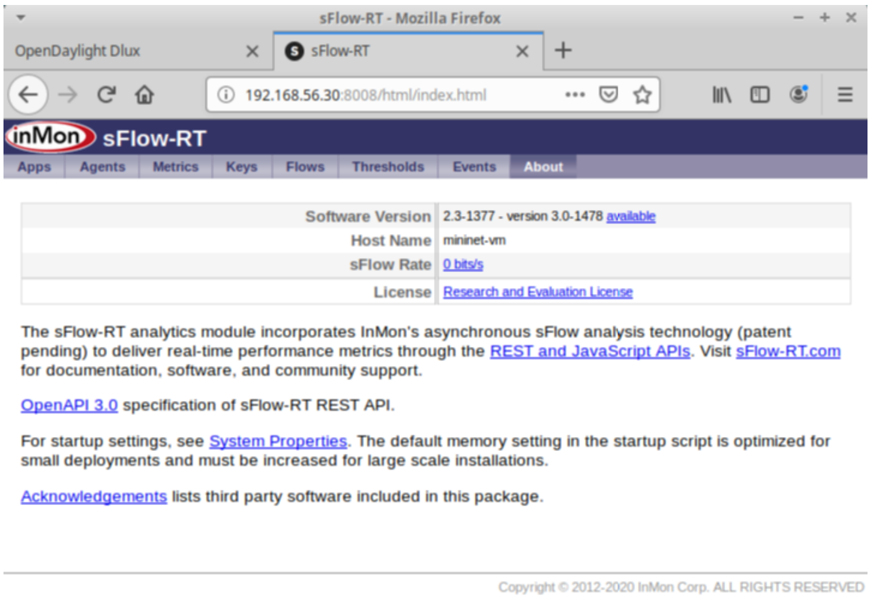


Figure 5.6 – OpenDayLight GUI



Software Version	2.3-1377 - version 3.0-1478 available
Host Name	mininet-vm
sFlow Rate	0 bits/s
License	Research and Evaluation License

The sFlow-RT analytics module incorporates InMon's asynchronous sFlow analysis technology (patent pending) to deliver real-time performance metrics through the REST and JavaScript APIs. Visit sFlow-RT.com for documentation, software, and community support.

OpenAPI 3.0 specification of sFlow-RT REST API.

For startup settings, see System Properties. The default memory setting in the startup script is optimized for small deployments and must be increased for large scale installations.

Acknowledgements lists third party software included in this package.

Copyright © 2012-2020 InMon Corp. ALL RIGHTS RESERVED

Figure 5.7 – sFlow-RT GUI

5.1 DDoS attack

The type of attack was done from one compromised host (h1), and the attacker (h2) does not have access to controller, so attacker (h2) must execute DDoS attack in the network by targeting h2 with an HTTP server on it. The intruder launches this kind of attack to overload the host 2 with the TCP packets over the switch connected to the controller, thereby consuming the entire controller bandwidth until the network does not respond, thereby affecting the entire network.

We used sFlow-RT to display the packets that we send via ODL, and it shows us whether any DDoS attack happens in the OpenFlow.

Attack was executed using **hping3**, which is a free packet generator and analyzer for the TCP/IP protocol. This tool has a wide variety of configurations to perform network exploration, traceroute, send pings and flood attacks at different levels. In summary, it offers a wide range of possibilities to configure packets and send them to different sites, with certain cadences. In addition, we can also specify the source IP address, using our IP, another IP or a random one.

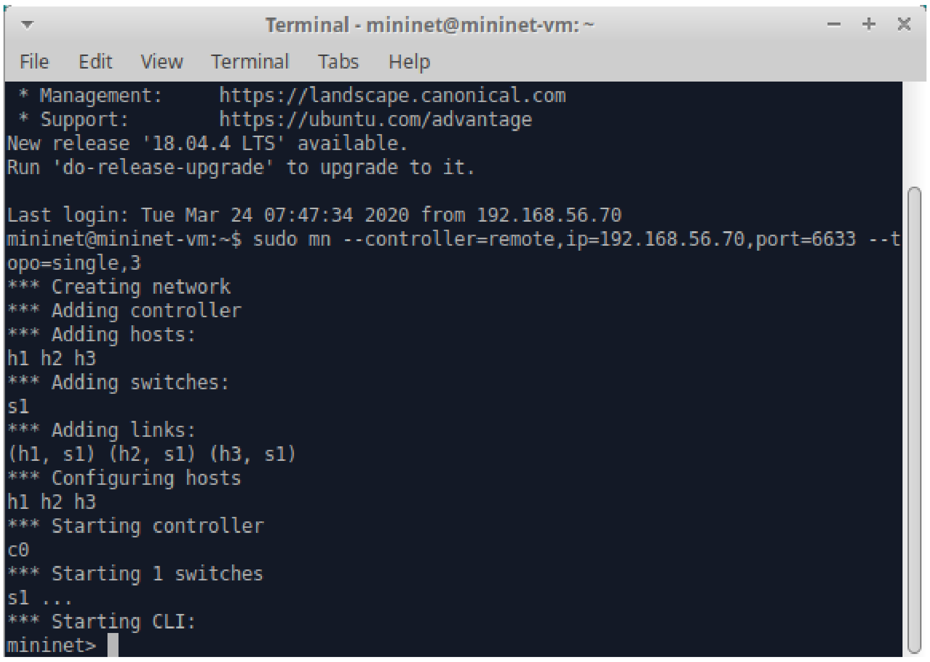
Implementation

First of all, we run pfSense VM to have our WAN and LAN networks with the host OS which was Xubuntu.

Secondary we start Mininet VM and run Mininet and sFlow-RT. This VM we use it only to run these two apps, because everything that was related with these two apps, we took it remotely with SSH from Xubuntu Desktop VM.

Thirdly we start OpenDayLight VM and run ODL and fourthly we start Xubuntu Desktop VM that was to monitor everything.

After the above, our next step was to remotely running Mininet (ssh -X mininet@192.168.56.30) from Xubuntu Desktop and with command: ***sudo mn --controller=remote,ip=192.168.56.70,port=6633 --topo= single,3*** we create through Mininet our network topology with one controller (c0), one switch (s1), and 3 hosts (h1, h2, h3).



```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
New release '18.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Mar 24 07:47:34 2020 from 192.168.56.70
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=192.168.56.70,port=6633 --t
opo=single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Figure 5.8 – Create Network at Mininet

After we run the above command to Mininet we can see our topology at ODL.

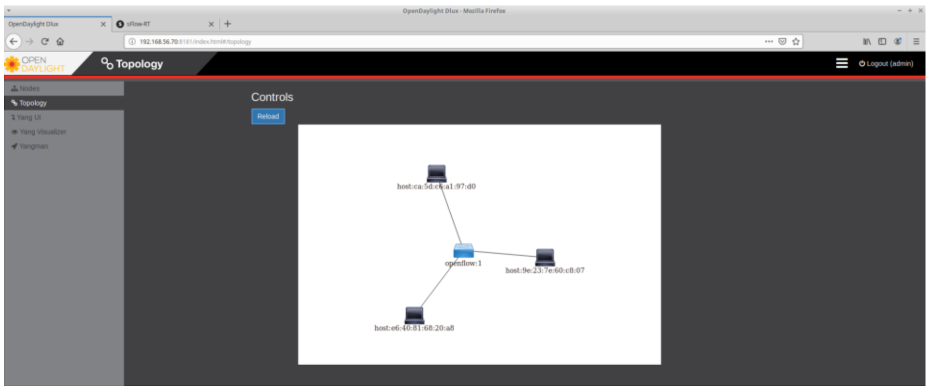
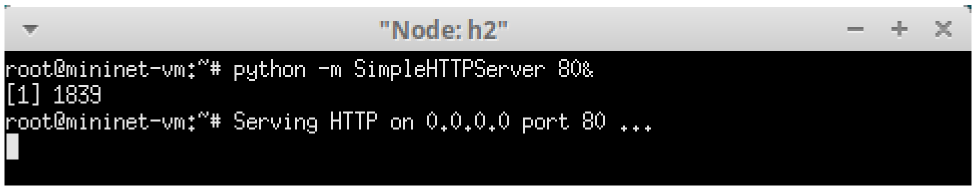


Figure 5.9 – Topology on ODL

Next we run again Mininet remotely from ODL vm and run the command: ***sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"192.168.56.30:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow.*** With this command we add sFlow-RT to our network devices and configures sFlow agent on each of the switches and collects samples of packet flows to be sent for analysis (if S = 10 and I = 20, it means, at every interval of 20 seconds, one out of 10 packets transiting the network will be forwarded to the collector for sampling). In our experiment, we have one switch, therefore, it will be repeated one time, for s1. If we had more switches, so many times it will be repeat it.

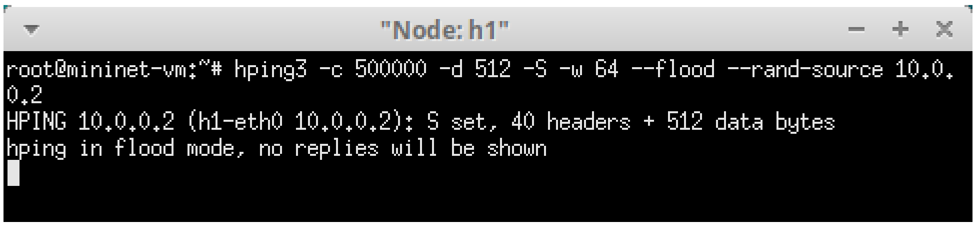
On Xubuntu Desktop we run also remotely Mininet and open with xterm h1 & h2. On h2 we create a virtual HTTP python Server with the command: ***python -m SimpleHTTPServer 80&***



"Node: h2"

```
root@mininet-vm:~# python -m SimpleHTTPServer 80&
[1] 1839
root@mininet-vm:~# Serving HTTP on 0.0.0.0 port 80 ...
```

Also, we made an attack to h2 from h1 with the command: ***hping3 -c 50000 -d 512 -S -w 64 --flood --rand-source 10.0.0.2*** (Flooding h2 with SYN TCP packets).



"Node: h1"

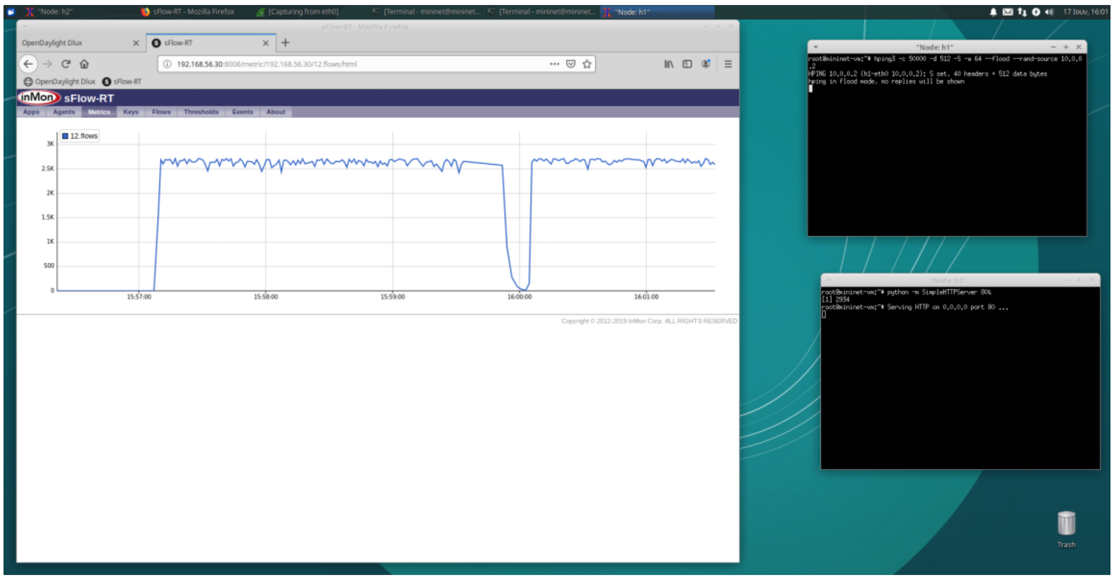
```
root@mininet-vm:~# hping3 -c 500000 -d 512 -S -w 64 --flood --rand-source 10.0.
0.2
HPING 10.0.0.2 (h1-eth0 10.0.0.2): S set, 40 headers + 512 data bytes
hping in flood mode, no replies will be shown
```

With these parameters we are indicating the following:

1. hping3 = Generator and analyzer of TCP / IP packets
2. -c = Define how many packets we send to target (50000 in our case)
3. -d = Define the size of a packet that we sent to target (512 in our case)
4. -S = Define the type of packet (SYN packets in our case)
5. -w = win (It se the TCP window size. Default is 64)
6. --flood = We send packets as fast as possible without show replies
7. --rand-source = We send packets with random source IP address. We can also use -a or –spoof to hide hostnames
8. 10.0.0.2 = IP of our host 2

Our Results:

In Figure 5.10 we can see the behavior of our controller after attack.

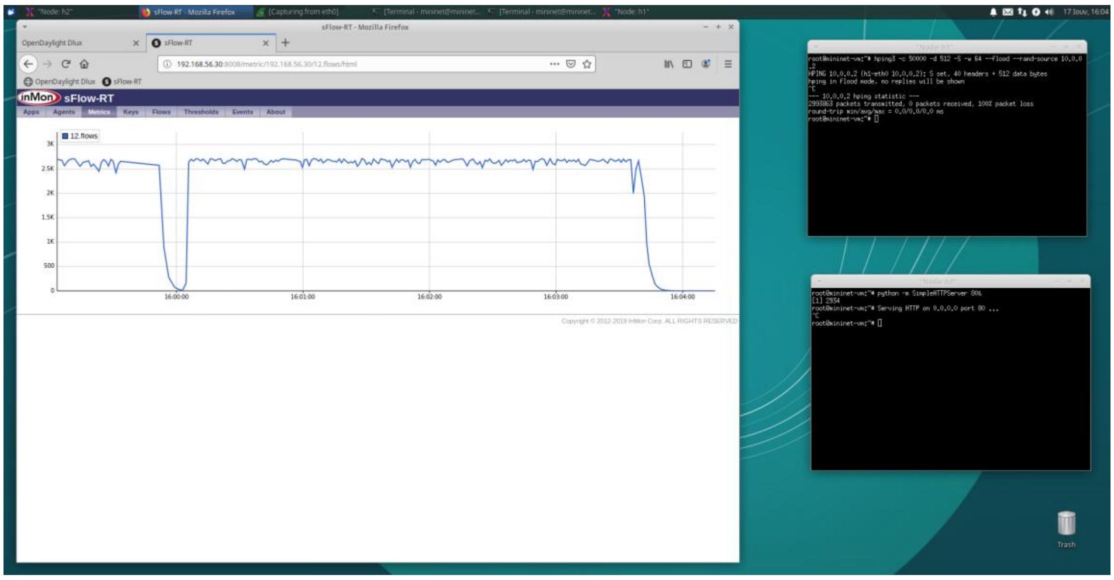


```
root@mininet-vm:~# hping3 -c 50000 -d 512 -S -w 64 --flood --rand-source 10.0.0
.2
HPING 10.0.0.2 (h1-eth0 10.0.0.2): S set, 40 headers + 512 data bytes
hping in flood mode, no replies will be shown
```

```
root@mininet-vm:~# python -m SimpleHTTPServer 80&
[1] 2934
root@mininet-vm:~# Serving HTTP on 0.0.0.0 port 80 ...
```

Figure 5.10

At Figure 5.11 we can see the end of the attack.



```
root@mininet-vm:~# hping3 -c 50000 -d 512 -S -w 64 --flood --rand-source 10.0.0
.2
HPING 10.0.0.2 (h1-eth0 10.0.0.2): S set, 40 headers + 512 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.2 hping statistic ---
2993863 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@mininet-vm:~# 
```

```
root@mininet-vm:~# python -m SimpleHTTPServer 80&
[1] 2934
root@mininet-vm:~# Serving HTTP on 0.0.0.0 port 80 ...
^C
root@mininet-vm:~# 
```

Figure 5.11

As we can see from the above diagrams, the controller resists DDoS attack even though we ran it for a long time and did not crash, which indicates that ODL is highly resistant to these attacks, but if we send more packages that we done, it will collapse as shown in Figure 5.12.

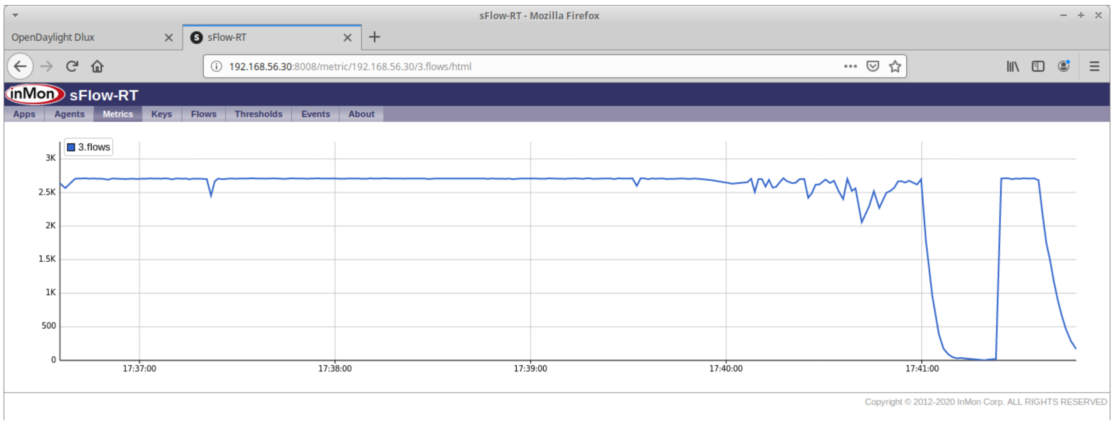
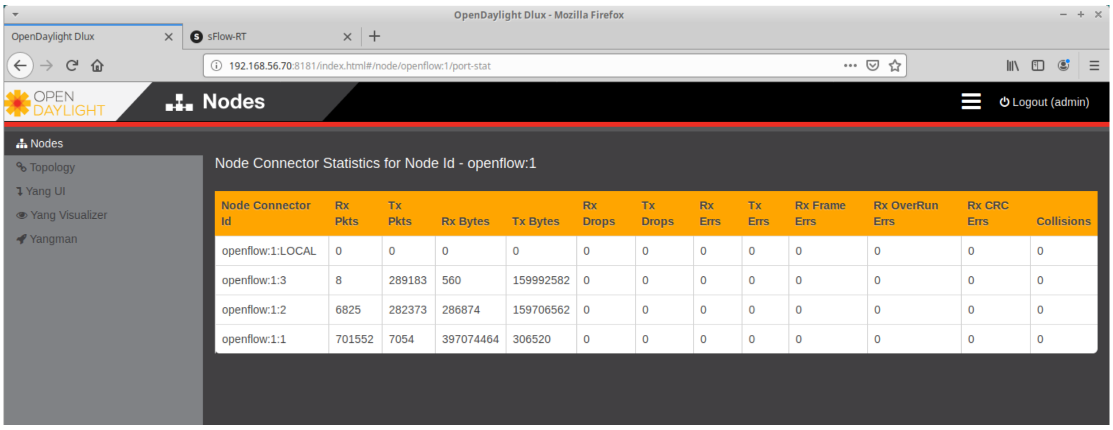


Figure 5.12

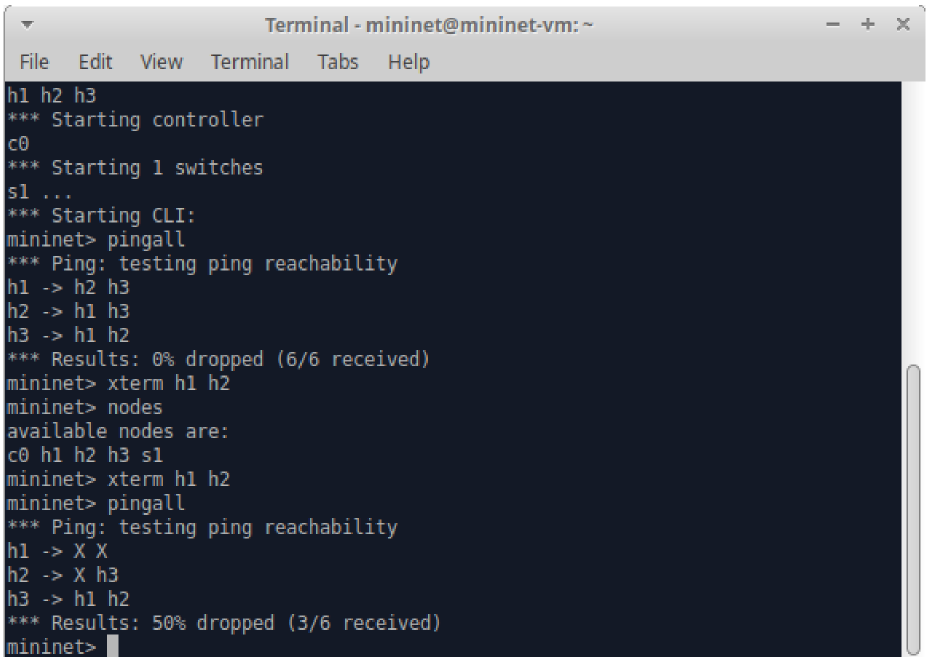
Also, in the below figures we can see the received and transmitted packets in our network and also the reachability to hosts after the execution of command **hping3**.



Node Connector Statistics for Node Id - openflow:1

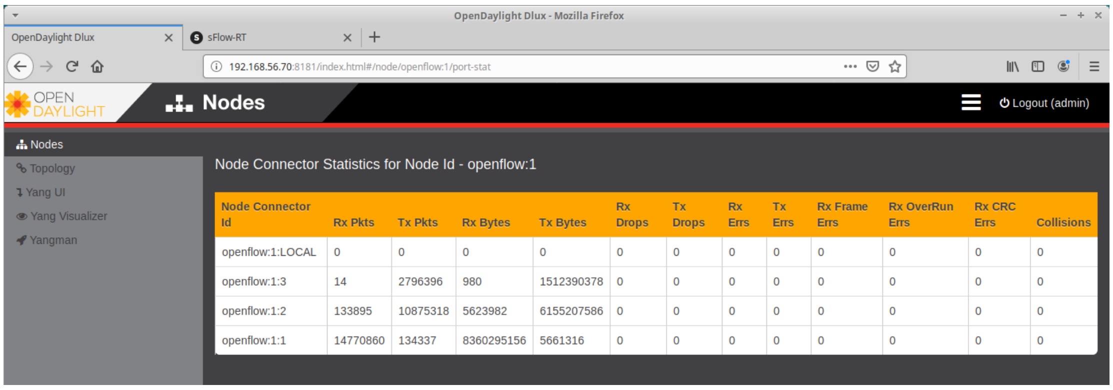
Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:3	8	289183	560	159992582	0	0	0	0	0	0	0	0
openflow:1:2	6825	282373	286874	159706562	0	0	0	0	0	0	0	0
openflow:1:1	701552	7054	397074464	306520	0	0	0	0	0	0	0	0

Figure 5.13 – We send 50000 packets


Terminal - mininet@mininet-vm: ~
File Edit View Terminal Tabs Help

```
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> xterm h1 h2
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet> xterm h1 h2
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X h3
h3 -> h1 h2
*** Results: 50% dropped (3/6 received)
mininet>
```

Figure 5.14 – Reachability after hping3


Node Connector Statistics for Node Id - openflow:1

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:3	14	2796396	980	1512390378	0	0	0	0	0	0	0	0
openflow:1:2	133895	10875318	5623982	6155207586	0	0	0	0	0	0	0	0
openflow:1:1	14770860	134337	8360295156	5661316	0	0	0	0	0	0	0	0

Figure 5.15 – We send 500000 packets

5.2 Analysis

As we saw ODL controller has avoided the DDoS for long time and still has not gone down, which indicates that ODL is highly resistant to this kind of attacks, but after we send more packets it went down because the virtual machine doesn't allow high-rate packets to crash. So, we can say that controller plane has bigger bandwidth relative to other controllers, and this is the reason that ODL resists more and has a logically distributed architecture that minimizes the effect of DDoS attack by not collecting all packets directly. To protect the SDN network, ODL controller requires a security communication channel between switch and controller.

ODL controller was chosen to run OpenFlow protocol to be tested by conducting a DDoS attack. The controller showed good behavior towards the DDoS attack after the start of the attack, since the idle timeout provided to the controller was small which allowed it to resist the attack. ODL had a wider bandwidth which helped him have better attack resistance. Ideal idle timeout and flow aggregation mechanisms have been found to making the SDN network more robust to DDoS attack. Since ODL is now the most common controller in SDN networks, its security measures need to be strengthened so that it can deal with DDOS attacks in order not to crash.

By design, ODL controller has a high bandwidth which is why it has been more resistant to the DDoS attacks. In addition, to detect and mitigate DDoS attacks, we must secure more the southbound API.

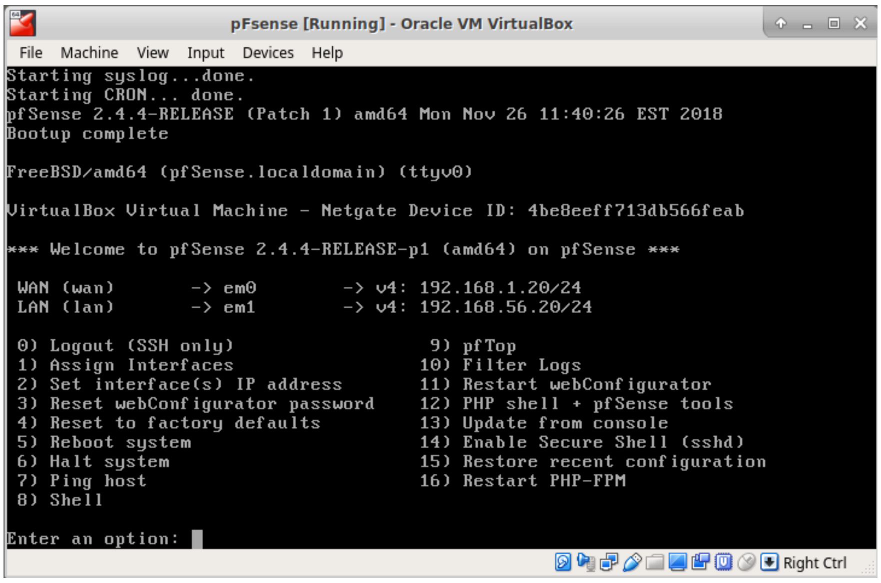
6. Proposed methodology for detection and response of cyberattacks over a production-based enterprise SDN network

After the above simulation with the DoS/DDoS attack that happened on our SDN infrastructure, we implemented the same scenario but this time we created an IDS/IPS mechanism where through it the intruder is detected and isolated in order to avoid such attacks on our infrastructure now but also in the near future.

For this purpose in VirtualBox we create the above VM's:

- PfSense VM as a firewall/router for WAN and LAN traffic control, so it help us to avoid any possible external factors in our infrastructure.
- Xubuntu Desktop VM for creating the topology through Mininet, run the controller which is Ryu and run the Intrusion Detection and Prevention System (IDS/IPS) that is Snort.

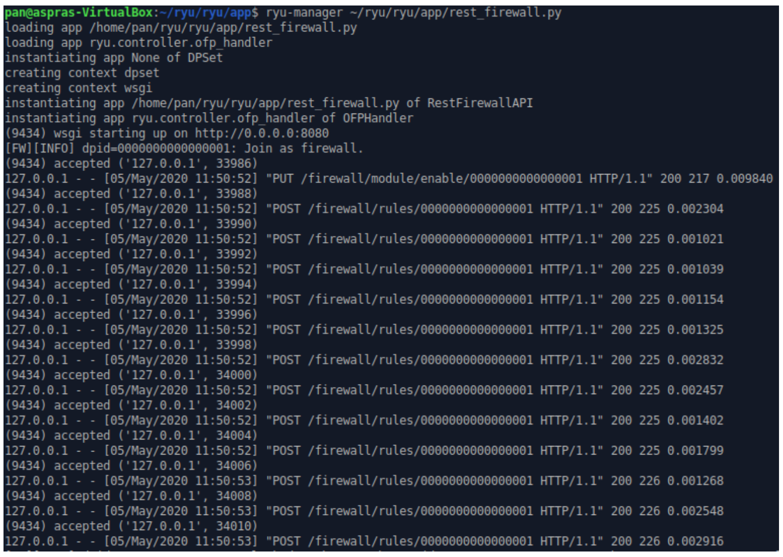
Snort used as IDS/IPS and is installed in the application plane while Ryu as the controller is in the control plane. Each packet on the network will first pass Ryu, which will then be checked by Snort based on matching with the rules contained in the Snort configuration. If the packet is considered secure, Ryu will be forwarded to the destination immediately, but if the packet is indicated to be hazardous, then Snort will send an alert and set the length of time that blocks the host from which the packet originated with an initial period. Unless the host keeps attacking in less time we put, then the blocking time will be decided by the frequency and type of attack.


pFsense [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Starting syslog...done.
Starting CRON... done.
pfSense 2.4.4-RELEASE (Patch 1) amd64 Mon Nov 26 11:40:26 EST 2018
Bootup complete

FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

VirtualBox Virtual Machine - Netgate Device ID: 4be8eeff713db566feab

*** Welcome to pfSense 2.4.4-RELEASE-p1 (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4: 192.168.1.20/24
 LAN (lan)       -> em1        -> v4: 192.168.56.20/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
 4) Reset to factory defaults         13) Update from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell

Enter an option:
```

Figure 6.1 – pfSense

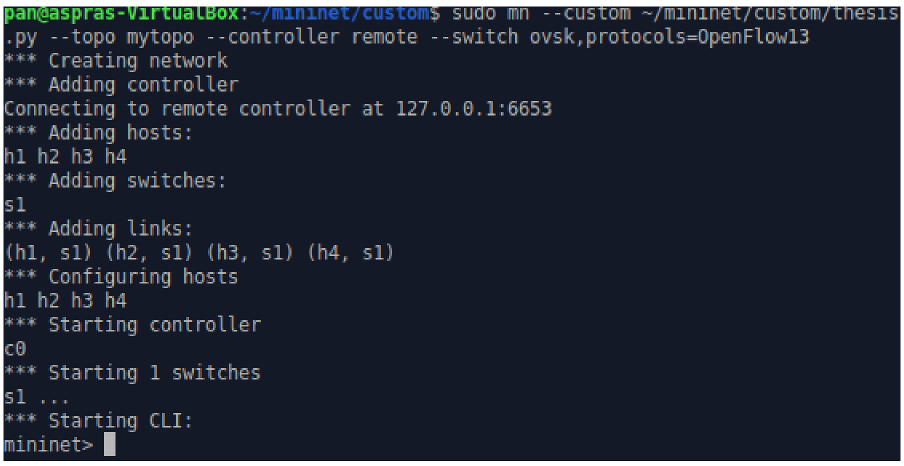
First of all, we run Ryu as Firewall with the **rest_firewall** app as shown below. By default the traffic to and from all Hosts will be blocked by Ryu due to Firewall app.



```
pan@aspras-VirtualBox:~/ryu/ryu/app$ ryu-manager ~/ryu/ryu/app/rest_firewall.py
loading app /home/pan/ryu/ryu/app/rest_firewall.py
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app /home/pan/ryu/ryu/app/rest_firewall.py of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(9434) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=0000000000000001: Join as firewall.
(9434) accepted ('127.0.0.1', 33986)
127.0.0.1 - - [05/May/2020 11:50:52] "PUT /firewall/module/enable/0000000000000001 HTTP/1.1" 200 217 0.009840
(9434) accepted ('127.0.0.1', 33988)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.002304
(9434) accepted ('127.0.0.1', 33990)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001021
(9434) accepted ('127.0.0.1', 33992)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001039
(9434) accepted ('127.0.0.1', 33994)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001154
(9434) accepted ('127.0.0.1', 33996)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001325
(9434) accepted ('127.0.0.1', 33998)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.002832
(9434) accepted ('127.0.0.1', 34000)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.002457
(9434) accepted ('127.0.0.1', 34002)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001402
(9434) accepted ('127.0.0.1', 34004)
127.0.0.1 - - [05/May/2020 11:50:52] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 225 0.001799
(9434) accepted ('127.0.0.1', 34006)
127.0.0.1 - - [05/May/2020 11:50:53] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 226 0.001268
(9434) accepted ('127.0.0.1', 34008)
127.0.0.1 - - [05/May/2020 11:50:53] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 226 0.002548
(9434) accepted ('127.0.0.1', 34010)
127.0.0.1 - - [05/May/2020 11:50:53] "POST /firewall/rules/0000000000000001 HTTP/1.1" 200 226 0.002916
```

Figure 6.2 – Ryu controller run as firewall

Via Mininet with the command: ***sudo mn --custom ~/mininet/custom/thesistopology.py --topo mytopo --controller remote --switch ovsk,protocols=OpenFlow13*** we create our network with one controller (c0), one switch (s1), and 4 hosts (h1, h2, h3 and h4).



```
pan@aspras-VirtualBox:~/mininet/custom$ sudo mn --custom ~/mininet/custom/thesis
.py --topo mytopo --controller remote --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Figure 6.3 – Create Network at Mininet

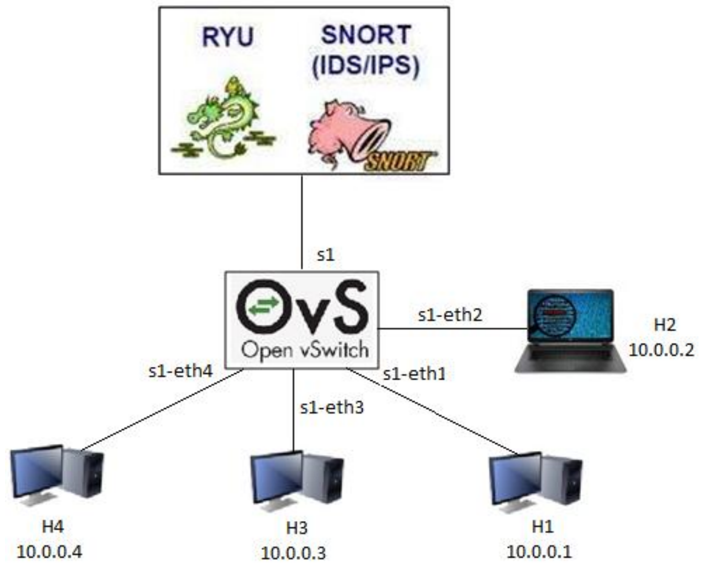
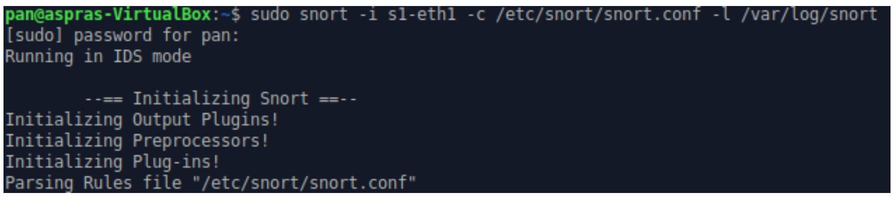


Figure 6.4 – Our topology

After running Ryu and Mininet, the next step is to add a flow entry to Ryu to instruct the OpenFlow switch about what to do with incoming packets so that each device can communicate. First we activate the firewall on each switch with the command: **curl -X PUT http://localhost:8080/firewall/module/enable/[idswitch]** and then we add flow entries for all devices and in two directions with the command: **curl -X POST -d '{"nw_src": "[nw_src]","nw_dst": "[nw_dst]"}' http://localhost:8080/firewall/rules/[idswitch]** as we can see in Figure 5.16.

Also with the following command, we integrate Snort so that it starts on s1-eth1 port.

```
sudo snort -i s1-eth1 -c /etc/snort/snort.conf -l /var/log/snort
```



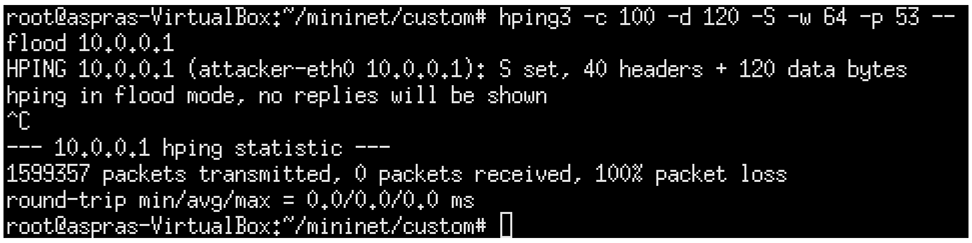
```
pan@aspras-VirtualBox:~$ sudo snort -i s1-eth1 -c /etc/snort/snort.conf -l /var/log/snort
[sudo] password for pan:
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
```

So with the above we have Ryu running like Firewall and in our s1-eth1 interface of our topology we running Snort where in the configuration file “snort.conf” we have set the Rules (Snort Database) that we want and also the file that we have created to record the time off attacks, the IP source, the IP destination, the protocol and the message that it receives from the Rules after each unwanted intrusion into our network.

Our next step is to execute the control file that will see the last line of our log file, and test the time difference from the last attack and the prior attack by a host.

6.1 DDoS attack

First we made a DoS attack from one of our hosts that have an IP address 10.0.0.2 to a host with an IP address 10.0.0.1 which can be everything like DNS server. This attack technique used to send a SYN flooding packet to the target host. To carry out this attack, the command we used is: ***hping3 -c 100 -d 120 -S -w 64 -p 53 --flood 10.0.0.1***



```
root@aspras-VirtualBox:~/mininet/custom# hping3 -c 100 -d 120 -S -w 64 -p 53 --
flood 10.0.0.1
HPING 10.0.0.1 (attacker-eth0 10.0.0.1): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.1 hping statistic ---
1599357 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@aspras-VirtualBox:~/mininet/custom# 
```

With these parameters we are indicating the following:

1. hping3 = Generator and analyzer of TCP / IP packets
2. -c = Define how many packets we send to target (100 in our case)
3. -d = Define the size of a packet that we sent to target (120 in our case)
4. -S = Define the type of packet (SYN packets in our case)
5. -w = win (It se the TCP window size. Default is 64)
6. -p = destination port
7. --flood = We send packets as fast as possible without show replies
8. 10.0.0.1 = IP of our host (It can be anything like Server)

The rules that we used so that Snort can recognize DoS attack:

alert tcp any any -> any any (sid: 2; msg: "TCP SYN packet flooding (simple or distributed) attempt"; threshold: type both, track by_dst, count 10000, seconds 60; flow:stateless; flags:S,12; rev:1;)

Rule Header analysis:

alert – Rule action. Snort will generate an alert when the set condition is met.

tcp – Protocol we use.

any – Source IP. Snort will look at all sources.

any – Source port. Snort will look at all ports.

-> – Direction. From source to destination.

any – Destination IP. Snort will look at all sources.

any – Destination port. Snort will look at all ports on the protected network.

Rule Options analysis:

sid – Snort rule ID (In our case is a rule with the Snort Rule ID **2**)

msg – Snort will include this message with the alert.

threshold – **Type both**: alerts once per time interval after seeing m occurrences of the event, then ignores any additional events during the time interval. **Track by_dst**: Rate is tracked by destination IP address. This means count is maintained for each unique destination IP addresses. Ports or anything else are not tracked. **Count 1000**: Number of rule matching in s seconds that will cause event_filter limit to be exceeded (**s = 1000**). **Seconds 60**: Time period over which count is accrued (**60 in our case**).

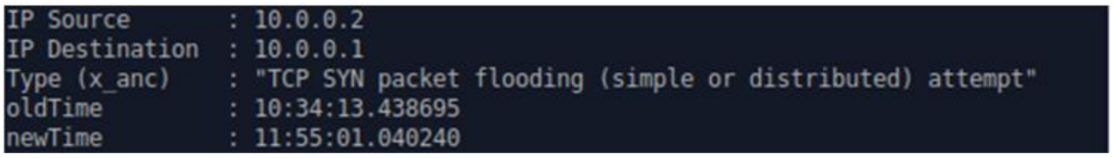
flow – **Stateless** option is used to apply the rule without considering the state of a TCP session.

flags – Finding a SYN (**S**) packet regardless of the values of the reserved bits 1 and 2.

rev – Revision number. This option allows for easier rule maintenance. In our case is a rule with the Snort Rule Revision of **1**.

Our Results:

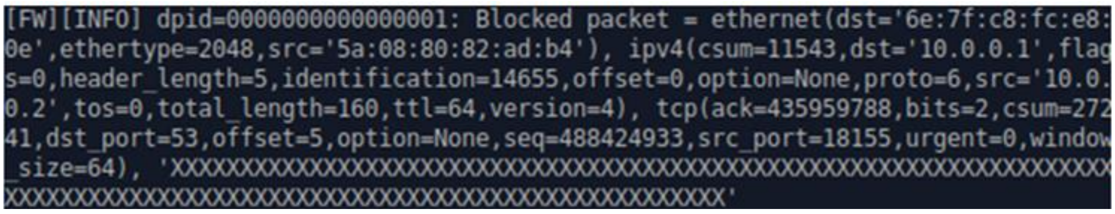
In Figure 6.5 we can see that our IDS mechanism detect DoS attack.



```
IP Source        : 10.0.0.2
IP Destination   : 10.0.0.1
Type (x_anc)     : "TCP SYN packet flooding (simple or distributed) attempt"
oldTime          : 10:34:13.438695
newTime          : 11:55:01.040240
```

Figure 6.5 – Detection of DoS attack

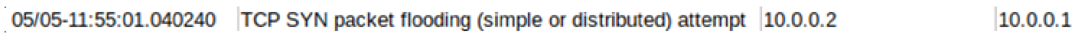
And in Figure 6.6 we can see that also our IPS mechanism blocking the attack.



```
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='6e:7f:c8:fc:e8:
0e',ethertype=2048,src='5a:08:80:82:ad:b4'), ipv4(csum=11543,dst='10.0.0.1',flag
s=0,header_length=5,identification=14655,offset=0,option=None,proto=6,src='10.0.
0.2',tos=0,total_length=160,ttl=64,version=4), tcp(ack=435959788,bits=2,csum=272
41,dst_port=53,offset=5,option=None,seq=488424933,src_port=18155,urgent=0,window
_size=64), 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
```

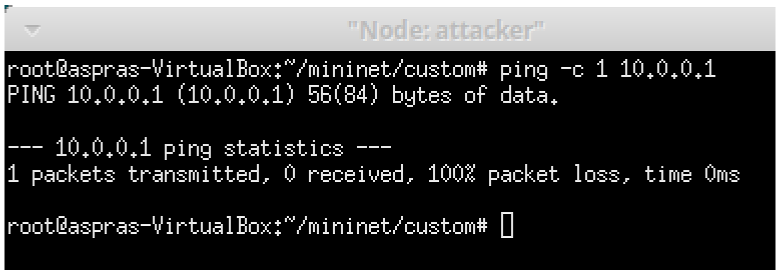
Figure 6.6 – Blocking DoS attack

And this is the message that was record in our csv file.



05/05-11:55:01.040240	TCP SYN packet flooding (simple or distributed) attempt	10.0.0.2	10.0.0.1

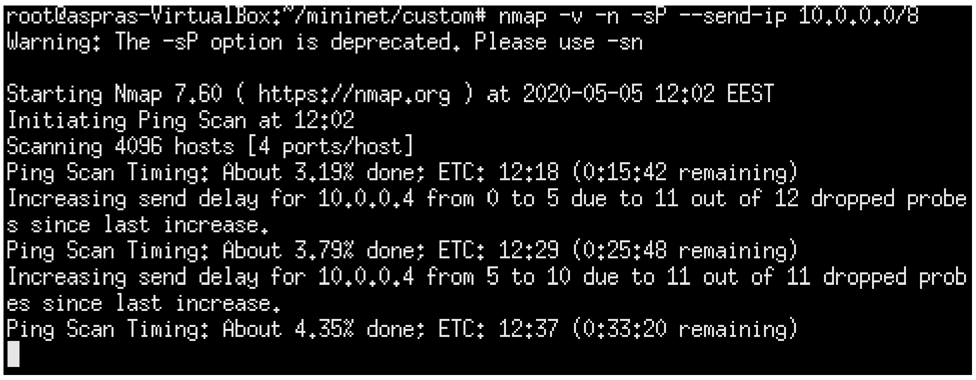
Host with IP 10.0.0.2 which is the attacker (h2) as shown below has been blocked.



```
root@aspras-VirtualBox:~/mininet/custom# ping -c 1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@aspras-VirtualBox:~/mininet/custom# 
```

6.2 Nmap attack

During this attack, Snort rules are checked on all hosts connected to the same subnet as the attacker (h2) for host discovery operation with a form of ping scan. The command we used to handle this is: **nmap -v -n -sP --send-ip 10.0.0.0/8**



```
root@aspras-VirtualBox:~/mininet/custom# nmap -v -n -sP --send-ip 10.0.0.0/8
Warning: The -sP option is deprecated. Please use -sn

Starting Nmap 7.60 ( https://nmap.org ) at 2020-05-05 12:02 EEST
Initiating Ping Scan at 12:02
Scanning 4096 hosts [4 ports/host]
Ping Scan Timing: About 3.19% done; ETC: 12:18 (0:15:42 remaining)
Increasing send delay for 10.0.0.4 from 0 to 5 due to 11 out of 12 dropped probe
s since last increase.
Ping Scan Timing: About 3.79% done; ETC: 12:29 (0:25:48 remaining)
Increasing send delay for 10.0.0.4 from 5 to 10 due to 11 out of 11 dropped prob
es since last increase.
Ping Scan Timing: About 4.35% done; ETC: 12:37 (0:33:20 remaining)
```

With these parameters we are indicating the following:

1. nmap = Network Mapper
2. -v = Giving more detailed information about the remote machine
3. -n = Disable reverse DNS resolution
4. -sP = Find out live hosts in a Network, Discover IP's in a subnet
5. --send-ip = Send packets using raw IP sockets
6. 10.0.0.0/8 = our network

By using Nmap, the attacker can scan all hosts that are currently connected. Information about the list of hosts that contains the ports and services that are being used for each host is quite sensitive because it allows users to take dangerous actions on the network.

An attacker host (h2) scanning the 10.0.0.0/8 network which is the subnet that we use in our topology. From this experiment, the attacker (h2) can find out information about the network. At the same time, Snort will release dangerous information that has been done by the attacker and keep it in the log. The rules that we used so that Snort can recognize packages that contain a host discovery experiment is:

alert icmp any any -> any any (sid:1; msg:"Host discovery (nmap ping scan) attempt"; dsize:0; itype:8; reference:arachnids,162; classtype:attempted-recon; rev:1;)

Rule Header analysis:

alert – Rule action. Snort will generate an alert when the set condition is met.

icmp – Protocol we use.

any – Source IP. Snort will look at all sources.

any – Source port. Snort will look at all ports.

-> – Direction. From source to destination.

any – Destination IP. Snort will look at all sources.

any – Destination port. Snort will look at all ports on the protected network.

Rule Options analysis:

sid – Snort rule ID (In our case is a rule with the Snort Rule ID **1**).

msg – Snort will include this message with the alert.

dsize – Is used to test the packet payload size. In our case is **0**.

itype – Is used to check for a specific ICMP type value. In our case is **8**.

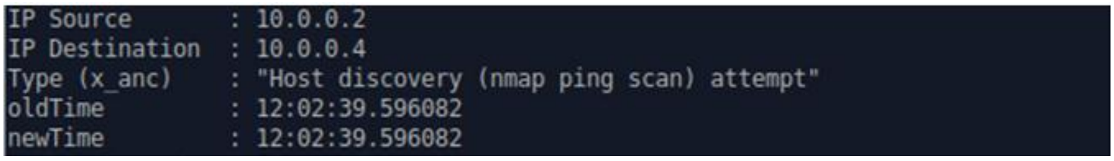
reference – Allow rules to include references to external attack identification systems. The plugin currently supports several specific systems as well as unique URLs. This plugin is to be used by output plugins to provide a link to additional information about the alert produced.

classtype - Is used to categorize a rule as detecting an attack that is part of a more general type of attack class. Snort provides a default set of attack classes that are used by the default set of rules it provides. Defining classifications for rules provides a way to better organize the event data Snort produces. **Attempted-recon =** Attempted Information Leak.

rev – Revision number. This option allows for easier rule maintenance. In our case is a rule with the Snort Rule Revision of **1**.

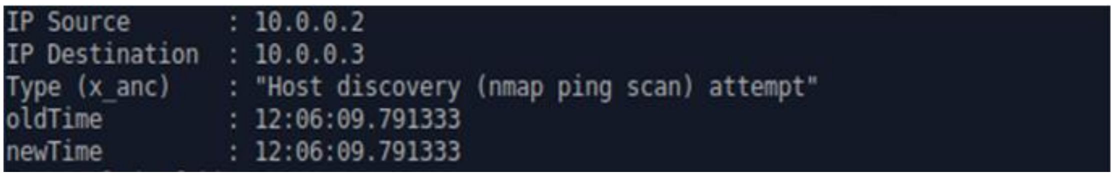
Our Results:

In Figure 6.7 and 6.8 we can see the Detection of nmap.



```
IP Source        : 10.0.0.2
IP Destination   : 10.0.0.4
Type (x_anc)     : "Host discovery (nmap ping scan) attempt"
oldTime          : 12:02:39.596082
newTime          : 12:02:39.596082
```

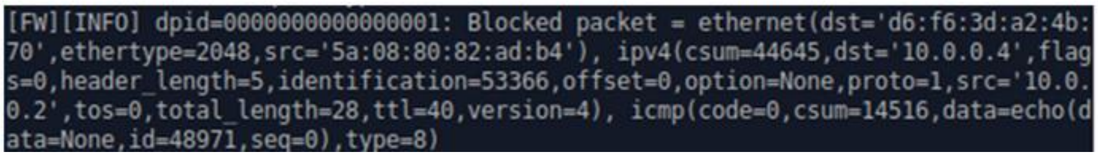
Figure 6.7 – Detection of nmap on 10.0.0.4



```
IP Source        : 10.0.0.2
IP Destination   : 10.0.0.3
Type (x_anc)     : "Host discovery (nmap ping scan) attempt"
oldTime          : 12:06:09.791333
newTime          : 12:06:09.791333
```

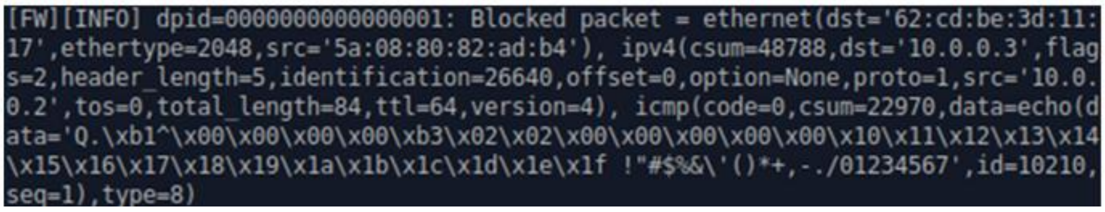
Figure 6.8 – Detection of nmap on 10.0.0.3

And in Figure 6.9 and 6.10 we can see the Blocking of this attack.



```
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='d6:f6:3d:a2:4b:
70',ethertype=2048,src='5a:08:80:82:ad:b4'), ipv4(csum=44645,dst='10.0.0.4',flag
s=0,header_length=5,identification=53366,offset=0,option=None,proto=1,src='10.0.
0.2',tos=0,total_length=28,ttl=40,version=4), icmp(code=0,csum=14516,data=echo(d
ata=None,id=48971,seq=0),type=8)
```

Figure 6.9 – Blocking attack on 10.0.0.4



```
[FW][INFO] dpid=0000000000000001: Blocked packet = ethernet(dst='62:cd:be:3d:11:
17',ethertype=2048,src='5a:08:80:82:ad:b4'), ipv4(csum=48788,dst='10.0.0.3',flag
s=2,header_length=5,identification=26640,offset=0,option=None,proto=1,src='10.0.
0.2',tos=0,total_length=84,ttl=64,version=4), icmp(code=0,csum=22970,data=echo(d
ata='Q.\xb1^\x00\x00\x00\x00\xb3\x02\x02\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14
\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567',id=10210,
seq=1),type=8)
```

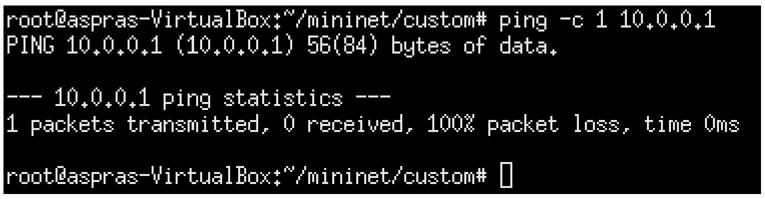
Figure 6.10 – Blocking attack on 10.0.0.3

And these are the messages that were record in our file.

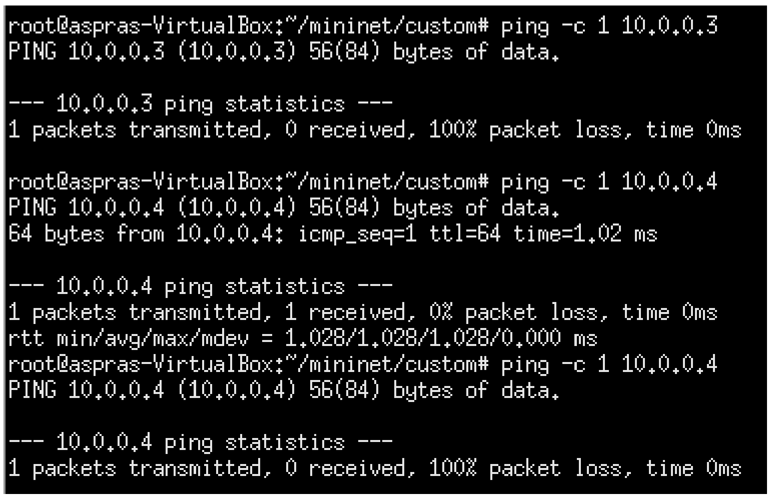
05/05-12:02:39.596082	Host discovery (nmap ping scan) attempt	10.0.0.2	10.0.0.4

05/05-12:06:09.791333	Host discovery (nmap ping scan) attempt	10.0.0.2	10.0.0.3

Host with IP 10.0.0.2 which is the attacker (h2) as shown below has been blocked.



```
root@aspras-VirtualBox:~/mininet/custom# ping -c 1 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@aspras-VirtualBox:~/mininet/custom# 
```



```
root@aspras-VirtualBox:~/mininet/custom# ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@aspras-VirtualBox:~/mininet/custom# ping -c 1 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=1.02 ms

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.028/1.028/1.028/0.000 ms
root@aspras-VirtualBox:~/mininet/custom# ping -c 1 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

6.3 Analysis

Snort is used as IDS/IPS and is installed at the application level while Ryu as the controller is at the control level. Every move on the network is first passed by Ryu, which is then controlled by Snort based on matching with our predefined database (snortdata.rules). If declared safe, then Ryu is immediately forwarded to the destination, but if it is declared dangerous, then Snort sends a warning and sets a timeout of minutes that we set. If the attack continues for less than the minutes we set, the time of exclusion is determined by the attack frequency and type.

The SDN network, which has been fitted with adaptive IDS / IPS, has the ability to detect cyber attacks from the results of our studies, and can also block attacker hosts with a length that is in line with the frequency and form of attacks that have been carried out. The final results gained make the SDN network more secure.

7. Conclusions and Future Extensions

In this thesis we analyzed the technologies of Software Defined Networks, Network Function Virtualization and Deep Packet Inspection and saw how they can coexist.

With the help of Virtual Machines, we implemented a small-scale production infrastructure using various operating systems and implemented an attack scenario on it and saw what our infrastructure was reacting to and what to do to avoid such attacks in the future.

The main purpose of this work was to analyze and implement some of the available tools for SDN architecture so as to present a safer way to use these networks.
As we know, various different security mechanisms can be used to ensure the integrity of SDN networks, which can be selected according to the desired configuration options.

A low-cost scenario has been designed and implemented to integrate an IDS / IPS into an SDN controller such as Ryu to detect and prevent attacks. RYU controller is programmed in the python language and security can be applied to the controller code itself. As a result, programming in a language that allows multiple options can bring significant benefits, and in addition, applying the controller and security in the same module involves concentration.

After several tests, we came to the conclusion that Ryu controller has a perfect partnership with Snort tool, which, if properly configured, gives us a strong and inexpensive solution for an attack detection and control system.

Compared to other ways of detecting and preventing attacks, we came to the conclusion that other ways of detect and prevent attacks are much more costly with the use of various tools such as Raspberry and Zodiac [58], another way is to use a SQL database [59] but this can create problems with long delays on the network with the conclusion that the system is crashing, and another way of detecting and preventing attacks is to tease the ports of our network [60] but that will it results in more exposure.

We are not saying that our choice for IDS/IPS is the best, but by having a system with limited capabilities we have come to the conclusion that the combination of tools we used to experimentally implement this position leaves us with an optimism based on the results that we took.

In the context of further development of this thesis, improvements can be made to the types of network attacks that can be addressed based on the above system.

In addition, instead of Snort, we can use other commercially available programs that can also detect and combat network attacks that have not yet been tested on SND networks.

As future research, one can use commercially available programs that do not detect malicious information with rules such as the Snort IDS / IPS program, but recognize it through contemplative data by comparing information that represents the image of the network before the attack and information representing the image of the same network after an attack. Thus, through this thoughtful analysis, these programs can recognize the information that threatens a network and in this case the network they are protecting.

Finally, another future extension is that when we detect and combat a threat, then intelligent network control platforms can exchange messages with each other to know an attack and also update user information that has tried to infect the network with malicious information and take automatic measures to exclude the malicious user, so that they do not manage to infect smart networks in the future and prevent the threat of networks before it even starts.

References

[1] Michael Rolbin, “Early detection of network threats using Software Defined Network (SDN) and virtualization” Carleton University, Ottawa, Ontario, Canada, December 2013

[2] Ahmed AlErouda, Izzat Alsmadib, “Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach”, Journal of Network and Computer Applications, Volume 80, 15 February 2017, Pages 152-164

[3] Patrick Dyroff, “Implementation of an SSL Proxy in a Software Defined Network Aboard a Naval Ship to Monitor Internet Traffic”, Open Networking Summit 2016, 14-17 March 2016, Santa Clara USA

[4] Network operators, “Network Functions Virtualisation. An Introduction, Benefits, Enablers, Challenges & Call for Action”, October 22-24, 2012 “SDN and OpenFlow World Congress”, Darmstadt-Germany

[5] Telco Systems, “Protecting SDN and NFV Networks from Cyber Security Vulnerabilities”, November 2015

[6] Georgios Sextos, Stamatis Mandilas, “Security of Digital Systems”, December 2015

[7] Ivan Kupalov, “Software Defined Networks Case Study”, February 2016

[8] Where SDN and DPI technology meet, http://searchsdn.techtarget.com/tip/Where-SDN-and-DPI-technology-meet-Centralized-control-and-automation, January 2019

[9] Network Virtualization Security, http://www.securityweek.com/network-virtualization-and-what-it-means-security, January 2019

[10] Defending Cyberspace with Software-Defined Networks, https://www.jinfowar.com/journal/volume-14-issue-2/defending-cyberspace-software-defined-networks, January 2019

[11] L. Hardesty, «Google brings SDN to the Public Internet - SDxCentral,» [En línea]. Available: https://www.sdxcentral.com/articles/news/google-brings-sdn-public-internet/2017/04/. [Último acceso: 13 Junio 2018], January 2019

[12] SDN, https://www.opennetworking.org/sdn-definition/, January 2019

[13] Software Defined Networking, https://en.wikipedia.org/wiki/Software-defined_networking, February 2019

[14] Benzekki, Kamal; El Fergougui, Abdeslam; Elbelrhiti Elalaoui, Abdelbaki (2016). "Software-defined networking (SDN): A survey". Security and Communication Networks. 9 (18): 5803–5833. doi:10.1002/sec.1737, February 2019

[15] Software-defined networking is not OpenFlow, companies proclaim, "Software-defined networking is not OpenFlow, companies proclaim". searchsdn.techtarget.com, February 2019

[16] InCNTRE's OpenFlow SDN testing lab works toward certified SDN product, "InCNTRE's OpenFlow SDN testing lab works toward certified SDN product". February 2019

[17] Predicting SD-WAN Adoption, "Predicting SD-WAN Adoption". gartner.com. 2015-12-15. Retrieved 2016-06-27. February 2019

[18] Software-Defined Networking (SDN), https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN, March 2019

[19] ForCES - Halpern et Hadi, Salim. (2010). «RFC 5812 Forwarding and Control Element Separation (ForCES) Forwarding Element Model». Repéré à <https://tools.ietf.org/html/rfc5812>, March 20192019

[20] Clarke, Salgueiro et Pignataro. (2016). «RFC 7922 - Interface to the Routing System (I2RS) Traceability: Framework and Information Model». Repéré à <https://tools.ietf.org/html/rfc7922>, March 20192019

[21] Enns, Bjorklund, Schoenwaelder and Bierman. (2011). "RFC 6241 - Network to Setup Protocol (NETCONF)." Spotted at <https://tools.ietf.org/html/rfc6241>. Giotis, Kostas Argyropoulos, March 20192019

[22] EFORT. (2016). «Le Protocole OpenFlow dans l'Architecture SDN». Repéré à <http://www.efort.com/r_tutoriels/OPENFLOW_EFORT.pdf>, March 20192019

[23] Ali Elamrani Joutei - Ali Elamrani Joutei. (2014). «Inside OpenFlow! ConfigNetworks». Repéré à <http://confignetworks.com/inside-openflow/>, April 2019

[24] Sowell - Greg Sowell Consulting. (2013). «OpenFlow And Mikrotik». Repéré à <http://gregsowell.com/?p=4442>, April 2019

[25] Transport Layer Security, https://en.wikipedia.org/wiki/Transport_Layer_Security, February 2019

[26] OSI model, https://en.wikipedia.org/wiki/OSI_model, April 2019

[27] Snort. http://www.snort.org, April 2019

[28] Network Functions Virtualisation (NFV), http://www.etsi.org/technologies-clusters/technologies/nfv, April 2019

[29] Commercial off-the-shelf, https://en.wikipedia.org/wiki/Commercial_off-the-shelf, May 2019

[30] Network Functions Virtualisation, https://portal.etsi.org/NFV/NFV_White_Paper.pdf, May 2019

[31] Kubernetes, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/, May 2019

[32] KubeVirt: Building a virtualization API for Kubernetes, https://kubevirt.io/, May 2019

[33] OpenStack Ussuri Release Delivers Automation for Intelligent Open Infrastructure, https://www.openstack.org/, May 2019

[34] VMware, https://www.vmware.com/, June 2019

[35] Bare-metal server, https://en.wikipedia.org/wiki/Bare-metal_server, June 2019

[36] ONF: OpenFlow Switch Specification, https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf, June 2019

[37] Information security, https://en.wikipedia.org/wiki/Information_security, June 2019

[38] Ntop, https://www.ntop.org/, June 2019

[39] Linux containers: LXC, https://linuxcontainers.org/lxc/introduction/, July 2019,

[40] https://linuxcontainers.org/lxd/introduction/, July 2019

[41] Get Started with Docker, https://www.docker.com/, July 2019

[42] Mininet: An Instant Virtual Network on your Laptop (or other PC), http://mininet.org/, July 2019

[43] OvS: Production Quality, Multilayer Open Virtual Switch, https://www.openvswitch.org/, July 2019

[44] Introduction to Mininet, https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what, August 2019

[45] The Linux Foundation Projects: OpenDayLight, https://www.opendaylight.org/, August 2019

[46] Harmonizing Open Source Networking, https://www.lfnetworking.org/, August 2019

[47] OpenDaylight, Most Pervasive Open Source SDN Controller, Celebrates Sixth Anniversary with Neon Release,

https://www.opendaylight.org/announcement/2019/03/26/opendaylight-most-pervasive-open-source-sdn-controller-celebrates-sixth-anniversary-with-neon-release, August 2019

[48] What is an OpenDaylight Controller, https://www.sdxcentral.com/networking/sdn/definitions/opendaylight-controller/, August 2019

[49] SDN Series Part Four: Ryu, a Rich-Featured Open Source SDN Controller Supported by NTT Labs, https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs, September 2019

[50] Ryu 4.34 documentation, https://ryu.readthedocs.io/en/latest/getting_started.html#what-s-ryu, September 2019

[51] Nippon Telegraph and Telephone, https://en.wikipedia.org/wiki/Nippon_Telegraph_and_Telephone, September 2019

[52] Nicira Extension Structures, https://ryu.readthedocs.io/en/latest/nicira_ext_ref.html, September 2019

[53] pfSense Overview, https://www.pfsense.org/about-pfsense/, September 2019

[54] inMon: sFlow-RT, https://inmon.com/products/sFlow-RT.php, November 2019

[55] Wireshark: Go Deep, https://www.wireshark.org/, November 2019

[56] Snort: Network Intrusion Detection & Prevention System, https://www.snort.org/, November 2019

[57] Martin Roesch, https://en.wikipedia.org/wiki/Martin_Roesch, November 2019

[58] DSpace ESPOCH: Integración de un IDS/IPS al controlador SDN para la prevención y detección de ataques de seguridad (DOS) en un escenario de Redes Definidas por Software, http://dspace.espoch.edu.ec/handle/123456789/10931, November 2019

[59] Ryu-kontrollerin ja Snort-tunkeilijan estojärjestelmän integraatio, https://www.theseus.fi/handle/10024/112213, December 2019

[60] A SDN-based Flexible System for On-the-Fly Monitoring and Treatment of Security Events, https://www.researchgate.net/publication/311706786_A_SDN-based_Flexible_System_for_On-the-Fly_Monitoring_and_Treatment_of_Security_Events, December 2019

[61] HACKMAGEDDON: Information Security Timelines and Statistics, https://www.hackmageddon.com/2020/06/03/april-2020-cyber-attacks-statistics/, December 2019

[62] AVANT-GUARD: Scalable and Vigilant Switch FlowManagement in Software-Defined Networks, http://faculty.cs.tamu.edu/guofei/paper/AvantGuard-CCS13.pdf, December 2019

[63] FLOWGUARD: Building Robust Firewallsfor Software-Defined Networks, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.683.3009&rep=rep1&type=pdf, December 2019

[64] NICE: A tool to test OpenFlow SDN controller, https://github.com/mcanini/nice/wiki, January 2020

[65] FRESCO: Modular Composable Security Servicesfor Software-Defined Networks, http://www.csl.sri.com/users/vinod/papers/fresco.pdf, February 2020

[66] FortNOX: A Security Enforcement Kernel for OpenFlow Networks, http://www.cs.columbia.edu/~lierranli/coms6998-8SDNFall2013/papers/FortNox-HotSDN2012.pdf, March 2020

[67] SnortFlow: A OpenFlow-Based Intrusion Prevention System in Cloud Environment, https://www.researchgate.net/publication/261049313_SnortFlow_A_OpenFlow-Based_Intrusion_Prevention_System_in_Cloud_Environment, April 2020

[68] Link Layer Discovery Protocol (LLDP), https://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol, May 2020

[69] The FlowAdapter: Enable Flexible Multi-Table Processingon Legacy Hardware, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.469.6931&rep=rep1&type=pdf, June 2020

[70] Vasos Hadjioannou, Constandinos X. Mavromoustakis, George Mastorakis, Jordi Mongay Batalla, Ioannis Kopanakis, Emmanouil Perakakis, Spyros Panagiotakis, "Security in Smart Grids and Smart Spaces for smooth IoT deployment in 5G", chapter contribution in the HandBook "Internet of Things (IoT) in 5G Mobile Technologies", Editors: Mavromoustakis, Constandinos X., Mastorakis, George, Batalla, Jordi Mongay, Springer-Verlag (2016), pp. 371-397.

[71] Koralia Papadokostaki, George Mastorakis, Spyros Panagiotakis, "Handling Big Data in the era of IoT", chapter contribution in the "Advances in Mobile Cloud Computing and Big Data in the 5G Era", Editors: Mavromoustakis Constandinos X., Mastorakis George, Dobre Ciprian, Springer-Verlag (2017), pp. 3-22.

[72] Maria Stoyanova, Yannis Nikoloudakis , Spyridon Panagiotakis , Evangelos Pallis, and Evangelos K. Markakis, "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues", accepted for publication in IEEE Communications Surveys & Tutorials, 2020.

[73] Konstantinos Karampidis, Spyros Panagiotakis, Manos Vasilakis, Evangelos K. Markakis, Giorgos Papadourakis, "Industrial CyberSecurity 4.0: Preparing the Operational Technicians for Industry 4.0", in proceedings of 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 11-13 Sept. 2019, Limassol, Cyprus.

[74] Information and Communication Technologies (ICT), https://searchcio.techtarget.com/definition/ICT-information-and-communications-technology-or-technologies, July 2020

[75] Internet of Things, https://en.wikipedia.org/wiki/Internet_of_things , July 2020

[76] Cyber-Physical Systems, https://en.wikipedia.org/wiki/Cyber-physical_system, July 2020