# Abstract

# Contents

[Abstract 1](#_Toc89136936)

[Chapter 1 – Introduction 3](#_Toc89136937)

[1.1 Background study 3](#_Toc89136938)

[1.2 Aim 3](#_Toc89136939)

[1.3 Objectives 3](#_Toc89136940)

[1.4 Research Qns 3](#_Toc89136941)

[1.5 Methodology 3](#_Toc89136942)

[1.6 Limitations 3](#_Toc89136943)

[Chapter 2 – Literature Review and Analysis – SDN Firewall 4](#_Toc89136944)

[2.1 Enterprise Networks 4](#_Toc89136945)

[2.2 Firewalls 4](#_Toc89136946)

[2.3 Software-Defined Networking 4](#_Toc89136947)

[2.3.1 Components of SDN Architecture 5](#_Toc89136948)

[2.3.2 SDN Controllers 6](#_Toc89136949)

[2.3.3 OpenFlow 7](#_Toc89136950)

[2.3.4 Switches 8](#_Toc89136951)

[2.3.5 Virtual Machines 8](#_Toc89136952)

[2.4 Vulnerabilities in SDN 9](#_Toc89136953)

[2.5 Threat Detection 10](#_Toc89136954)

[2.6 Chapter conclusion and research gap 10](#_Toc89136955)

[References 11](#_Toc89136956)

No table of figures entries found.

# Glossary of terms

**SDN – Software Defined Networking**

# Chapter 1 – Introduction

## 1.1 Background study

## 1.2 Aim

The aim is to simulate an SDN with firewall properties capable of thwarting DDoSNet attacks using machine learning or deep learning algorithms.

## 1.3 Objectives

The objectives include,

1. Learn and understand the elements in an SDN using a literature survey
2. Understand the threats in the environment and the latest advancements in the area
3. The use of ML/DL techniques in the process
4. Finding and understanding data sets in the process
5. Theoretical assessment on the deployability and limitations of the developed system.

## 1.4 Research Qns

The following are questions in the research,

1. How is SDN significant in network enhancement and increasing security?
2. What does AI mean in this field?
3. Is it possible to assess the gap between a deployable and experimental system?

## 1.5 Methodology

The research shall use Microsoft's team data science process in data process, algorithm design and deployability assessment. The detailed discussion is in the methodology chapter.

## 1.6 Limitations

The availability of a conclusive dataset for the experiment is a limitation. The collection of threat-related network data for the same is impossible due to the cross-sectional study timeline. It shall use only a theoretical assessment in the deployability of the model.

# Chapter 2 – Literature Review and Analysis – SDN Firewall

## 2.1 Enterprise Networks

## 2.2 Firewalls

## 2.3 Software-Defined Networking

According to Kirkpatrick, a Software-Defined Network (SDN) is a three-tier stacked architecture with an SDN controller sitting in the middle managing and directing data traffic, with two Application Programming Interfaces, namely the Northbound APIs and the Southbound APIs **(Kirkpatrick, 2013)**. All computer networks have three planes of functionality: the data, control, and management planes. The control plane maps network traffic, decides for the data plane in forwarding the traffic accordingly. Whereas the management tools mainly involve software services for monitoring and configuring control functions. In all traditional vertically integrated networks, there is a coupling in the data and the control planes, in contrast to SDN decoupling. The decoupling feature is the most powerful feature of the Software-Defined network. The separation of the control plane (the network's control logic) from the resident routers and switches forwarding the traffic (the data plane) makes the switching nodes become simple forwarding devices controlled logically by a centralised controller. A logically centralised controller model, however, should not be confused with a physically centralised controller model. Significant advantages of achieving this decoupling of the data and control planes include the desired flexibility, ease of development and deployment of new networking features, and network evolution and innovations scope. **(Kreutz et al., 2015) (Kirkpatrick, 2013) (Casado and McKeown, 2005)**.

According to Kreutz et al., there are three fundamental abstractions in the SDN concept, namely i) the forwarding abstraction, ii) the distribution abstraction, and iii) the specification abstraction. The function of the forwarding abstraction is to allow any desirable forwarding behaviour desired by the network application or the control program while hiding details of the hardware involved. On the other hand, the distribution abstraction functions as a shield for the various SDN applications from the vagaries associated with the distributed state by replacing the distributed control problem with a logically centralised controller. The feature requires a common distribution layer, should lie in an SDN's network operating system, install control commands on the forwarding devices, collect status information about the network devices and links in the forwarding layer, and offer a global network view to the network application. The last abstraction, the specification abstraction, is similar in function to the forwarding abstraction in that it should also allow any desirable network behaviour. However, it should not be responsible for implementing that behaviour itself. Network virtualisation solutions and network programming languages achieve this functionality in an SDN. These approaches relate the abstract configurations expressed by the network applications into the physical configuration abstraction of the global network view presented by the SDN controller. **(Kreutz et al., 2015) (Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**.

There are four significant features of an SDN network architecture,

1. The separation in data and control planes – with in-built controls removed, the routers, switches and other network devices will revert to be simple packet forwarding elements.
2. Flow-based – instead of being destination-based, the data forwarding decisions will be flow-based. In Software-Defined Networking, flow is the sequence of packets between the source and a destination.
3. Logical centralisation - the control logic shifts to a logically centralised SDN controller or a Network operating System (NOS), an external entity.
4. Programmed network – software applications running on the Network Operating System interact with the data plane devices and can be used to program the network, thus defining the characteristics of any Software-Defined Network. **(Kreutz et al. (Kirkpatrick, 2013) (Casado and McKeown, 2005)**

### 2.3.1 Components of SDN Architecture

The following are the different elements present in a typical SDN,

* Forwarding Devices (FD): Forwarding devices are either Hardware or Software-based data plane devices. They are used to take action on incoming data packets. They have well-defined instructions (flow rules) defined by the Southbound interfaces. The SDN controllers implement these Southbound protocols.
* Data Plane (DP): The interconnected forwarding devices (via wired or wireless media) together forms a network infrastructure referred to as the Data Plane.
* Southbound Interface (SI): The Southbound Application Programing Interfaces (Southbound APIs) defined the instructions for the forwarding devices part of the Southbound Interface. The southbound interface also defines the communication protocols between the control plane and the underlying forwarding devices.
* Control Plane (CP): Control plane acts as the central hub for all applications and controllers. Control plane elements abstract the programming of the forwarding devices in the network.
* Northbound Interface (NI): the Northbound Interface abstracts the low-level instruction sets used by Southbound interfaces in programming the forwarding devices. Northbound API refers to the familiar interfaces for developing applications used by the application developers.
* Management Plane (MP): The functions offered by the Northbound Interface to implement network control and operation logic leveraged by a set of applications makes up the management plane. Such applications include routing, firewalls, monitoring, load balancing etc. **(Kreutz et al., 2015) (Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**.

Diagram

Description automatically generated

Figure 1. A three-level logical abstraction structure of SDN. Source - (Kreutz et al., 2015)

### 2.3.2 SDN Controllers

An SDN controller sits in the middle of the SDN architecture and manages the data traffic. According to Nadeau and Gray, an SDN controller is a collection of physical or software systems that work up a network state management and distribution. These network systems might involve a database. The information controlled by the SDN Controller includes data on the network state, learned topology, configuration data, and control session data. In-memory database strategies may apply to manage this information, and the controller may have multiple purpose-driven data management processes. An SDN controller also provides a high-level data model to capture the relations between the services provided by the SDN controller, managed resources and policies. **(Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**

The SDN controller services are exposed to the applications using a Representational State Transfer (REST) API. The controller and its API together formulate a development environment to generate the API code. Additionally, in some cases, it may even allow expansion of core capabilities and subsequent publishing of APIs for new modules like those supporting the dynamic addition on the controller capabilities. The SDN controllers are also responsible for providing a secure TCP control session between the controller and the network elements for stateful provisioning in the network elements, such as topology and service discovery mechanisms, path computation systems and other network-centric or resource-centric information services. In a distributed control paradigm, the network information is constructed as part of the host build process and run as a host or SDN controller service. SDN controller thus becomes a critical management interface for software switches and routers on hosts in a data centre, also responsible for the associated states of their ephemeral network entities like analytics and event notification. **(Nadeau and Gray, 2013) (Casado and McKeown, 2005)** **(Kobayashi et al., 2014)** **(Kreutz et al., 2015)**.

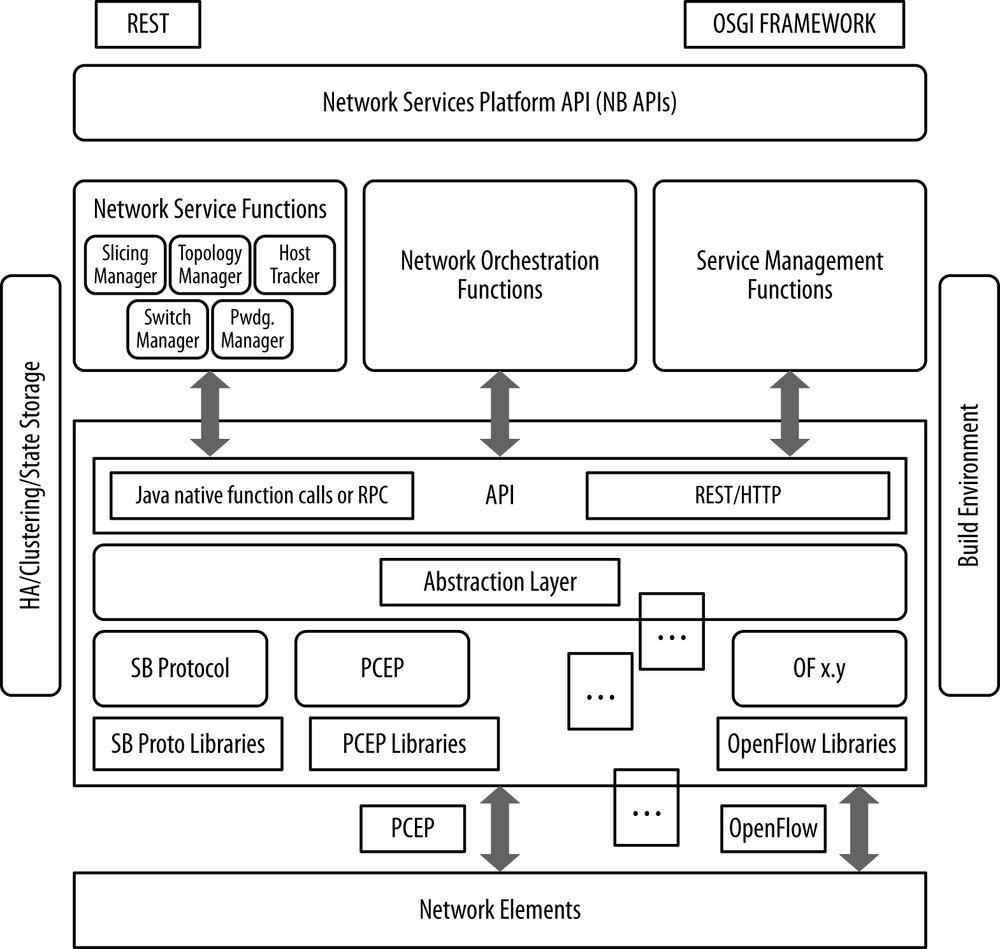


Figure 2. An ideal controller framework. Source -(Nadeau and Gray, 2013)

Ryu is a popular open-source Software-Defined Network controller, and a component-based framework implemented entirely in Python, supporting components developed in other languages. Features of a Ryu controller includes an OpenFlow wire protocol support, a series of reusable libraries, application management, event management, and infrastructure services. Additionally, applications like Snort (a layer-2 switch), Virtual Router Redundancy Protocol (VRRP), GRE tunnel abstractions and topology and statistics services are available in Ryu controllers. Ryu also supports a REST interface for its OpenFlow operations and has an Open stack Quantum plug-in to support GRE overlay and VLAN. **(Nadeau and Gray, 2013)** **(Nippon Telegraph and Telephone Corporation, 2011)**.

### 2.3.3 OpenFlow

Southbound Interfaces or APIs connect links between control and forwarding elements, and OpenFlow is the widely deployed open southbound standard for SDN. OpenFlow is a communication protocol and not an API and allows the user to communicate between the SDN controllers and the forward plane of network devices, which direct traffic through the switches and routers from various vendors and manage them. OpenFlow remotely controls the access points, network switches and routers and provides an open interface for these networking nodes. There are two divisions in OpenFlow protocols, namely a Wire protocol and a Configuration and management protocol. The wire protocol establishes a control session, define the fundamental structure of switches, and message structures for exchanging flow mods and collecting statistics. However, a Configuration and management protocol allocates physical switch ports to a particular controller and ensures high availability and behaviours upon controller connection failure. OpenFlow protocols cannot directly provide network slicing, i.e., dividing an element into separately controlled groups of ports or a network into separate administrative domains. Additional tools like Flow Visor and specific vendor implementations can achieve this function. OpenFlow rules can also define various firewall applications. **(Nadeau and Gray, 2013)** **(Casado and McKeown, 2005)** **(Das and Mckeown, 2012)** **(Kobayashi et al., 2014)**.

### 2.3.4 Switches, virtual machines and virtual switches

All internetworking between LANs uses LAN switches. Layer 2 switches or Bridges filter and forward data packets and learn solely based on the destination locations and source addresses. As long as the network supports unique addressing, they can be interconnected using bridges. The bridge does not have any privileges on its attached Local area networks (LANs) and should not violate any access control rules. Therefore, a bridge becomes a networking device with many ports running an application to transparently forward data among those ports. A layer-3 switch or a routing switch is a networking switch used for forwarding data packets from one network to another network or device. A layer-3 switch can do all the traditional functions of a Layer-2 switch and perform additional tasks like static and dynamic routing. Multi-layer Switch (MLS) refers to networking devices that can fulfil all Layer-2 and Layer 3 functionalities and some Layer-4 features. **(Seifert and Edwards, 2008)** **(Casado and McKeown, 2005)** **(Das and Mckeown, 2012)** **(Kobayashi et al., 2014)**.

Virtual Machines (VMs) are logical machines comparable to a real host machine with an operating system. A VM system comprises multiple virtual machines, running operating systems in a single physical host machine. The software for hosting and managing all the VMs in a Virtual Machine system is the Virtualisation layer. Virtualisation, Resource scheduling, Migration, Security and Performance Evaluation are the critical technologies of VM Systems. Network virtualisation is the abstracting of traditional hardware to software delivered network resources. Migration refers to transferring one resource in a network to another location or network, i.e., making the VM, the operating system or memory transferable. Resource Scheduling refers to scheduling system resources for different tasks by using different resource scheduling algorithms. As VMs are more prone to attacks, the security expectations are higher in Virtual machine systems. The Performance Evaluation is achieved by a lot of different network parameters and methods. Although current performing monitoring methods can monitor and predict the behaviour and performance of a single VM, it becomes challenging in cases of multiple VMs in a virtual machine system. **(Li, Li and Jiang, 2010)**

Studies say virtual machines (VMs) running on the same host system need interconnection and physical network connections. For switching between VMs, software switches, called virtual switches, thus perform VM bridging, providing additional features apart from traditional switching features. The lowest common denominator in a networking environment is the Virtual switches or routers. In a Virtual LAN, the Virtual switches facilitate the forwarding of a small quantity of data packets compared to physical switches and routers **(Nadeau and Gray, 2013) (Emmerich et al., 2014) (Li, Li and Jiang, 2010)** **(Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**.

## 2.4 Vulnerabilities in SDN

According to Zerkane et al., a Security vulnerability is a weakness in the network, if exploited, will transform the same into a flawed system. A breach in the network system security will expose additional weaknesses and violate its security characteristics, and identifying and eliminating vulnerabilities in any system becomes crucial to data and operations. **(Zerkane et al., 2016) (Kreutz et al., 2015)**.

Any possible security vulnerabilities in a Software-Defined Network occur under one of the four layers of an SDN architecture – 1) the Application layer, 2) the Control layer, 3) the Data-Plane layer, and 4) the Management layer. Considering the attacks on these four layers, the attacks on the assets located in the Data-Plane and Application layers primarily come from the network and result from user interactions or external processes. But, the attacks on the Control layer does not require user interactions to succeed and may come from adjacent neighbours or a limited vector. All issues in these four layers fall into seven categories, namely

1. Unauthorised Access
2. Data Leakage
3. Data Modification
4. Malicious/Compromised Application,
5. Denial of Service
6. Configuration Issues, and
7. System-level Issues

The pathway by which the attacker exploits vulnerabilities in any form is called the attack vector, classifiable based on the attacker location. If it is an external location, then the attack is Remote, and if it is from a neighbouring network, which shares the same infrastructure with the enterprise, it classifies as Adjacent. A third scenario is when the attacker exploits the vulnerabilities within the SDN network; it becomes Local. Additionally, if the attacker is physically accessing the material, the attack vector will be considered physical. Finally, complexity refers to the conditions and efforts required by the attacker to exploit vulnerabilities in a system, which is in a low state if the attacker can identify the network state in a few steps just by going through SDN specifications and standards; the reverse is valid for high complexity. **(Zerkane et al., 2016) (Kreutz et al., 2015)**.

## 2.5 Threat Detection

## 2.6 Chapter conclusion and research gap

# Chapter 3 Methodology and data analysis

The process used to design the project is Microsoft's TDSP framework – Team DataScience Process.

# Diagram, schematic Description automatically generated

Figure 3. The TDSP lifecycle. Source - (marktab et al., 2021)

# Chapter 4 Artefact and discussion

# Chapter 5 Conclusion and future direction

# References

Emmerich, P., Raumer, D., Wohlfart, F. and Carle, G. (2014). Performance Characteristics of Virtual Switching. *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. [online] Available at: https://ieeexplore.ieee.org/document/6968979 [Accessed Nov. 2021].

Kirkpatrick, K. (2013). Software-defined Networking. *Communications of the ACM*, [online] 56(9), p.16. Available at: https://dl.acm.org/doi/10.1145/2500468.2500473 [Accessed Nov. 2021].

Kreutz, D., Ramos, F.M.V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S. and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, [online] 103(1), pp.14–76. Available at: https://ieeexplore.ieee.org/document/6994333 [Accessed Nov. 2021].

Li, Y., Li, W. and Jiang, C. (2010). A Survey of Virtual Machine System: Current Technology and Future Trends. *2010 Third International Symposium on Electronic Commerce and Security*. [online] Available at: https://kevincurran.org/com320/papers/VmsPaper.pdf [Accessed Nov. 2021].

Nadeau, T.D. and Gray, K. (2013). *SDN: software defined networks*. 1st ed. O'Reilly Media Inc.

Seifert, R. and Edwards, J. (2008). *The all-new Switch Book: the Complete Guide to LAN Switching Technology*. Indianapolis: Wiley Pub.

Zerkane, S., Espes, D., Le Parc, P. and Cuppens, F. (2016). Vulnerability Analysis of Software Defined Networking. *Foundations and Practice of Security*, [online] pp.97–116. Available at: https://www.researchgate.net/publication/311961662\_Vulnerability\_Analysis\_of\_Software\_Defined\_Networking [Accessed Nov. 2021].

Casado, M. and McKeown, N. (2005). The virtual network system. *ACM SIGCSE Bulletin*, [online] 37(1), pp.76–80. Available at: http://yuba.stanford.edu/~casado/vns\_sigcse.pdf [Accessed Nov. 2021].

Das, S. and Mckeown, N. (2012). Why OpenFlow/SDN Can Succeed Where GMPLS Failed. *Stanford database*, [online] 2012(2012). Available at: http://yuba.stanford.edu/~nickm/papers/ECEOC-2012-Tu.1.D.1.pdf [Accessed Nov. 2021].

Kobayashi, M., Seetharaman, S., Parulkar, G., Appenzeller, G., Little, J., van Reijendam, J., Weissmann, P. and McKeown, N. (2014). Maturing of OpenFlow and Software-defined Networking through deployments. *Computer Networks*, [online] 61(August), pp.151–160. Available at: http://yuba.stanford.edu/~nickm/papers/openflow\_deployment\_journal\_paper\_aug2012.pdf [Accessed Nov. 2021].

Nippon Telegraph and Telephone Corporation (2011). Welcome to RYU the Network Operating System(NOS) — Ryu 4.34 documentation. *ryu.readthedocs.io*, [online] 2011(537f35f4). Available at: https://ryu.readthedocs.io/en/latest/index.html [Accessed Nov. 2021].

marktab, Alexbikgit, v-kents and microsoft (2021). What is the Team Data Science Process? - Azure Architecture Center. *docs.microsoft.com*, [online] 2021(11). Available at: https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview [Accessed Nov. 2021].