SDN BASED SECURITY SOLUTIONS FOR MULTI-TENANCY NFV

Lejaha Retselisitsoe

This Dissertation is submitted in partial fulfilment of the academic requirements

for the Degree of

Master Engineering in Telecommunications

in the Faculty of Engineering and The Built Environment

University of Cape Town

October 2016

As the candidate's supervisor, I have approved this dissertation for submission.

Name: Ms. Joyce Mwangama

Signed by candidate

Date 01 November 2016

Declaration

I declare that the work presented in this dissertation is originally my own. Ideas and material generated from other researchers is explicitly stated with appropriate references. I confirm that my supervisor Ms Joyce Mwangama has reviewed and approved submission of this work.

This work is submitted for the Master of Engineering specialising in Telecommunication at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Name Retselisitsoe Lejaha

Signed by candidate

Date 31 October 2016

ABSTRACT

The Internet continues to expand drastically as a result of explosion of mobile devices, content, server virtualization, and advancement of cloud services. This increase has significantly changed traffic patterns within the enterprise data centres. Therefore, advanced technologies are needed to improve traditional network deployments to enable them to handle the changing network patterns. Software defined networks (SDN) and network function virtualisation (NFV) are innovative technologies that enable network flexibility, increase network and service agility, and support service-driven virtual networks using concepts of virtualisation and softwarisation. Collaboration of these two concepts enable cloud operator to offer network-as-a-service (NaaS) to multiple tenants in a data-centre deployment.

Despite the benefits brought by these technologies, they also bring along security challenges that need to be addressed and managed to ensure successful deployment and encourage faster adoption in industry. This dissertation proposes security solution based on tenant isolation, network access control (NAC) and network reconfiguration that can be implemented in NFV multi-tenant deployment to guarantee privacy and security of tenant functions.

The evaluation of the proof-of-concept framework proves that SDN based tenant isolation solution provides a high level of isolation in a multi-tenant NFV cloud. It also shows that the proposed network reconfiguration greatly reduces chances of an attacker correctly identifying location and IP addresses of tenant functions within the cloud environment. Because of resource limitation, the proposed NAC solution was not evaluated. The efficiency of this solution for multi-tenancy NFV has been added as part of future work.

Acknowledgement

I thank my supervisor, Ms J. Mwangama, for her encouragement, constructive criticism, direction and support in completing this research work. The experience of working with Ms Mwangama has always been delightful and enlightening. I feel privileged to have worked with her.

I also thank my caring and supportive family—especially my mother and my brother—for always uplifting my spirit and cheering me to work as hard as I could.

Finally, I thank the Almighty God for giving me strength and courage to never fear my weaknesses but learn from them to grow personally and professionally.

Table of Contents

List of Figures

List of Acronyms

AAA	Authentication, Authorisation and Accounting
AES	Advanced Encryption Standard
BGP-LS	Border Gateway Protocol-link State
CLI	Command Line Interface
DES	Data Encryption Standard
DHCP	Dynamic host configuration protocol
ETSI	European Telecommunication Standard Institute
ICMP	Internet Control Message Protocol
IT	Information Technology
IP	Internet Protocol
JVM	Java Virtual Machine
LISP	Locator ID Separation Protocol
MANO	Management and Orchestration
MTD	Moving Target Defence
NaaS	Network as a Service
NAC	Network Access Control
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualisation
NFVI	Network Function Virtual Infrastructure
NFVIaaS	Network Function Virtualisation Infrastructure as-a-service
ODL	OpenDayLight
ONF	Open Network Foundation
OSGi	Open Services Gateway initiative
OVS	Open Virtual Switch
OVSDB	Open vSwitch DataBase Management Protocol
pIF	Physical Interface
PoC	Proof of Concept
REST	Representational State Transfer
SAL	Service Abstraction Layer
SDN	Software Defend Network

vIF	Virtual Interface
rIP	Real IP address
vIP	Virtual IP address
VLAN	Virtual Local Area Network
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFaaS	Virtual Network Function as a Service
VNFM	Virtual Network Function Manager
VNPaaS	Virtual Network Platform as a Service
VTN	Virtual Tenant Network
VxLAN	Virtual eXtensible Local Area Network

CHAPTER 1

INTRODUCTION

1.1. Introduction

The Internet continues to expand in size as a result of rapid increase in mobile traffic, online services (search engines, social networks and web content in multiple data centres), higher data rates, server virtualisation and advent of cloud services. This increase has significantly changed traffic patterns within traditional network architectures [1]. While attempting to rapidly deploy services to meet the dynamic customer demands, the proprietary nature of hardware poses a huge challenge for network operators [2]. These network devices that are usually vendor locked offer dedicated services, which implies the need for additional nodes to introduce new customer services. This translates to cost of purchasing the nodes, additional floor space, power and need for specialised expertise to handle the increased operational complexity. The operator is thus faced with tedious error prone process of manually configuring the ever increasing number of devices. To cope with these demands, networks have to be much smarter and be able to control and manage themselves better [3].

Collaboration of academic and industry research has proposed innovative solutions that allow networks to dynamically adapt to fluctuating network demands. Network Function Virtualisation (NFV) and Software Defined Networks (SDN) are examples of such solutions. These are two independent technologies that are based on cloud computing technologies. They promise network flexibility, increased network and service agility, and support service-driven virtual networks using concepts of virtualisation and softwarisation [2].

SDN implements a network control plane that has a global view of the network and decision making capabilities in a logically centralised and fully-programmable software platform by decoupling it from the network forwarding devices [3]. The network programmability enables automation of network policies that define dynamic control, change, and management of network behaviour. The functionality is implemented from a single central point and can therefore be easily inherited by all the nodes in the network. This reduces administration tasks, deployment time and configuration errors. The rules can flexibly be upgraded or modified using high level programming languages to enhance network services based on customer and network requirements.

NFV on the other hand defines network functions as software instances that can be instantiated in open and standardised IT virtualization environment [4]. NFV enables deployment of multiple functions in one computing node therefore greatly reducing CAPEX and OPEX as compared to traditional networks where one device is mapped to only one network function. The software defined function can easily be modified to suit the required service and they can be instantiated or terminated on demand, hence enabling rapid service provisioning at a much reduced cost.

Logical integration of SDN and NFV enables cloud operators to provide a new cloud-based service model—Network-as-a-Service (NaaS) similar to the traditional cloud models Infrastructure as-a-service (IaaS), Platform as-a-service (PaaS) and software as-a-service (SaaS) [5]. This model allows operators to lease virtual resources that can be used to run software defined network functions for different tenants. SDN can be used to enable innovative, agile and simplified management of these virtual networks [6]. In such an environment the cloud operator has to guarantee privacy and security for the hosted tenant networks.

This integration introduces automation, simplifies monitoring and management, and enables networks to adapt to different traffic loads [2]. This dissertation proposes SDN based security solutions that can be implemented by cloud operator offering NaaS to multiple tenants in a data-centre. The solution implements tenant isolation, network access control (NAC) implemented by authentication, authorisation and accounting (AAA), and network reconfiguration that makes it hard to learn virtual network topologies and their properties. The proposed framework and solutions are implemented using components of two open-source tools—OpenDaylight (ODL) and OpenStack that enable collaboration of SDN and NFV.

1.2. Problem Statement

In the above discussed NaaS model, tenant networks will come and go, VNFs and service chaining will change frequently to balance load across the infrastructure [7]. This continuously changing environment makes network management more complex. NFV also inherits most of the cloud characteristics such as virtualisation, multi-tenancy, application sharing and open source software. Therefore, it adopts associated security threats like authentication, information leakage and data corruption [6]. This architecture allows for flexible and dynamic provisioning of virtual functions based on demand. In this type of deployment, tenants are bound to be concerned about data confidentiality, integrity and isolation from other tenants.

Cloud operators therefore need to come up with a convincing NAC solution to assure

tenants' security. One of the major security challenges as discussed in [5] and [10] is tenant authentication, authorization and accounting (AAA). For the service model considered in this work, each tenant is granted administrative rights to combine and configure VNFs to form a network service, software upgrades and function management. The tenant may also want to add or terminate functions into the architecture by migrating VMs in or out of the cloud environment. The tenant domain consumes resources from the cloud infrastructure using cloud orchestration tools to orchestrate and operate virtual infrastructure resources required by functions and associated service chains [4]. These resources are the same resource shared between all tenants. The cloud operator has to ensure that each tenant only has access to its hosted functions and cannot access colleague tenants data.

In a multi-tenant set-up, it is important to guarantee tenants complete isolation and privacy. The tenants hosted may sometimes even be competitors. The tenant should not be aware of the presence of other tenants and their functions or operations. Tenants may also at times attempt to utilize more than maximum allocated resource eating up those reserved for colleague tenants. Administrators also need to minimize risk of the VPN connection used by the tenants to access the cloud from being hijacked by an external attacker to access the cloud resource in the pretence of being the tenant.

Static assignment of IP addresses to network functions give attackers an asymmetric advantage in that they have the time to study a network, identify its vulnerabilities, and choose the time and place of attack to gain the maximum benefit [17]. Attackers normally use scanning tools and worms to send probes to random IP addresses in the network in order to discover their targets. When a target responds, it can then be identified and attacked. Unreachable addresses will therefore be considered unused. The static nature of current networks and computing systems makes it easy for them to be attacked and harder to defend [16].

The dynamic nature of the NFV cloud environment requires an automated, scalable, agile and flexible management mechanism. This is achievable by using the software defined networks (SDN) phenomena which enables definition of network rules using software. This rules can be used to implement NAC to regulate tenants requesting the cloud resources, virtual network isolation and network reconfiguration strategies.

1.3. Aim and Objectives

The aim of this work is to propose security measures to secure tenant networks in a multi-tenancy NFV deployment in a data-centre. One of the solutions implements a NAC that ensures

functions hosted by cloud service provider are manipulated by only authorized tenants. The solution introduces a policy based SDN application that allows the cloud operator to define security rules and to partition the cloud resources into logical security zones to achieve tenant isolation. It also proposes an SDN based network reconfiguration solution that will make the VTNs to dynamically change IP addresses and interfaces. This can be deployed by a cloud operator to minimise internal attack initiatives from other tenants and from external attacks. This aim can be achieved by the set of objectives below:

- To review security challenges introduced by NFV multi-tenant deployment and existing related SDN and NFV solutions.
- To propose and model security rules to regulate tenant access into the NFV cloud resources and authorisation of tenants to rightfully allocated resources.
- To design a framework that is able to partition the network into logically isolated zones to offer tenant isolation and privacy.
- To model a solution that pro-actively change the network configuration using MTD techniques to make virtual networks secure while keeping network functionality transparent.
- To implement and evaluate the proposed SDN based security solution for multi-tenancy NFV.

1.4. Scope

The solution proposed in this work is designed for SDN based multi-tenancy deployment of the NFV in a data-centre. This arrangement allows tenants to administer their virtual functions and service chaining. This dissertation focuses on tenant authentication, authorization and accounting for tenants accessing their respective control traffic in the NFV architecture. It also proposes logical security zones that ensure isolation and confidentiality between tenants hosted functions, therefore tenants will only be allowed to modify functions and manipulate service chaining in their respective predefined zones. It further defines network reconfiguration of VNF properties to make it tricky for attackers to identify their location and IP addresses. The proposed security solutions focus mainly on the following:

- Only manages access on the control plane and the user plane is left default as defined by the ETSI NFV.
- Tenant data isolation.
- Tenant authentication, authorization and accounting.
- Reconfiguration of IP address and interfaces of the VNFs in a data-centre.

1.5. Dissertation Outline

In this chapter, a concise introduction to the concept of network programmability and virtualization that enable cloud operators to offer NaaS to multiple tenant is made. The chapter further discusses security challenges faced by cloud operators offering this kind of deployment. The aims and objectives of this research work are also presented. The scope of the dissertation and an outline of this dissertation report is given in this chapter.

Chapter 2 discusses in detail the SDN and NFV technologies and their respective architectures. A literature review on the proposed security solution for multi-tenancy NFV is conducted by analysing previous approaches that tackle similar security challenges. Ideas from SDN based solutions proposed for traditional networks and cloud architectures are reviewed and used to model NAC solution for multi-tenancy NFV. Security solutions proposed for NFV are also reviewed. The idea of security zoning is proposed to built logical partitions between VTNs. The concept of Moving target defence (MTD) is also studied and the idea of network reconfiguration is extended to the NFV cloud deployment.

Chapter 3 presents the framework that is proposed for the implementation of multi-tenancy NFV in a cloud environment using SDN technology. The three main components of the framework —data plane cloud environment that provide virtual machines that hosts the VNFs, Control plane consisting of an SDN controller and a cloud orchestrator, and application plane built of software-defined network rules defined by the cloud operator. The chapter further explains in detail how the there planes interact with each other using APIs.

Chapter 4 presents a proof of concept (PoC) implementation of the proposed NaaS architecture from chapter 3. The chapter introduces the tools and technologies used to implement the SDN based multi-tenancy NFV framework. Components from two main open-source tools – OpenDaylight and Openstack are used for implementation of the PoC. A detailed design of how these tools and technologies can be set-up used to implement the PoC is also presented. The chapter also discusses in detail the proposed security challenges.

Chapter 5 evaluates the implemented NFV multi-tenancy framework and analyses the efficiency of the proposed security solutions. An analysis is made on the isolation mechanism proposed between VTNs residing in one physical compute node. A test scenario is presented and analysed to justify the logical separation between the VTNs. The effectiveness of the network reconfiguration concept is also analysed and evaluated based on three matrices: deception, deterrence and length of flow table against scanning tools that are normally used by attackers to probe a network for vulnerabilities.

In chapter 6, a conclusion for the entire dissertation is made. The technologies behind the problem statement have been introduced. Past solutions based on SDN and NFV have been reviewed and analysed against the proposed security solution in this work. An SDN based framework that enables multi-tenancy NFV deployment has been presented. The PoC for the proposed architecture is also presented that illustrates how Openstack and OpenDaylight were used to implement the proposed framework. The efficiency of the proposed security solution and implemented SDN based multi-tenancy NFV deployment is evaluated and analysed in this work. This dissertation illustrate the benefits of integrating an SDN controller into the NFV architecture to simplify implementation and management of security in the fluctuating NFV environment.

1.6. Chapter Summary

Chapter 1 gives a brief introduction of the technologies on which this dissertation is based—NFV and SDN. It discusses how integration of these two enables cloud operators to offer NaaS to multiple tenants. A discussion on the security challenges faced by operators offering this model is also made. The chapter further explicitly defines how this dissertation aims to solve these challenges in section 1.3 by presenting the aims and objectives of the dissertation. The scope covered by this work is presented, as well as the outline of this dissertation report.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

This chapter presents in detail the technologies on which this work is based—Software defined networks (SDN) and Network function virtualisation (NFV) – and studies how the two can be used to complement each other to offer Network as-a-service (NaaS) to multiple tenants. Security solutions that have been proposed for NFV in the past are revised and their strengths and weaknesses identified. A review on SDN based security solutions proposed for traditional networks and cloud architectures is also made and some of the ideas are used to model a network access control (NAC) solution and define logical partitions for multiple virtual tenants deployed in one compute environment. The moving target defence (MTD) concept is introduced and past solutions using this technique are also reviewed. From this, a network reconfiguration concept is extended to the NFV cloud that will dynamically change IP addresses and interfaces of the network functions in the NFV cloud deployment. The security solutions proposed in this dissertation are therefore implemented as SDN application managing an NFV cloud environment.

2.2. Network Function Virtualisation

2.2.1 NFV Architecture

NFV is an initiative defined by European Telecommunication standards Institute (ETSI) to address the operational complexities and high costs of managing the closed and proprietary devices deployed throughout Telecom networks, with the aim to enable agile deployment of service at a reduced cost [4]. NFV defines virtualised infrastructure for network functions using cloud technologies [5]. It decouples the network functions from underlying hardware so they run as software instances in any compute node. This technology enables efficient utilisation of resources by allowing different functions to share computing resources (storage, CPU and network). It also improve network flexibility by enabling instantiation of functions on demand and rapid service provisioning based on software deployments.

The NFV functions, named virtual network functions (VNFs) can be implemented as virtual machines (VM), and can therefore be accessed like Information Technology (IT) services as benefited from cloud computing technologies [4]. The VNFs can be dynamically added and

removed on-demand reducing administration tasks, response times and costs. Figure 2.1 below shows the NFV architecture as discussed by ETSI draft document in [3]. This NFV framework has three main working domains:

- Network Function Virtual Infrastructure (NFVI): includes hardware servers, the virtual layer which can integrate and virtualise the hardware resources (compute, storage, and network) and the abstract virtual resources.
- Virtual Network Function (VNF) : the software images that are capable of running over the NFVI.
- Management and Orchestration (MANO): built of an Orchestrator, Virtual Network Function Manager (VNFM) and Virtual Infrastructure Manager (VIM) functional blocks. The two main responsibilities of the MANO are orchestration of NFVI resources across multiple VIMs and management of Network Services. The VNFM is responsible for VNF life-cycle management. The VIM is in charge of controlling and managing the NFVI resources.

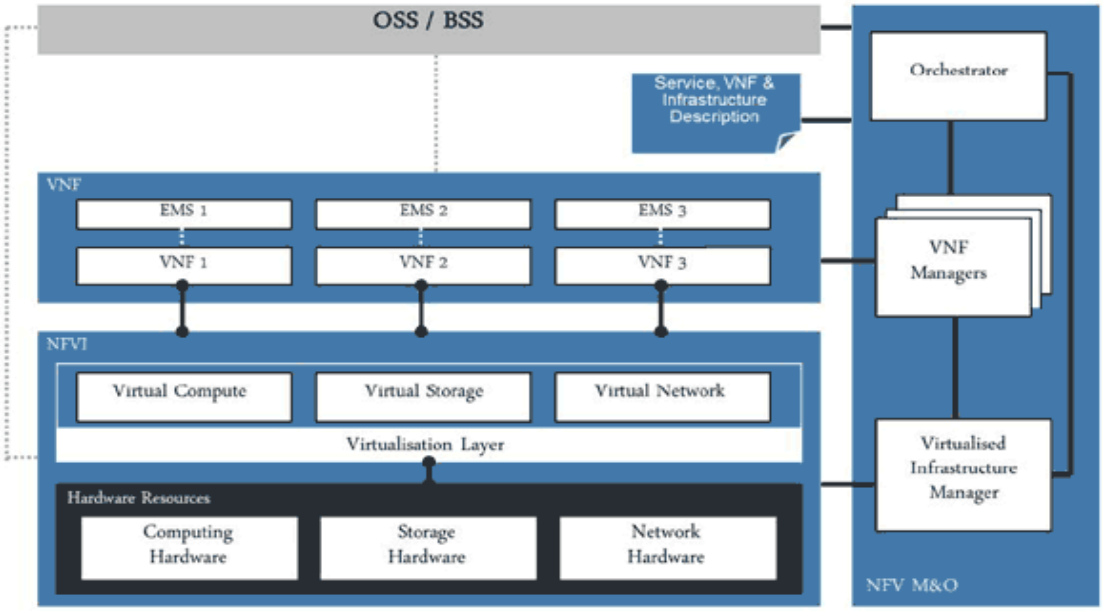


Figure 2. 1 NFV architecture[6]

2.2.2. Multi-tenancy NFV

The concept of NFV introduces new business models for cloud service providers. Operators can provide network-as-a-service (NaaS) to multiple tenants in a data centre deployment by leasing VMs that can be used to run the VNFs [7]. In multi-tenancy, operators have to implement logical separation of resources residing in the same physical infrastructure to guarantee tenant isolation and

privacy. The resources offered must provide security and isolation of sensitive data and applications in tenant networks. Therefore, cloud operators need to find smart management solutions for seamless multi-tenancy deployments in data centres [4]. Cloud services providers can deploy NFV in either one of the following service model [7]:

- Network Function Virtualisation Infrastructure as a service (NFVIaaS): allows enterprises to host VNF instances inside the cloud infrastructure which is operated as a service by a Cloud Service Provider.
- Virtual Network Platform as a service (VNPaaS): operator provides infrastructure and applications as a platform on which the Enterprise can deploy and manipulate their functions.
- Virtual Network Function as a service (VNFaaS): operator administered networking features are provided as a service to tenants.

2.2.3. NFV security solutions

This section presents solution that have been proposed in the past to secure the NFV environment. B. Jaeger [19] and B. Cataldo et.al [20] presents NFV security solutions that can be incorporated into the ETSI NFV architecture. Bernd proposes a security Orchestrator. This Orchestrator is responsible for the management of security of the entire architecture and the security functions deployed within the architecture. [20] presents a policy manager that is also incorporated into the NFV architecture. The policy manager translates users' High Level Policy defined using sentences close to natural language. The policy specify security requirements and defines processes for deployment and configuration of security functions. The solution in this work proposes security functionality that can be implemented as an SDN application, with the controller incorporated into the NFV architecture. The solution is based on SDN to allow fast and easy modification of the policies based on tenant or changing security requirements.

Security zoning has been discussed in [20], [21], and [22] as solution for tenant isolation in multi-tenant NFV deployment. It suggests partitioning of the architecture into logical segments to isolate and protect virtual tenant networks. R. Mijumbi et al. [22] further explains that management of this zones can be deployed using SDN to allow automation and easy update of security rules for the zones. From these ideas, the proposed solution defines the logical security zones to provide complete privacy and isolation of tenant networks. Figure 2.2 below illustrates the concept of zoning based of three datacenter deployment.

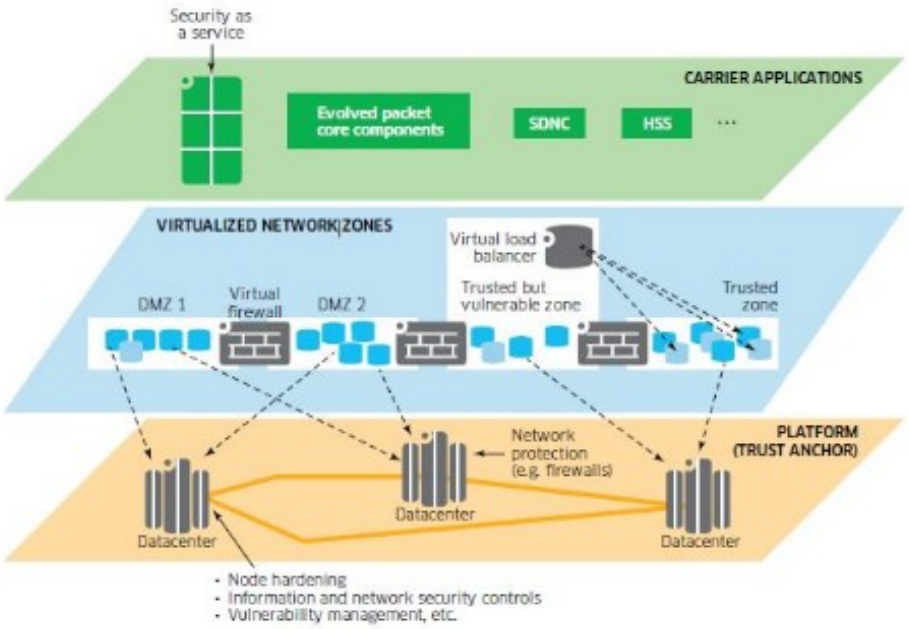


Figure 2. 2 Security zoning [19]

As discussed in the ETSI NFV problem statement [8] the NFV inherits most of the multi-tenant cloud deployment models. Solutions proposed for the cloud security can also be extended to NFV. D. Zissis and D. Lekkas in [23] presents a solution that proposes tenant authentication, authorisation and accounting (AAA) by introducing a trusted third party that will validate tenants security certificates before allowing them through to the cloud. Cloud security solutions presented in [24] and [25] suggests another security measure for cloud platform architecture. They proposes use of IPSec based Virtual private networks (VPN) to secure link between cloud and enterprise sites. The NFV solution proposed by this work also uses the idea of digital certificates together with IPSec based VPN to validate tenants into the NFV cloud. Unlike in [23], here the solution is based within the cloud infrastructure which reduces cost of paying a third party, increases agility that allows cloud administrator to update and modify the security rules, and reduces the security target surface by decreasing connection route.

2.3. Software Defined Networks (SDN)

2.3.1. SDN Architecture

SDN is another innovative technology defined by Open Network Foundation (ONF) with closely related objectives to those of the NFV [2]. It proposes a centralised architecture that decouples control plane from data plane and introduces network programmability, which makes it possible to dynamically control, change, and manage network behaviour through software, providing programmable interfaces into the network [4]. SDN architecture consists of three planes:

application, control and data as illustrated in Figure 2.3 below. The application layer consists of customised network applications that specifies data management rules and upgrade logic to the controller. Northbound APIs (NB-API) are used for communication between these two planes. The control plane consists of one or more centralised controllers. The controller uses the logic from the smart application to make forwarding decisions in the data plane. The data plane consists of forwarding devices. Southbound API is used to enable communication between the data and control planes. Below are examples of southbound APIs that exist [8]:

- OpenFlow Protocol: Most popular southbound protocol defined by ONF for used to transport a control logic function from a switch to a controller.
- Open vSwitch DataBase Management Protocol (OVSDB): Management protocol used by OpenvSwitch open source software switch.
- Locator ID Separation Protocol (LISP): It uses IP address and location framework suitable for overlay network applications, such as data centre network virtualization.
- Network Configuration Protocol (NETCONF): The protocol is used for networking devices configuration.

This evolution makes it possible to intelligently and flexibly control traffic flows to achieve best network throughput, perform traffic monitoring and analysis, and smart intrusion detection using customised user algorithms [2].

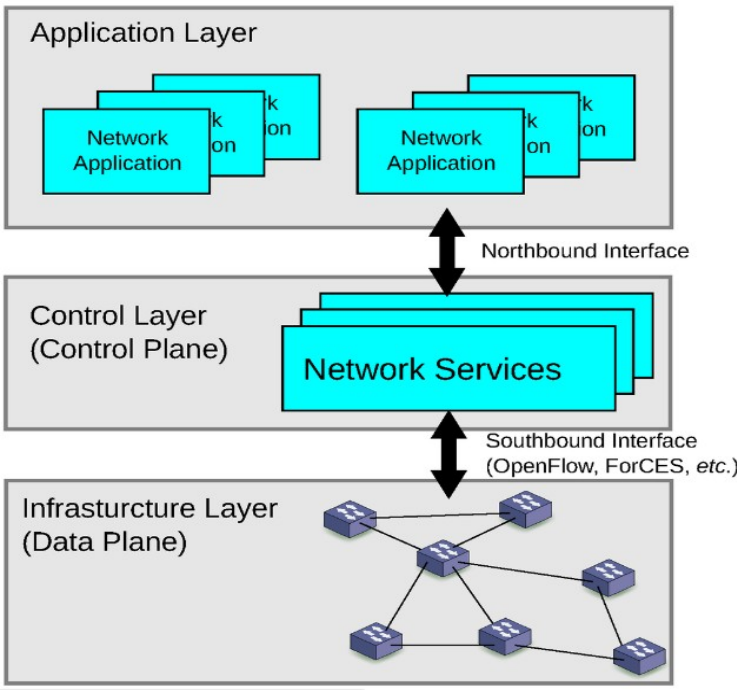


Figure 2. 3 SDN architecture[12]

SDN controller can be integrated into the NFV architecture to simplify monitoring and management by offering a scalable, elastic, automated and on-demand architecture necessary for the dynamic NFV communications [2, 9]. This work illustrates the benefits of using collaboration of SDN and NFV for implementation of NAC, tenant isolation and network reconfiguration. The implementation plan discussed later proposes the use of a JAVA based open-source tool—OpenDayLight and OpenStack for execution of the design. The cloud operator can add, manipulate and update SDN applications within the controller as need arises. The central controller defines the rules once for the entire data-centre multi-tenancy NFV deployment. This unique feature of SDN helps to avoid multiple definitions, contradictory decisions and to assure the coherence in the definition [10].

2.3.2. OpenFlow Protocol

OpenFlow is the most common communications interface defined between the control and forwarding layers of an SDN architecture [2]. It has been standardized by the The Open Networking Foundation. The OpenFlow architecture consist of the OpenFlow switches, OpenFlow controller and a dedicated channel connecting the switches and the controller. The switch functionality is executed on basis of a table called the flow table. The flow tables composed of headers- used to identify packets, counter – incremented each time the flow is executed and the actions- give instruction on how to handle packet [8]. The controller installs flow table entries in switches, so that these switches can forward traffic according to these entries. By insertion, modification and removal of flow entries, the controller can modify the behaviour of the switches with regard to forwarding.

Three classes of communication exist in the OpenFlow protocol: controller-to-switch, asynchronous and symmetric communication. The controller-to-switch communication is responsible for feature detection, configuration, programming the switch and information retrieval [6]. Asynchronous communication is initiated by the switch without any solicitation from the controller and it is used to inform the controller about packet arrivals, state changes at the switch and errors. Finally, symmetric messages are sent from either side, i.e., the switch or the controller are free to initiate the communication without solicitation from the other side.

2.3.3. Open Virtual switches (OVS)

The ONF white paper in [8] explain that SDN enables network functions to be programmed through flow-based software switches and OpenFlow protocol. Open vSwitches (OVS) defined by the Linux foundation and OpenFlow Switch (OFS) from the ONF are examples of these software switches that support fine-grained and flow-level control for packet switching [28]. With the help of

the central controller, all OpenFlow-based switches can be monitored, manipulated and managed. The NFV cloud environment proposed in this dissertation is implemented using OVS in the data plane. Figure 2.4 below shows an OVS, some of the features it provides and enabling protocols. Tenant VNFs are implemented as software inside OVS VMs. An SDN and Openstack controllers use OpenFlow and OVSDB management protocol to manage and manipulate the OVS traffic flows. The controller uses the concept of VLAN isolation to implement isolation and privacy of different tenant virtual networks.

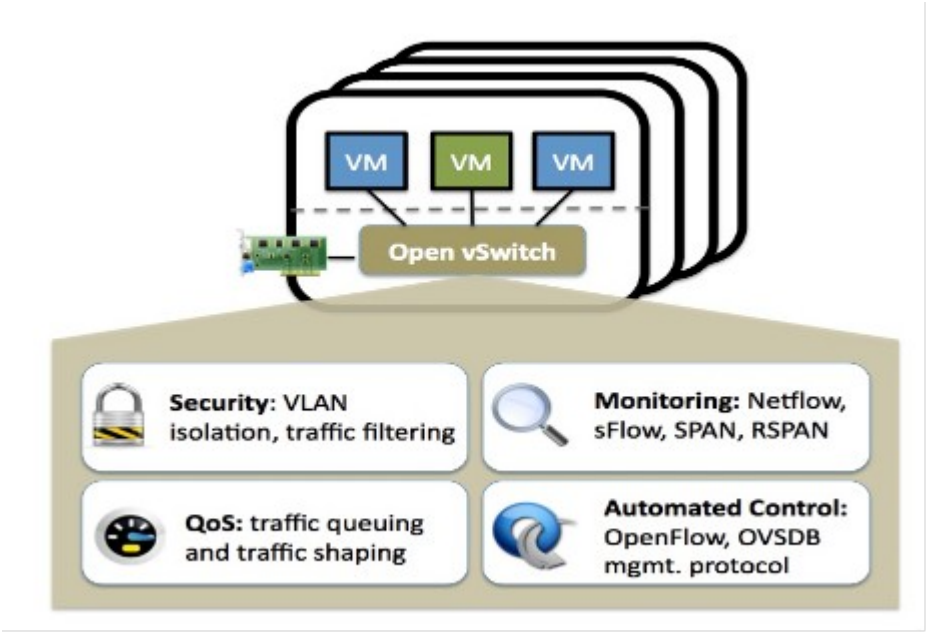


Figure 2. 4 Open vSwitch [28]

2.3.4. SDN based Security Solutions

SDN has been used in the past to implement NAC solutions for traditional networks. The solution FlowNAC discussed in [7] is one such example that presents an SDN based user plane traffic management solution that authorises users to access network resources according to target service. It pro-actively defines all services provided by the network operator as flows. It uses the concept of Virtual Local Area Networks (VLANs) to logically separate network resources. The solution presented in this work extends this idea to the NFV architecture. The proposed deployment architecture pro-actively defines logical network partitions for which only permitted tenants can access.

FLOWGUARD solution presented in [26] proposes an OpenFlow-based framework that enables detection and resolution of firewall policy violations in traditional networks. This solution implements a firewall functionality for the dynamic nature of SDN based solution in which network

state and traffic flow change frequently. During the flow update, the FLOWGUARD checks network flow path spaces to check any security policy violations. It also defines intelligent resolution techniques designed for different update situation to provide violation resolutions in real-time. This solution demonstrates how SDN introduces significant granularity, visibility and flexibility to implementing security policies in the network.

Vmware white paper in [27] presents vCloud Networking and Security Services that implement software-defined firewall, virtual private network (VPN), load balancing, network address translation (NAT), Dynamic host configuration protocol (DHCP) and virtual eXtensible local area network (VxLAN) functionalities for cloud environments. The solution has multiple virtual network interfaces that gives network operators flexibility and confidence to design and implement business-critical applications, build secure and agile private clouds, and protect their virtual desktop solutions [26].

24. Moving Target Defence (MTD)

2.4.1. Introduction to MTD

Before attackers initiate attacks they normally first learn the network topology and its properties using scanning tools. These freely available tools use Internet Control Message Protocol (ICMP) such as ping and trace route functionalities. The static nature of current networks and computing systems give the attackers an asymmetric advantage for the attackers and makes it easy for them to be probed and identified but harder to defend [14].

The concept of Moving Target Defence (MTD) is to impose the same asymmetric disadvantage on attackers by making systems dynamic and therefore harder to explore and predict [15]. With a constantly changing system and its ever adapting attack surface, attackers will have to deal with a great deal of uncertainty just like network administrators do. This concept is one of the solutions proposed by the United States Deputy Secretary of Defence report of 2010 addressing threats in the cyberspace domain. The solution was proposed as part of the Operation Buckshot Yankee counter-measure after an incident of malware attack of that spread of malicious computer code across the U.S. military’s computer networks [17]. The attack was from a foreign intelligence service that resulted in information leak. The ultimate goal of MTD is to increase the attacker’s workload so as to level the cybersecurity playing field for both defenders and attackers, leading to even tilting it in favour of the defender [16].

MTD techniques can be classified into three categorises: network based, host-based and instrument based. Network-based includes shuffling of IP addresses and dynamic access control [17]. Host-based techniques mutates the instructions of processes, code, address space

randomisation using cryptography. In Instruction based techniques network administrators can use honeypot techniques to capture new attacks and to dynamically change the IP addresses [18].

2.4.2. MTD past solutions

Moving Target Defence techniques have been proposed to increase complexity of learning and identifying network nodes and their vulnerabilities. This is achieved by dynamically changing network topologies and their attributes. Applications That Participate in Their Own Defense (APOD) scheme presented in [29] is an example of this approach. APOD implements address and port randomisation using hopping tunnels to hide identity of network nodes from sniffers. However, this approach requires cooperation between client and server during the mutation process. This disrupts active communication between the end hosts. S. Xu et al in [30] uses MTD to describe reactive adaptive network solution that adjusts security measures to control and have global security state. The solution only considers solution after an attack has been launched it is also important to defend networks to prevent the attacks from happening.

The network address space randomisation solution presented in [31] is an MTD solution proposing IP hopping technique that protects the network against hit-list worms. The solution uses network address randomization scheme implemented using DHCP update. However, during address hopping active communication within the network will be disrupted. This solution also requires change of host operating systems which makes it costly to deploy.

Self-shielding dynamic network architecture presented in [32] dynamically mutates appearance of network nodes by installing hypervisors in all network nodes. This approach protects honeynets from systematic mappings that identifies live and monitored IP addresses, and blacklist monitored IPs. This is archived by making the blacklist short-lived. The nodes retain their Operating system, applications and network communication flow. However, the hypervisor per host approached use here makes the solution expensive to implement.

Security solutions in [33] and [34] use the MTD techniques that allow mutation of host IP addresses to deceive external and internal scanners. These solutions are transparent because they retain network communication during the randomisation process. However, the IP mutation is performed at constant interval making it easy for attackers to learn the pattern and initiate malicious activities. In this dissertation the IP mutation is mutated at random intervals to improve effectiveness of the solution against biased scanning techniques.

2.4.3. SDN based network reconfiguration

The Open-Flow Random Host Mutation (OF-RHM) scheme proposed in [14] explains how varying IP addresses can protect and hide network nodes from worm propagation and scanning

tools. The solution is proposed for traditional networks and is implemented on the NOX controller by mapping to DNS records. The vIPs are assigned from the remaining unused IP addresses in a network address. This scheme offers transparent IP mutation while maintaining network configuration and minimizing operational overhead. The proposed solution in this work extents the idea from OF-RHM into the NFV architecture. Here, a JAVA controller—Opendaylight is used to transparently mutate vIPs of functions residing in VMs in a data-centre environment. Likewise the solution is evaluated for scanners inside and outside the data-centre.

S. Robertson et al. in [34] presents another SDN based MTD solution that transforms the static nature of networks into a dynamic environment that deceives attackers from correctly identifying network nodes. This solution offers transparent mutation that does not disrupt active communication to legitimate hosts. Authors also elaborate that the use of SDN enables cost-effective security deception solution. Likewise the solution in this dissertation takes advantage of this benefit attainable from the SDN technology to define security for NFV data-centre deployments.

The Honeynet solution presented in [35] proposes an SDN based collection of honeypots called HoneyMix. The HoneyMix attracts attackers to learn patterns, tactics, and behaviours. The programmability introduced by SDN enables the solution to intelligently circumvent attackers' detection mechanisms and manipulate data in the honeypots. The Mix sets up multiple connections to various Honeynets and decides the most desirable connection to retain attackers connection. This solution emphases the benefit of using SDN to manipulate data flows in the network.

2.5. SDN based solution for NFV

2.5.1. SDN based security solution for NFV

There exists a number of solutions that illustrate the benefits of integrating SDN and NFV to implement network security. Multistage OCDO in [36] is one such solution that illustrates how these two technologies can be used to efficiently manage and manipulate network security functionality in a cloud network. The solution partitions the network into different blocks and groups security chain between end-to-end nodes into different zones. These assignments are used to generate the availability of software defined security modules (SM) in the nodes of each zone in each block. The SMs are responsible for translating tenant's security requirements into different cloud zones. SDN enables optimal placement of software defined security functions per request in a cloud environment.

The SN-Security Architecture (SN-SECA) presented in [37] is an IEFT draft that presents architecture that aims to ensure effective security evaluation and integration on the SDN/NVF

deployment. This architecture shows how implementation of security solution between the control and data plane using IPSes, TLS or HTTPS can be achieved. The solution also proposes use of intermediary security devices such as IPS/IDS to improve security across the southbound communication. This work proves that SDN can be used to simplify and automate well-known security algorithms. Likewise SDN is used to implement known security solution to protect the NFV cloud deployment.

Collaboration of companies in the ICT industry have presented Verizon Network Infrastructure Planning that proposes SDN-NFV Reference Architecture that suggests how an SDN controller can be integrated into the ETSI NFV framework [38]. Authors here also explain that the concepts of network Virtualisation and programmability using cloud computing technologies enable telecommunications industry to simplify operations, administration, maintenance and provisioning of networks. This technology also promises to lead us into the next-generation networks which is 5G technology.

The ETSI NFV ISG is a body that defines requirements that network operators can adapt for their receptive environments. ONF presents [8] a similar architecture to that proposed by the Verizon. The framework illustrates how operators can combine NFV and SDN to achieve the common goals of both technologies. Figure 2.5 below shows functional blocks of the proposed model. The work presented by this dissertation uses a similar framework that incorporates an OpenDaylight SDN controller into an OpenStack based multi-tenancy NFV deployment.

Secure Data Centre for Enterprise solution presented in [39] is a software defined initiative defined by Cisco to provide secure and reliable provisioning of network services. Cisco define Cisco ASA 5585-X cluster that uses load balancing technique by using one device for monitoring and configuration of the firewall cluster. The cluster redirects the connection flows to a proper inspection manager process backing up each flow. The centralised control in this solution simplifies operations, increases availability, enhances security for Data centres. These benefits are attainable using centralised SDN controller.

NFV can be used to implement VNF that define security functionality. Authors in [40] propose a Deep Packet Inspection (DPI) VNF within a Software-Defined Infrastructure (SDI) that is responsible for detecting and preventing network intrusions. The DPI application was evaluated on a SAVI SDI test-bed to detect and block an attack or to re-direct it to a honey-pot for further analysis. This paper also discusses cost reduction bought by programmability of functions and management rules. This dissertation work takes advantage of these benefits hence the solution is based on SDN.

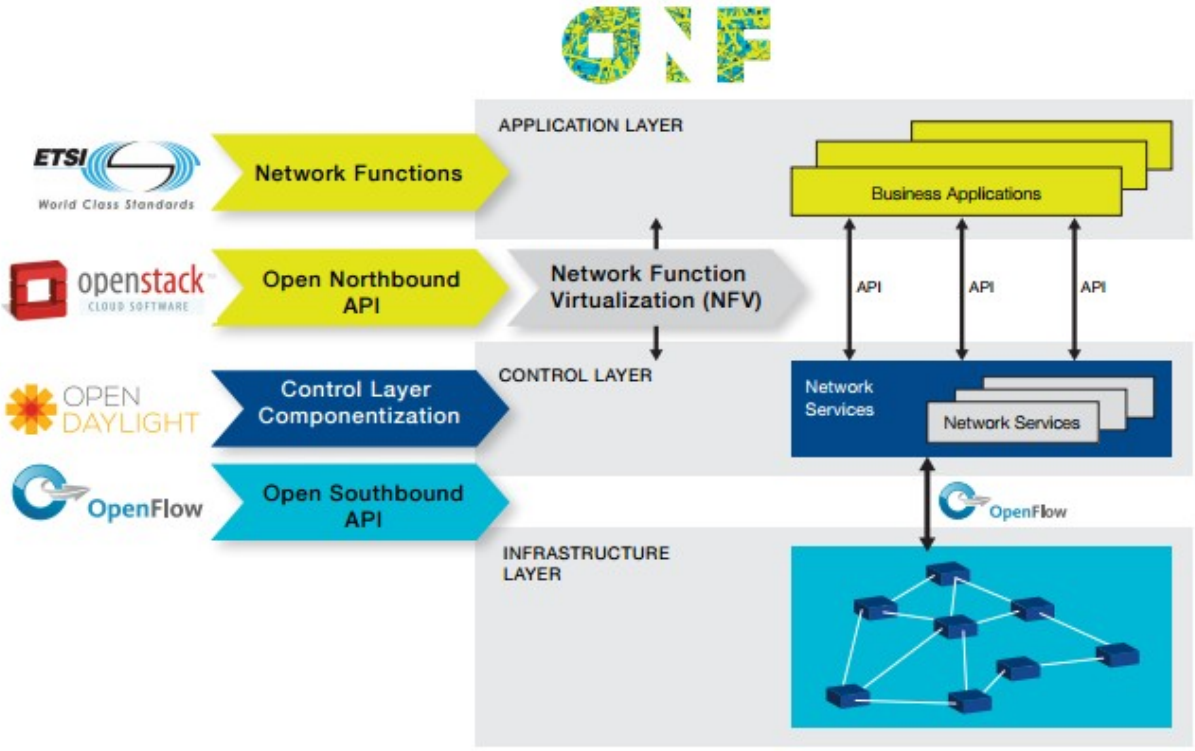


Figure 2. 5 NFV-SDN architecture

2.5.2. Benefits of Security based SDN

There are several features of the SDN technology that makes it a promising approach for implementing network security solutions. The SDN Security consideration in the Data-centre paper presented in [20] discusses attributes brought by SDN that can be used to implement secure and manageable cloud environment. ONF white paper [8] also clearly outlines the benefits of using SDN to secure networks. Below are some of the attractive features of SDN:

- Centralized control: allows for effective performance and threat monitoring across the entire network. networking devices from multiple vendors.
- Open-Source: this feature enables open innovation for network operators to customise traffic flows in their network based on specific network requirements rather than using vendor specific generalised devices and software.
- Agile service provisioning: the software based rule can easily be modified and updated to meet the ever evolving customer requirements.
- Programmability: this feature enables open innovation for operators, enterprises, independent software vendors and users to build custom-made traffic management depending on individual requirements.

- Improved automation and management: by using common APIs to abstract the underlying networking details from the orchestration and provisioning systems and applications.
- Increased network reliability and security: as a result of centralized and automated management of network devices, uniform policy enforcement, and fewer configuration errors.
- CAPEX and OPEX : automation of network rules makes service provisioning cheaper ad faster.
- End-user experience: as applications exploit centralized network state information to seamlessly adapt network behaviour to user needs.

NFV uses cloud technologies to deploy virtual networks in a Data-centre. The solution proposed by this dissertation takes advantage of the attributes above to implement security for virtual networks in a Data-centre.

2.6. Chapter Summary

This chapter studied in detail the concepts of SDN and NFV, and reviewed security solutions enabled by these technologies. One of the security solutions presented in this dissertation uses the ideas from SDN based solutions proposed for traditional networks and cloud architectures. The solution models NAC for multi-tenancy NFV. This work also adopts the concepts of security zoning and MTD solution that implements network reconfiguration. Unlike other security solutions proposed for NFV, the proposed solutions are based on SDN therefore allowing the Operator to upgrade and modify the security policies using software whenever needed. SDN enables flexible, elastic and scalable security management for the dynamic multi-tenancy NFV environment.

CHAPTER 3

SDN BASED MULTI-TENANCY NFV DEPLOYMENT

3.1. Introduction

The need by service providers to rapidly and cost-effectively deliver network services has led to the emergence of Network Functions Virtualization (NFV), which enables network functions to be deployed as software instances [4]. NFV promises scalability, flexibility and agility for service providers offering cloud-based network connectivity services to multiple tenants. Since virtualised network functions (VNFs) for different tenants in a multi-tenant virtualised network environment often share the same physical infrastructure, appropriate isolation, access rules and security need to be implemented to protect tenants from each other's malicious actions, which may be intentional or unintentional. This dissertation addresses the problem of tenant network isolation, network access control (NAC) and static nature of network topologies in a multi-tenant virtualised network environment. These solutions need to be defined and implemented within well-defined multi-tenant network functions virtualisation framework. This chapter, therefore, presents an elaborate multi-tenant network functions virtualisation framework, which uses SDN for automated control and efficient network policy implementation.

While NFV deals with virtualisation of network functions, SDN introduces programmability and automation of virtual network functions. The key feature of SDN is the decoupling of network control from forwarding planes [2]. SDN also provides a centralised view and control of the network, which can play a crucial role in achieving efficient orchestration and automation of the VNFs. The NFV framework presented in this chapter illustrates how NFV and SDN can be architecturally combined, thus converging to deliver a true 'softwarised' network services environment. The framework forms a basis for the implementation of tenant network isolation and security as presented in the next chapter.

The rest of the chapter is organised as follows. Section 3.2 presents an overview of the NFV framework. Section 3.3 explains virtual components that make up the NFV cloud (the data plane) of the proposed framework. Section 3.4 introduces the control plane and its interaction with the data and application planes. The application plane is presented in Section 3.5. Section 3.6 summarises the chapter.

3.2. SDN based NFV Framework

As highlighted in the previous sections, NFV separates network functions from the underlying hardware, thus running the functions as software images that do not depend on purpose-built hardware. Service providers can take advantage of this and move network components to a common physical infrastructure, which can significantly lower cost and shorten time to market. These network components can then be controlled and managed centrally and programmatically using the capabilities of SDN [3]. This dissertation proposes to combine NFV and SDN as illustrated by the framework depicted in Figure 3.1.

The figure shows several VNFs deployed in a cloud-like setting. VNFs that belong to different tenants can be implemented within the same physical infrastructure, thus enabling operators to rapidly and flexibly offer network-as-a-service (NaaS) to multiple tenants at low cost. The VNFs are in fact virtual machine (VM) instances deployed in a multi-tenant virtualised cloud environment. To accomplish efficient multi-tenancy, it is critical to ensure that there is no interference between the different tenants, as that can pose serious security risks for the tenants. To this end, this work comprehensively demonstrates in the next chapter how the tenants sharing the same cloud environment can be effectively isolated and secured from each other.

The proposed security mechanism is a set of regulatory and addressing policies that cloud operators can define to ward off intentional and unintentional actions that can compromise tenant security. The proposed security mechanisms and policies are implemented as SDN applications within a centralised control entity called an SDN controller. The controller in the framework in figure 3.1 sits right above the set of VNFs. The SDN controller has complete visibility of the VNFs, thus making it an appropriate point to initiate and propagate network policy rules. It is worth-noting that a single SDN controller can be a single point of failure.

To introduce more resiliency and availability, the active-standby SDN controller set-up can be implemented [1]. In this set-up, when the active SDN controller fails, an automatic fail-over is performed to hand over the control of VNFs to the standby controller. Another important layer in the SDN-NFV ecosystem is management and orchestration allowing for user defined policy definitions. The NFV cloud in the presented framework can be viewed as tiered model consisting of the data plane (VNFs), the control plane (SDN controller, management and orchestration) and the application plane (SDN rules that add logic to the control plane) respectively. These planes are discussed below. The framework in figure 3.1 share some similarities with the SDN-NFV Reference Architecture and ONF NFV-SDN architectures presented in section 2.5.

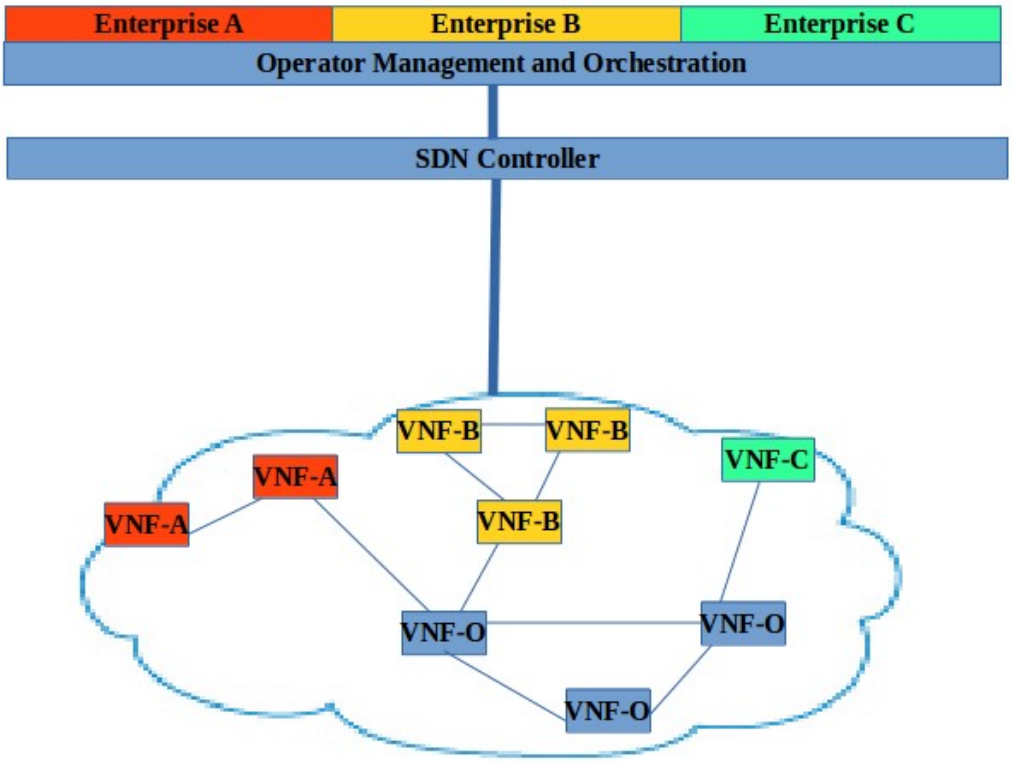


Figure 3. 1 VNF-SDN Multi-tenant Deployment

3.3 Data Plane in NFV-SDN Multi-tenant Deployment

The VNFs shown in the cloud environment depicted by figure 3.1 form the data plane of the proposed NFV-SDN multi-tenant framework. The data plane consists of virtual components—including a hypervisor—which is a software layer that enables simultaneous execution of multiple network operating systems (OS) in one compute node [45]. Each OS appears as a self-contained logical machine with independent processor, memory, network interface and other computing resources. The hypervisor is responsible for allocating physical resources amongst the different logical machines and ensuring that they do not disrupt each other. These logical machines are often termed virtual machines (VMs). The physical infrastructure that runs the VMs does not necessarily have to be purpose-built. Figure 3.2 illustrates the basic components of the data plane architecture providing VMs that can run software instances implementing network functions. Typically, the physical infrastructure belongs to the cloud-based network connectivity services provider. The provider leases out the VMs to multiple tenants to run their network functions.

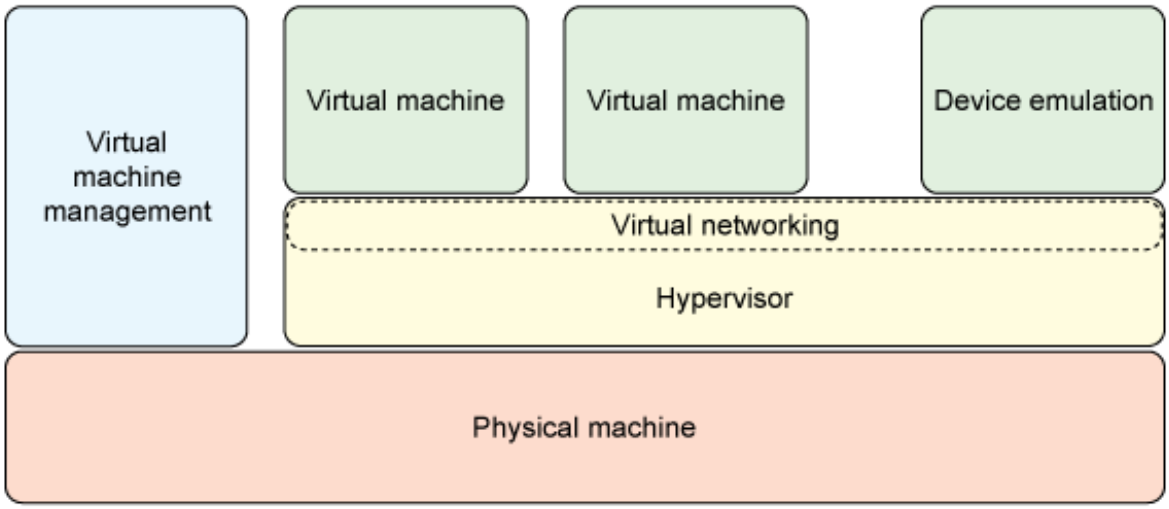


Figure 3. 2 Components of Cloud compute nodes

To optimize network communication among VMs, virtual switches can be introduced. For the framework presented in this chapter OpenFlow enabled switches—Open Virtual Switches (OVS) are used to facilitate communication between VNFs [28]. Figure 3.3. below presents an OVS and its interfaces. These OVSs are the resources which will be shared by VMs from different virtual tenant networks (VTN) to forward data between the functions. Figure 3.3. below shows the virtual interfaces (vIFs) associated with VMs communicated through the OVS to the physical interfaces (pIFs). The OVSs contains flow tables which define forwarding actions between VMs in one OVS using vIFs and between adjacent OVSs through the pIFs. The flow table is updated by the controller through the southbound interfaces.

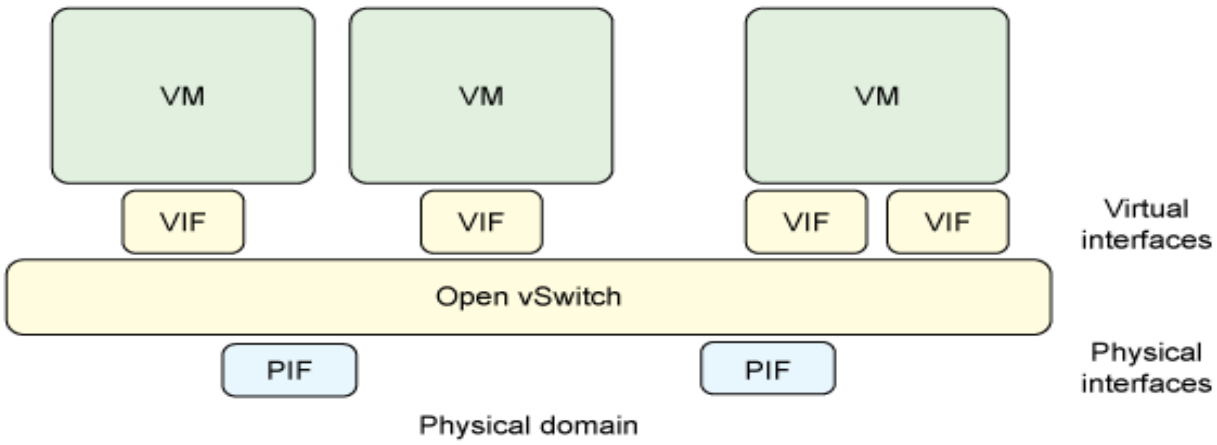


Figure 3. 3 Open vSwitch with virtual and physical interfaces

3.4 Control Plane in SDN-NFV Multi-tenant Deployment

The control and management plane of the proposed framework enables network virtualisation through SDN and NFV by providing simple abstractions of the underlying cloud environment. This plane is made of an SDN controller and a cloud orchestrator. The cloud orchestrator manages the virtual resources within the cloud platform hosting multiple virtual networks, while an SDN controller simplifies monitoring, management and manipulation of traffic in this environment using logic from network applications residing in the application plane. The controller offers a scalable, elastic, automated and on-demand architecture necessary for the dynamic NFV communications. This control plane uses north-bound APIs (NB-API) to communicate with the network applications and southbound APIs to communicate with the cloud environment.

3.5 Application Plane in SDN-NFV Multi-tenant Deployment

As discussed in section 2.2, the logic in the control plane is provided by the user defined network policies in the application plane. This work proposes security solutions that can be implemented as SDN applications allowing the cloud operator to protect the VTNs hosted in the cloud environment. One of these solutions implements logical separation to ensure complete privacy and isolation between different VTNs residing in one compute environment. The proposed idea aligns with security zoning discussed in the literature review. This proposition is similar to the concept of Virtual Local Area Networks (VLAN) deployment that is used in traditional networks, where users in different geographical areas can be brought together in one logical segmentation and can be granted access to predefined network resources. In a similar manner, the zones define virtual partitions for which tenants can be allowed to access VNFs that are not necessarily in adjacent VMs.

The controller also acts as a security guard monitoring and ensuring only authorised tenants are allowed into the cloud environment and that they access only their respectively designated logical zones. The authentication and authorisation can be implemented by the cloud orchestrator and, managed and monitored by the SDN controller. The controller takes note of all login attempts and reports all the failed attempts. This information can then be audited by the cloud operator to establish the source of any suspicious activities. This functionality enables the operator to timely notice threats and therefore implement preventive and reactive security measures. This dissertation also proposes a network reconfiguration SDN based application. The controller uses logic from this application to periodically re-assign IP addresses of the VMs hosting the VNFs from the unused address pool for each tenant network. The application also hops the vIFs of the VMs amongst

available OVSs in the cloud environment. The controller keeps record of all original IP addresses and allow only respective tenants to access the VNFs using the real IP addresses. This application translates actions defined for the original IPs and interfaces to the re-assigned values to enable transparent end-to-end communication. The application aims to decrease the chance of an attacker using scanning tools to correctly identify VNFs and their properties.

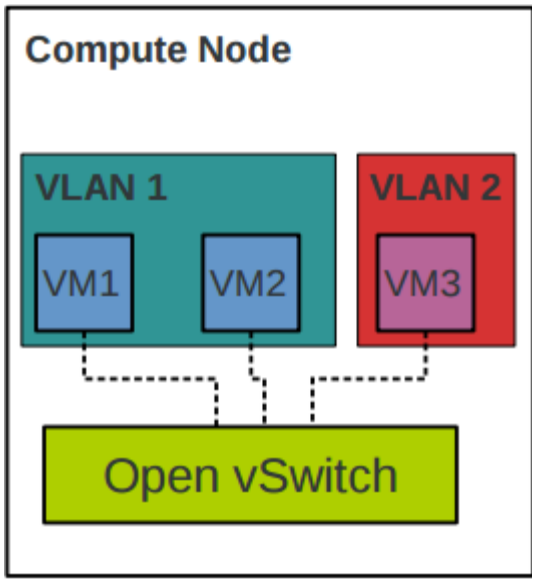


Figure 3. 4 Security zoning between VM in one OVS

3.5 Chapter Summary

To realise an efficient and scalable deployment of virtual network functions, NFV can be implemented in an SDN controlled environment. In this chapter, a framework that illustrates how NFV and SDN can be combined has been presented. The framework describes the main components that constitute the coupling of NFV and SDN. The interaction between these components has been explained in details. Functionally, the NFV and SDN components in the framework fall into three categories: application, control and data planes. The data plane consists of virtual resources in the form of virtual instances sharing the same physical compute infrastructure and running software defined network functions. The control plane orchestrates and manages the data plane virtual network functions using logic defined by the application plane. This application plane consists of the software defined security solutions proposed for the NFV cloud environment. The next chapter demonstrates how security mechanisms and policies can be defined at the SDN controller level to securely isolate the different tenant networks that may share the same physical compute space.

CHAPTER 4

PROOF OF CONCEPT IMPLEMENTATION

4.1. Introduction

In this chapter, a proof of concept (PoC) implementation for the proposed multi-tenancy NFV architecture is presented. Two open-source projects: OpenDaylight (ODL) and OpenStack are used to provide an SDN controller, applications and APIs, and for building an NFV cloud environment respectively. Integration of these two platforms enables implementation of the framework presented in chapter 3. An elaborate discussion is also made on the implementation of the proposed security solutions. The isolation is achieved by using a built-in SDN application from the ODL project, while the access control solution can be implemented using virtual private network (VPN) feature from the Openstack and all the access activities can be monitored by an SDN controller for any suspicious login attempts. This chapter also discusses in detail the proposed network reconfiguration application and its main modules.

This chapter is organised as follows: Section 4.2 presents the various components and technologies from ODL and Openstack that are used for implementation of the PoC model. Section 4.3 discusses a detailed design of the proposed framework and how the different components interact with each other and it also elaborates on proposed security solutions. Finally, this chapter is concluded in section 4.4.

4.2. Tools and Technologies

This section gives a description of the tools and technologies used to implement a PoC scenario for the proposed SDN based multi-tenancy NFV framework. Various components of the ODL project from the Linux foundation are used for implementation of an SDN controller, APIs and SDN applications. Openstack is used for orchestration of the cloud environment. The PoC is implemented by integrating these two tools.

4.2.1 OpenDaylight (ODL)

ODL project is an open-source platform for SDN hosted by the Linux foundation that encourages academia and industry to come up with solutions and use cases at a rapid pace [43]. It is a consolidation of components including a pluggable controller, interfaces, protocol plug-ins and

applications that offer ready-to-install network solutions [44]. Figure 4.1 illustrates some of the existing projects under the ODL Beryllium version of 2016. The subsequent section will elaborate more on the ODL components used for implementation of the proposed framework.

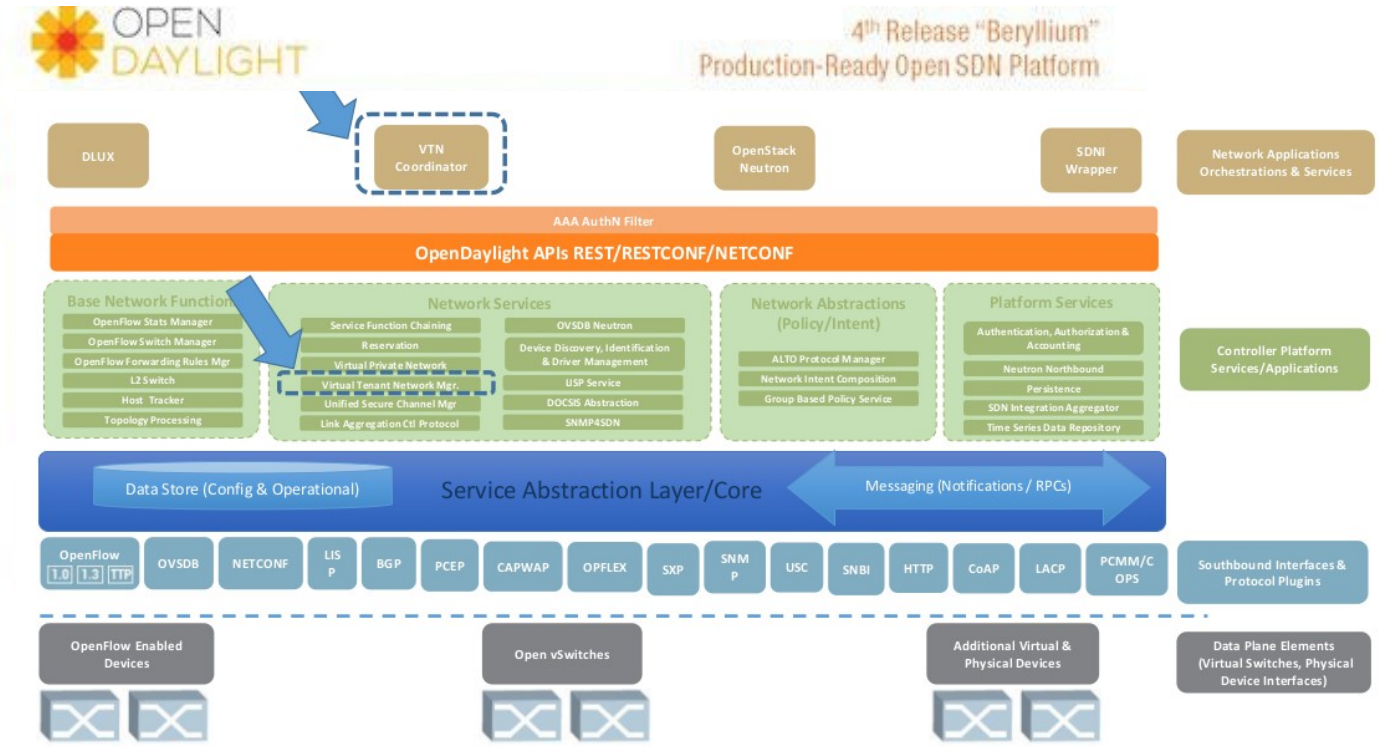


Figure 4. 1 ODL architecture [43]

4.2.1.1. ODL Controller

The controller is implemented as software that resides within a Java Virtual Machine (JVM). The Beryllium controller uses Open Services Gateway initiative (OSGi) and Representational State Transfer (REST) as its main northbound APIs (NB-API) [43]. OSGi is used for communicating with applications within the controller address space whereas the REST is used for applications remote from the controller. For the southbound interfaces separate plug-ins such as OpenFlow (version 1.0-1.3) and Border Gateway Protocol-link State (BGP-LS) are part of the platform. These modules are connected in a dynamic way into a Service Abstraction Layer (SAL) that defines how to accomplish requested service [44].

4.2.1.2. Virtual Tenant Network (VTN)

VTN is one of the applications incorporated into the Beryllium ODL controller. This project enables construction of virtual networks on top of virtual switches (vSwitches). This work uses open vSwitches (OVS) which are open-source software switches defined by the Linux foundation

[44]. Below is a list of built-in VNFs available in the Beryllium VTN package [43]:

- vBridge: a logical representation of L2 network
- vRouter: a logical representation of L3 network
- vTunnel: a logical representation of tunnel
- vLink: a logical representation of physical connectivity tenant network.

VTN uses a logical abstraction plane to decouple the control plane from the data plane. This abstraction enables users to design and deploy virtual networks without stressing about the underlying network topology or bandwidth restrictions [44]. The designed network is then automatically mapped and configured into the underlying vSwitches. The logical abstraction provides simplified resource management of the underlay network, reduces reconfiguration time of services and minimises configuration errors [43]. The application consists of two components:

- **VTN Manager** resides inside the ODL Controller and therefore uses the OSGi plugin to communicate with other modules within the controller. It also has a REST interface for VTN components configuration in the ODL controller.
- **VTN Coordinator** is a northbound application that provides a REST interface to the user to apply the VTN virtualisation. The user configurations will then be referred to the Manager to deploy.

4.2.2. OpenStack

OpenStack is an open-source cloud operating system that can be used to control pools of processing, storage, and networking resources throughout a data-centre [41]. It enables users to deploy VMs and other instances which handle different tasks for managing a cloud environment on the fly through a web-based dashboard, command-line, or a REST API [45]. An OpenStack controller is used to orchestrate the virtual resources in one or more OpenStack compute nodes that form the cloud environment. OpenStack functionality has been integrated into the Beryllium ODL project. OpenStack uses a networking controller called Neutron to provide virtual networking services between managed devices [46]. The Neutron communicates with the ODL controller using the Modular Layer 2 (ml2) plugin as the NB-API. The ODL controller has a Neutron API service which serves as a northbound interface. The API acquires data from the ml2 and makes it available to other ODL services. For the southbound API OpenStack uses Open vSwitch Database Management protocol (OVSDB) which is an OpenFlow configuration protocol designed to manage OVS implementations [44].

4.2.2.1. Open vSwitches

OpenStack supports use of Openflow based software defined virtual switches and for implementation of the proposed framework Open vSwitches were used. The hypervisor usually has finite small number of network interface cards (NICs) that can be shared between the VMs running in a single host [28]. This limits the number of VMs that can be connected per host. The OVSs provides virtual interface (vIF) that can be used to connect the VMs to the physical interfaces (pIF) on the hypervisor which are limited number. These vSwitches enable the cloud environment to be more scalable [46]. Each OVS has an Openflow flow table that contains action rules for forwarding data using MAC addresses, IP addresses or L4 ports.

4.2.2.2. DevStack

Devstack is a series of extensible scripts used to bring up OpenStack services from the software repository—git master [45]. It can be used to demonstrate running OpenStack services and provide examples of using them from a command line interface. It aims to provide and maintain tools used for the installation of the central OpenStack services that are suitable for development and operational testing [45]. It also demonstrates and documents examples of configuring and running services as well as command line client usage. The Devstack scripts were used to build an OpenStack cloud environment in this work.

4.2.2.2. Virtual Private Network as a service (VPNaaS)

VPN is a technology that enables creation of dedicated, encrypted connections to access a private network. It can be used to set up secure connections from tenant access points into the NFV data-centre. The VPNaaS extension feature in OpenStack neutron allows tenants to extend private networks across public telecommunications infrastructure [45]. The extension defines IPSec based connections which is a suite of protocols that provides IP security by offering authentication, integrity and confidentiality. The VPN feature includes the following features [45]:

- Internet Key Exchange policy (ikepolicy): a key management protocol that is used for key management, recreating and exchanging keys. It is also used to identify, authentication and encrypt keys during the VPN set up of security associates.
- IPSec policy: defines a protocol suite used for secure IP communication that authenticates and encrypts all IP packets for all communication session.
- Encryption algorithms: This extension supports triple Data Encryption Standard (DES) and Advanced Encryption Standard (AES) 128, 256 and192 encryption, sha1 authentication, ESP, AH and AH-ESP transform protocol and tunnel or transport mode encapsulation.

4.2.3. OpenDaylight and OpenStack

The Beryllium ODL platform enables integration of the controller and OpenStack to enable design and deployment of multi-tenant virtual networks. The VTN Manager application can be used as a network service provider for OpenStack. Neutron enables OpenStack to work in pure OpenFlow environments, in which all vSwitches in the data plane support OpenFlow [42]. This technology enables creations of VMs on top of the vSwitches and the VMs can be used to host tenant VNFs. Integration of these two components is used to propose the architecture of the security solutions presented in this work.

4.2.6. APIs

To enable communication between the application plane and control plane in an SDN architecture NB-API are used and between the control and data planes the Southbound APIs are used. Below is a list of APIs that have been used in the POC implementation.

- OSGi: It enables communication between the VTN Manager and the controller. As stated in section 4.2.1, this API is used for applications residing within the controller address space and VTN Manager is an example of such an application [43].
- REST API: The network reconfiguration application implemented in this work runs in a different address space to the controller and it therefore uses REST to communicate to the controller. The REST commands include POST, GET, PUT, PATCH and DELETE that are used to Create, Read, Update/Replace, Update/Modify and delete respectively [18].
- OpenFlow: It enables communication of control logic function between the OVSs and the controller. The controller uses OpenFlow to update flow rules in the OVS flow tables and the OVSs are able to query rules for new packets with no predefined action rules [2].
- OVSB: This API enables implementation of Open vSwitch Database management protocol that allows southbound configuration of vSwitches [46].

4.3. Design

In this section a detailed description on how the tools and technologies in section 4.2 have been used to implement a PoC system in this dissertation is presented. The section further describes in detail the proposed security solutions–tenant isolation, network access control (NAC) and network reconfiguration—that are based on an ODL controller .

4.3.1. Multi-tenancy NFV deployment

Figure 4.2 below illustrates the ODL deployment architecture that has been used to

provision multi-tenancy NFV in data centres. The ODL controller acts as a network service provider for the OpenStack. VTN Manager has the ability to subscribe to the events from OVSDB running in one or more OpenStack compute nodes. The ml-2 plug-in between the ODL and Openstack controllers ensures that when Open vSwitch is started to create networking for Neutron, the ODL IP address is set as the manager [44]. ODL receives updates of the OVS through the management port: port 6640 [43]. VTN Manager handles the events from OpenStack and adds a bridge with required port mappings to the vSwitch in the compute node.

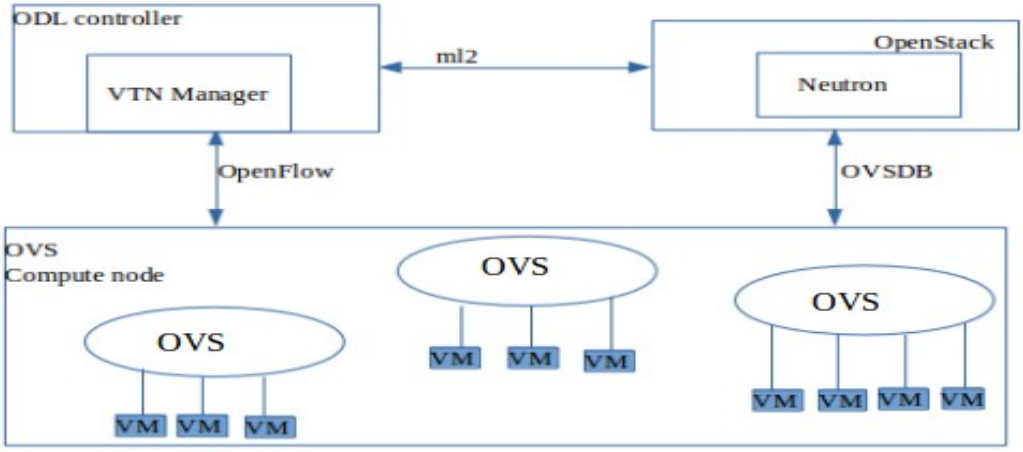


Figure 4. 2 ODL architecture for multi-tenant NFV deployment

Through the OpenStack command-line, the Cloud operator can create tenants and their respective user accounts using the keystone identity service [45]. Subnets with unique network Id can be created and associated to respective tenant Id. Tenant administrators can now design their networks. When Neutron starts up, a new 'network create' REST command is POSTed to the controller then the VTN Manager creates a virtual network by translating the design to the infrastructure resources and defining unique Virtual Local Area Network (VLAN) Id for each tenant. The Manager creates a vbridge interface with port mapping for the particular port on the vSwitch. It also installs flows in vSwitches to facilitate communication between network functions.

The tenant can also add new VNFs by creating new VMs in the OpenStack, the interfaces of the VM will be added to the integration bridge and network is provisioned for it by ODL's VTN neutron bundle [44]. The VTN Manager captures the new PORT and adds a vbridge interface with port mapping for the particular port. The Manager will then update flows of the vSwitches to accommodate the new function. OpenStack and ODL communicate with the OpenStack node resources through the OVSB and the OpenFlow protocols respectively. Figure 4.3 below presents the sequence of events when creating a virtual network.

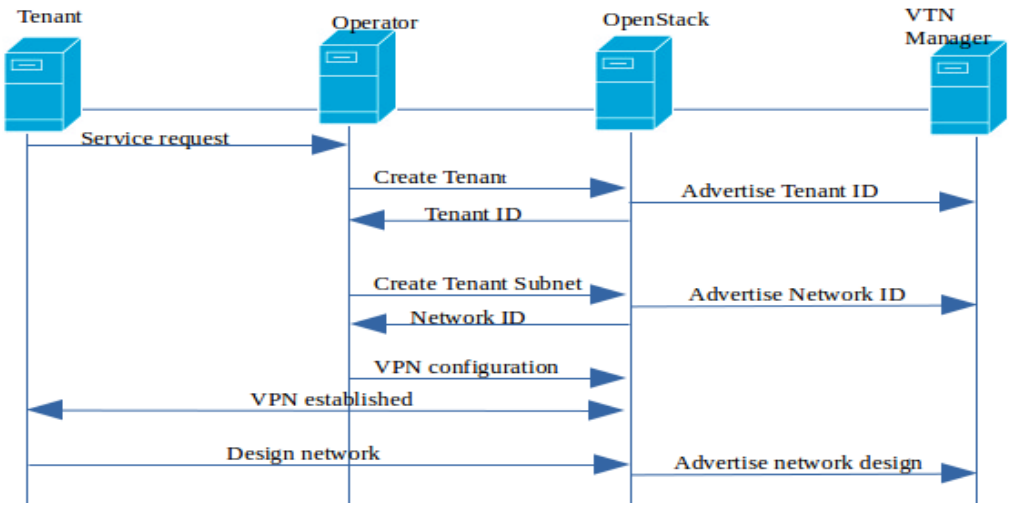


Figure 4. 3 Tenant network creation

4.3.2. Tenant Privacy and Isolation

OpenStack assigns tenant Ids and associates them to respective network Ids. The tenant is therefore not aware of the existence of other tenant networks. When a tenant has designed a network the VTN Manager maps the network into the NFV infrastructure. In Figure 4.4 tenant A and B are not aware of how resources in the compute node have been allocated and where their VNFs reside. Only OpenStack and ODL have the complete view of the compute and networking resources in the compute node; each tenant is only given access to its allocated resources. This deployment mechanism implements isolation and privacy needed for multi-tenancy architectures.

According to the ODL multi-tenancy NFV deployment, the VTN manager maps tenant logical virtual tenant networks (VTN) into the physical nodes which will be OVSs in this case. The manager creates unique VLAN Ids for each tenant network and installs flow entries into the OVS in-order to forward packets from respective interfaces according to the designed logical networks. The OVSs use the VLAN Ids to filter traffic of different tenants.

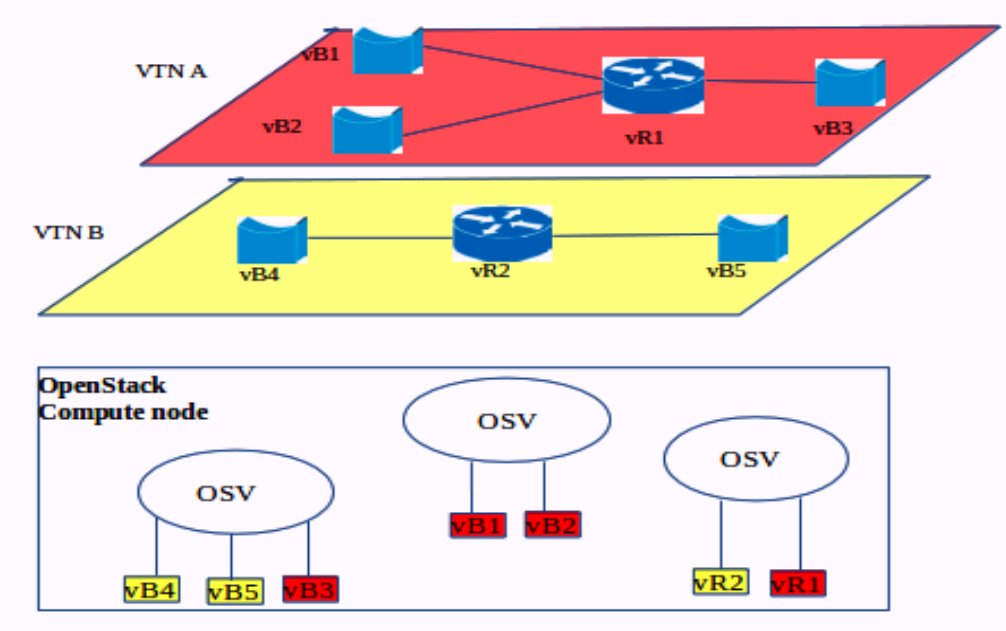


Figure 4. 4 Tenant logical Isolation

4.3.2. Tenant Authentication, Authorisation and Accounting

The VPNaaS feature in OpenStack can be used to set up secure VPN connection between the enterprise tenant access point and the NFV cloud. The VPN can be created from the OpenStack command-line to associate tenant access point IP-address to a specific network Id. The access point is a peer gateway to the tenant. The ikepolicy will be used for key exchange through the Internet. Digital certificates—sha1 can be configured using the AES-256 encryption algorithm. The peers can now negotiate and bring the tunnel up. To limit the risk of attack the Service Level Agreement (SLA) between tenant and operator specifies connection time for different tasks (network design, upgrading or adding VNF). When negotiating a connection, the tenant has to include VPN description specifying task to perform, therefore the corresponding SLA time will be set as time-to-life for the connection. This feature guarantees only valid tenants are allowed into the cloud architecture and that each tenant is only able to access VNFs in associated network ID.

While the Openstack neutron authenticates and authorises tenants to their respective designated logical network, the ODL controller can be used for monitoring all the login information and for reporting all failed login attempts to the Cloud operator for auditing. This enables the

operator to timely identify any malicious attempts and hence implement preventative and reactive measures to improve security.

4.3.3. Network Reconfiguration

This dissertation also proposes an SDN based network reconfiguration moving target defence (MTD) solution for multi-tenancy NFV deployment in a data-centre that dynamically changes IP addresses and vIFs of the VNFs. This can be deployed by a cloud operator to minimise internal attack initiated from other tenants or from external attacks. This solution decreases the chance of an attacker using scanning tools to correctly identify reachability of potential targets. An SDN controller periodically changes the virtual IP addresses (vIPs) and vIFs in tenant networks in intervals of mutation time (T_m). From each tenant network mask the unused IP addresses will be randomly assigned as vIPs to the VNFs. The controller also maps the vIFs of the VMs randomly amongst OVSs in the compute node and updates new interfaces to the VTN vbridge.

An attacker scanning the network will get different IP addresses and location for a single VNF in different time intervals. The controller is responsible for updating the flow tables in the OVSs. It maps the original action rules of each VNF to the changing vIP and interfaces to enable transparent end-to-end communication. The controller will also be used to update DNS responds by responding with the vIP for the queried device.

The IP reconfiguration solution allows only the rightful owners of the virtual networks to use the original IP addresses—real IP (rIP) to access the functions. First a tenant is authenticated and then allowed to access the VNFs using the original flow entries created when the virtual networks were deployed that defines traffic flows using the rIPs. The controller acts as the central authority managing IP mutation, mapping vIFs amongst OVSs, installing flows in the OVS and responding to DNS requests. The proposed solution is built of three main modules discussed below that use REST API for the northbound communication.

4.3.3.1 vIP Assigner

The controller has record of all functions in each VTN and corresponding virtual interfaces connecting to the functions. Each tenant network created in Openstack has a defined network subnet-mask that determines the number of addresses—block size—that can be allocated to unique functions. The number of OVSs in the cloud environment are also noted and all usable vIF are seen as one pool of interfaces. From the unused IP address range in the vIP assigner randomly selects and assigns vIPs for all interfaces in a VTN and maps the vIFs to a different interface from the vIF pool in the cloud environment. This action is performed periodically in intervals of the selected T_m. The assigner has to ensure that a vIP or vIF is not assigned to more than one function at a time to avoid

collision and that it is not assigned one function in consecutive mutation intervals to increase destruction and confusion for an attacker attempting to scan the network properties.

4.3.3.2 DNS responding

When a DNS query is sent to resolve the name of a host, the DNS response is updated by the controller to replace the rIP of the network nodes with currently active vIP. The controller also sets the TTL value in the DNS response to a small value. The source host can then initiate the connection using the vIP of the destination. After T_m time interval all vIP will be mutated meaning that successive DNS queries of the same nodes is likely to give a different vIP causing confusion and complexity for a potential attacker.

4.3.3.3 Network Address Translation (NAT)

After every mutation interval performed by the vIP assigner the controller updates the NAT with new vIPs to corresponding static rIPs. When a source sends packets for the first time the OVS encapsulates and sends the initial packet to the controller. The controller installs flow entries in all the OVSs in the route to the destination host that translates the vIP to the rIP and initial vIF to current vIF hence packets will be forwarded using action associated with the rIP and original vIF to ensure transparent communication. All the OVSs in the route will be configured to route traffic based on vIP addresses. Successive packet can now be matched and forwarded by the OVS within the virtual networks according to the installed flows. The NAT application guarantees transparent end-to-end reachability of hosts, because the rIP-vIP and vIF translation for a specific connection remains unchanged regardless of subsequent mutations.

The algorithm for the proposed system is presented below. The OpenDaylight controller first identifies the unused IP addresses in the network and adds vIFs from all OVSs into a pool of vIFs in the cloud environment. This IP range will be used to assign the vIP for the VNFs. The controller also hops vIFs of the VNFs to randomly selected interfaces from the pool. The controller is also responsible of responding to DNS request by giving out the current vIP of the functions.

OpenDaylight IP reconfiguration algorithm

```
1. determine unused IPs in network block
2. determine number of OVSs in compute node
3. aggregate all usable vIF in pool of interfaces
4. for (packet p from OVS)
5.      if (p.type = Type-A DNS response for hos hi)
6.           set DNS address to current vIP(hi)
7.       else if (p.type=  TCP-SYN or UDP from hi to hj )
8.            if (p.src in internal)
9.                 install in flow in src OVS with
10.                    action srcIP(p)= vIP(hi)
11.             else
12.                install out flow in src OVS with
13.                  action dstIP(p) = vIP(hj)
14.       else if (p.dst = rIP)
15.           if (hi  is authorised tenant)
16.                install in and out flows in OVS
17.        else (p.dst=vIP)
18.              install in flow in src OVS with
19.                  action srcIP(p)= vIP(hi)
20.              install out flow in src OVS with
21.                   action dstIP(p) = vIP(hj)
22.  for all mutation of each host hi do
23.     if (vIP not used)
24.           set vIP(hi) to new vIP
25.     else
26.           find another vIP
27.            set vIP(hi) to new vIP
28.     if (vIF not used)
29.           map vIF (hi) to random vIF
30.     else
31.          find another vIF
32.          map vIF (hi) to random vIF
```

4.4 Chapter Summary

To validate and evaluate the multi-tenancy NFV framework and its proposed security solutions from chapter 3, this Chapter presented a PoC implementation. The tools and technologies used to implement the framework are introduced and a detailed discussion about how different component interact is made. The ODL project is proposed for implementation of the SDN controller with APIs enabling communication to various SDN applications and the cloud environment. Openstack is proposed for implementation of the cloud environment providing VMs that are used to host the VNFs. Devstack, which is a series of scripts used to quickly bring up a complete OpenStack environment is used for the proposed PoC. The Openstack and ODL are intergrated allowing the controller to manipulate the traffic flowing in OVSs within the cloud environment. The chapter further introduces the proposed security solution: Tenant isolation that sets up logical partition between virtual networks in one cloud , AAA ensure only rightful tenant access only their respectively allocated resources and monitor any suspicious login attempts, and network reconfiguration that randomly mutates IP addresses and vIFs of the VNFs in the order to confuse an attacker attempting to scan the network.

CHAPTER FIVE

EXPERIMENTAL PERFORMANCE EVALUATION AND VALIDATION

5.1 Introduction

The previous chapter explained the interaction between the different tools that are used to build a proof of concept (PoC) for the proposed SDN based multi-tenancy NFV framework. This chapter evaluates and analyses the efficiency of the proposed security solutions. First, an analysis is made on the isolation mechanism proposed between virtual tenant networks (VTNs) residing in one compute node. The evaluation is made based on traffic, address space, control and performance isolation metrics. A test scenario is presented and analysed to justify the logical separation between the VTNs. The network reconfiguration concept is also analysed and evaluated based on three merics: deception, deterrence and length of flow table against scanning tools that are normally used by attackers to probe a network for vulnerabilities. Due to constraints in resources and time evaluation of the network access control (NAC) solution has been included as part of future work.

5.2 Isolation Enforcement

In this section an analysis is made about the effectiveness of the isolation aspect implemented for virtual networks in a multi-tenancy NFV deployment. As elaborated in section 4.2.1 the VTN Manager application residing within the OpenDaylight (ODL) controller address space is used for mapping tenant networks into the cloud environment ensuring isolation and privacy between individual networks. The VTN Manager achieves virtual network functions (VNFs) isolation by using the concept of virtual local area networks (VLANs) that allows implementation of port groups associated with unique VLAN Ids for each VTN. The ODL controller then updates Openflow tables within the Open virtual switches (OVSs) in data plane, so they can use the VLAN Ids to filter traffic for different tenants. In this section, an analysis is made based on : Traffic Isolation, Address Space Isolation, Control Isolation and Performance Isolation.

5.2.1 Traffic Isolation

The VTN Manager ensures that service chaining and traffic routing defined for individual virtual networks affect only traffic flow for the designated network [44]. This is achieved by updating the action rules in the flow tables of the OVSs using the Openflow v3 protocol. The ODL controller also provides northbound APIs (NB-API), which is the REST API in this case, that

enables tenants to insert policies on the fly without concerning about the complexities and arrangement of the physical plane. The VTN manager then implements the intended logic for that particular tenant without affecting traffic in colleague networks.

5.2.2 Address Space Isolation

The virtual networks created in the OpenStack dashboard have independent network Ids with unique subnet masks [44]. This feature allows communication between VMs in one subnet. The VTN manager maps the VNFs running inside virtual machines (VMs) into the compute node resources and decides to which OVS each VM will be attached. The VTN manager further deploys layer 2 isolation by introducing VLAN Ids which are independent and unique for each tenant [44]. The Openflow rules installed in the OVSs filter traffic between the different tenant virtual networks using both the VLAN Id and IP addresses. This implies that knowing the network mask of a VTN is not enough to initiate communication with the VNFs. This enables the logical zones as proposed in chapter 3 and offers isolation between tenant networks.

5.2.3 Control Isolation

In the proposed solution, tenants have the complete control over their accounts and all configurations created by a tenant are mapped to the corresponding domain by the VTN manager [44]. This dissertation proposes authentication using tenant Id and security certificates to ensure only the rightful network administrators are allowed into the cloud environment, and that they only have access to only their respectively leased resources. Each tenant is only aware of corresponding VM Ids hosting their respective VNFs. Tenants only have access to the logical designs of their networks but are not able to affect the actual global administrative configurations implemented on the cloud environment.

5.2.4 Performance Isolation

The logical isolation implemented by the VTN Manager also isolates performance metrics between the virtual networks. Any performance metric varying in one network will not be able to affect colleague tenant networks. These metrics include congestion, link fail and security attacks. Deviation in one of these metrics in one network will only affect the performance in that particular network [44]. The VTN Manager ensures that each tenant is confined in only allocated resources.

5.2.5. Isolation Test Scenario

To evaluate the NFV isolation and privacy solution proposed in section 4.3.2 a test scenario is set up based on Openstack cloud environment that is managed by an ODL controller. The VTN

manager within the controller address space was used to set-up three VTNs illustrated in figure 5.1 using REST commands on the ODL command line interface (CLI). The figure presents the logical view of the networks and the physical layer—which represent the OVSs in the Openstack environment that are able to forward packets using Openflow defined action rules implemented by the VTN manager. VMs of each virtual network are added to one VLAN Id and same port group defined on the OVSs. To enable communication within one VTN, the OVSs uses the combination of destination virtual interface (vIF) IP address and VLAN Id to select action rule from the flow table. If there is no flow entry with this combination, the first packet will be forwarded to the controller and the controller will then update the flow tables of all OVS that will need to handle this type of traffic. This test case focuses mainly on traffic and address space isolation between tenants in a single compute environment. The hosts and virtual function were instantiated as VM in the Openstack environment.

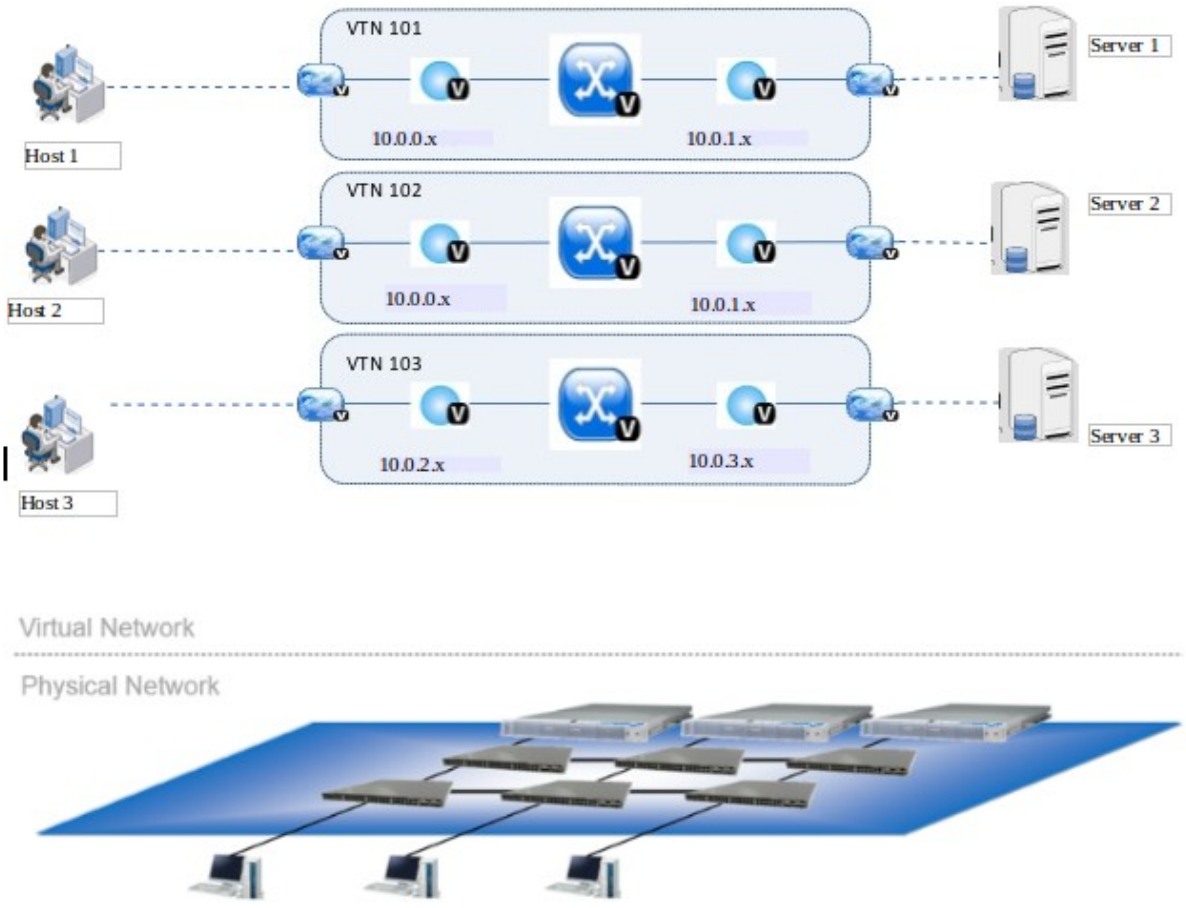


Figure 5. 1 Network Topology observed in Openstack

Figure 5.1 above shows that host 1 and server 1 belong to VTN 101, host 2 and server 2 to VTN 102, and then host 3 and server 3 are members of VTN 103. VTN 101 and VTN 102 are assigned the same network ID. Each of the two virtual networks provides basic IP End-to-End connectivity and consists of one virtual router (vRouter) separating two L2 switches (vBridges) with the IP spaces 10.0.0.0 /24 for the hosts and 10.0.1.0/24 on the server side. The hosts in VTN 101 and 102 are assigned IP address 10.0.0.1, the virtual interfaces on the three vRouters 10.0.0.254 and for the server side all the two servers are assigned 10.0.1.2 with gateway 10.0.1.254. VTN 103 is assigned IP block 10.0.2.0/24 for the host network and 10.0.3.0/24 for the server side. For testing the isolation, the Iperf tool [48] was used to make simultaneous connections from the hosts to the corresponding servers. Each host attempts to set up a TCP connection with their respective server.

5.2.6. Analysis of Results

As expected, all end-hosts 1, 2 and 3 were able to successfully connect to corresponding servers on the other end of the VTNs. As explained in Section 4.5.1, The VTN manager maps the virtual networks into the physical devices and aggregates interfaces for each VTN into logical port group that below to a unique VLAN Id. Despite the same IP address set in VTN 101 and 102 the OVS are able to forward traffic using the defined logical isolation. The OVSs are able to filter traffic of the respective tenants using the VLAN Id which is unique for each tenant. In real live deployments each Tenant has a distinct network mask and therefore traffic of respective tenants will never in any way overlap or interfere. The Iperf results below depict the sample captured packet flow in the respective VTNs above. From the code it can be observed that even though the IP addresses are the same the MAC addresses of the host VMs and server VMs are different for VTN101 and VTN102. Therefore the VTN Manager ensure traffic and address isolation by using unique VLAN ids for each tenant.

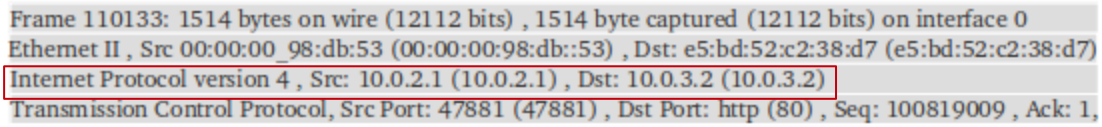
Packet from host 1 to server 1 in VTN101:

```
Frame 16745: 1514 bytes on wire (12112 bits) , 1514 byte captured (12112 bits) on interface 0
Ethernet II , Src 00:00:00_95:38:ad (00:00:00_95:38:ad) , Dst: f8:bf:d2:62:18:c7  (f8:bf:d2:62:18:c7)
Internet Protocol version 4 , Src: 10.0.0.1 (10.0.0.1) , Dst: 10.0.1.2 (10.0.1.2)
Transmission Control Protocol, Src Port: 47879 (47879) , Dst Port: http (80) , Seq: 42918489 , Ack: 1,
```

Packet from host 2 to server 2 in VTN102:

```
Frame 188946: 1514 bytes on wire (12112 bits) , 1514 byte captured (12112 bits) on interface 0
Ethernet II , Src 00:00:00_7c:ec:5e (00:00:00:7c:ec:5e) , Dst: c8:bf:c7:58:76:d2  (c8:bf:c7:58:76:d2)
Internet Protocol version 4 , Src: 10.0.0.1 (10.0.0.1) , Dst: 10.0.1.2 (10.0.1.2)
Transmission Control Protocol, Src Port: 47880 (47880) , Dst Port: http (80) , Seq: 5308065 , Ack: 1,
```

Packet from host 3 to server 3 in VTN103:



The VTN manager allows tenants to define only the logical design of the network and it will deploy the design of the actual cloud environment. Therefore tenants do not have access to change any configuration in the data plane. This ensures control and performance isolation within the cloud environment.

5.3. Network Reconfiguration

The network reconfiguration solution proposed in this dissertation reduces chances of an attacker correctly identifying IP addresses and location of VMs hosting VNFs in an Openstack environment. This is achieved by dynamically assigning virtual IPs (vIPs) for the VMs from unused address space, and hopping virtual interfaces (vIFs) amongst the OVSs in the Openstack compute node. This application uses VTN manager REST commands to update action rules in the OVS flow tables to facilitate communication of the newly added vIP and vIF maintaining the tenant defined logical design and service chaining. In this section the effectiveness of the reconfiguration solution is studied and evaluated using the following metrics:

5.3.1. Deception

The reconfiguration mechanism proposed in this dissertation invalidates attackers' view and assumptions of the virtual networks [14]. This metric measures the ratio of targets that an attacker falsely identifies targets compared to misses in static networks. Assuming N_r and N_s denote number of scans made for an attacker to identify the correct network details in the reconfigured network and static network respectively. Deception ratio can therefore be computed as N_r / N_s. This measure calculates the percentage of resources that evade the attack by deceiving attackers.

5.3.2. Deterrence

This metric compares time an attacker takes to identify on which OVS is the VM connected to and what is its IP address to time that is taken to identify nodes in the static network with the same configuration. Assume T_r is time taken to complete a scan and correctly identify nodes in the reconfiguration enabled network and T_s as time to complete scan in static networks. The deterrence ratio can be defined as T_r/T_s. This metric quantifies time cost exerted on the attacker to successfully complete a scan in a reconfigured network.

5.3.2. Flow Table Size

In static networks, flows are normally matched using destination addresses which are stationery, the length of flow table in the vSwitches is an order of O(n) [33]. For the proposed reconfiguration solution two flows—one for rIPs and the other for vIPs—must be defined for all sessions between hosts. For n hosts that take an average of s seconds to establish connection and the session lasts for t seconds, according to Little's law the length of the flow table is 'nst'.

5.3.2. Network Reconfiguration Scenario

This section presents the experimental scenarios carried out on a VTN constructed in an Openstack environment and controlled remotely by the VTN Manager residing within the OpenDaylight Beryllium controller. VTN 101 from section 5.2.5 was set up for the evaluation of reconfiguration solution on a data plane made of three OVSs that were assumed to have 20 vIFs each. 1000 experiments were conducted each for 3 different randomisation intervals (30, 60, 120 and no mutation) using the attack interval—time between adjacent attacks—of 30s. The randomisation interval implies the time interval between controller's next mutation. The network mutation algorithm presented in section 4.3 is implemented as a JAVA program automating REST commands to re-assign the VM IP addresses and vIF port mappings, and to update flow tables in the OVSs to enable continuity of communication with the VTNs. The table below shows the experimental parameters used to validate the network reconfiguration concept.

Factorial Experimental Parameters

Network size	10-to-10
Randomisation Technique	IP Address and vIF Randomisation
Randomisation Interval(s)	30,60,120 and static
OVS(s)	3 OVSs with 20 vIFs each
Number of experiments	100-1000
Attack interval	30s

5.3.3. Network Reconfiguration Results

The results below presents the experimental data collected using the Wireshark tool [47] based on the above experimental parameters. For different mutation intervals a study was conducted on how many scans are needed and time for successful identification of VNFs. A study for flow table length was made by varying number of sessions established per second for different mutation intervals .

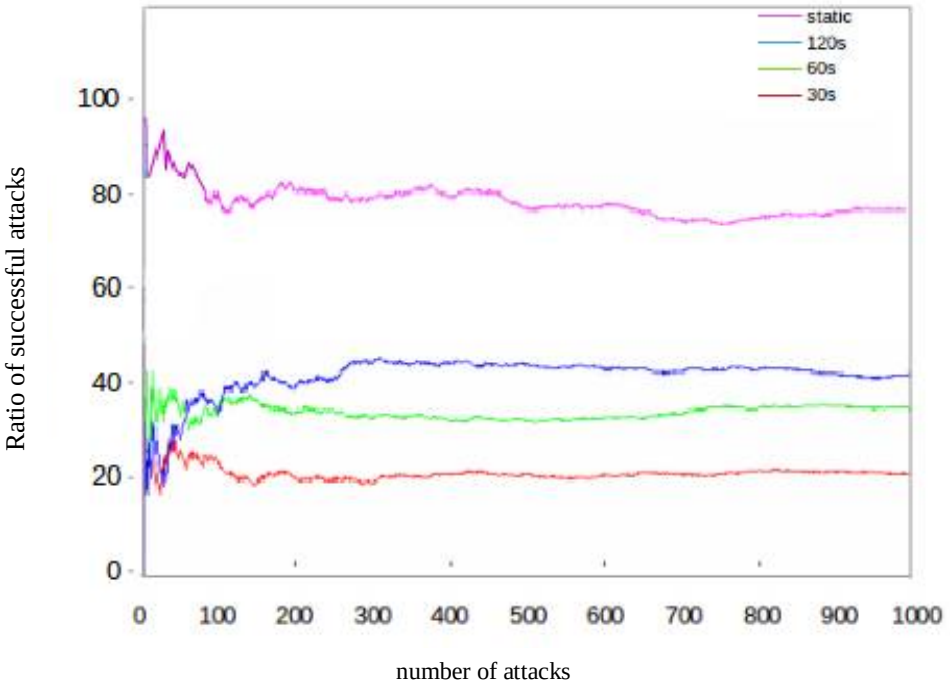


Figure 5.2 Success rate of individual attacks

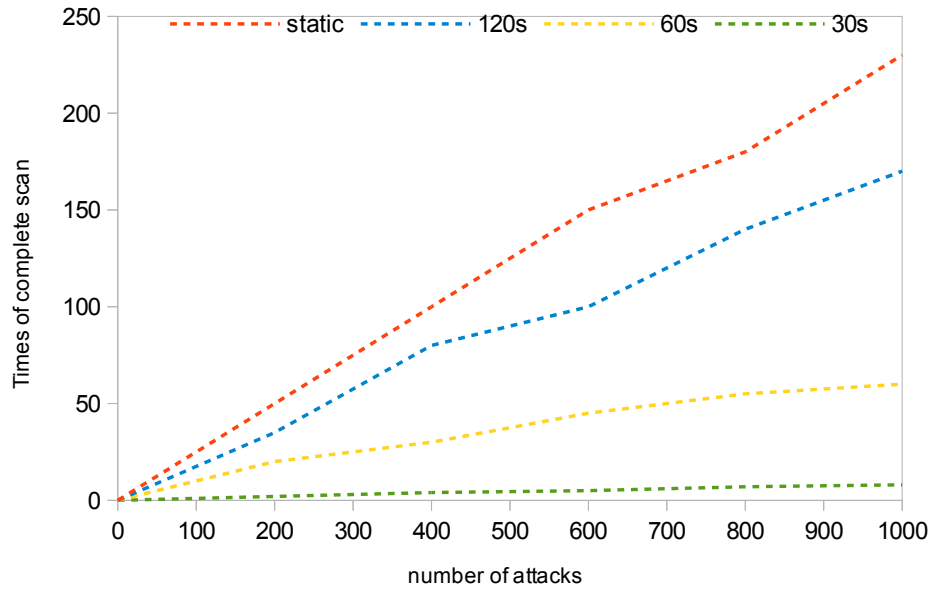


Figure 5.3 Complete attack against target NFV

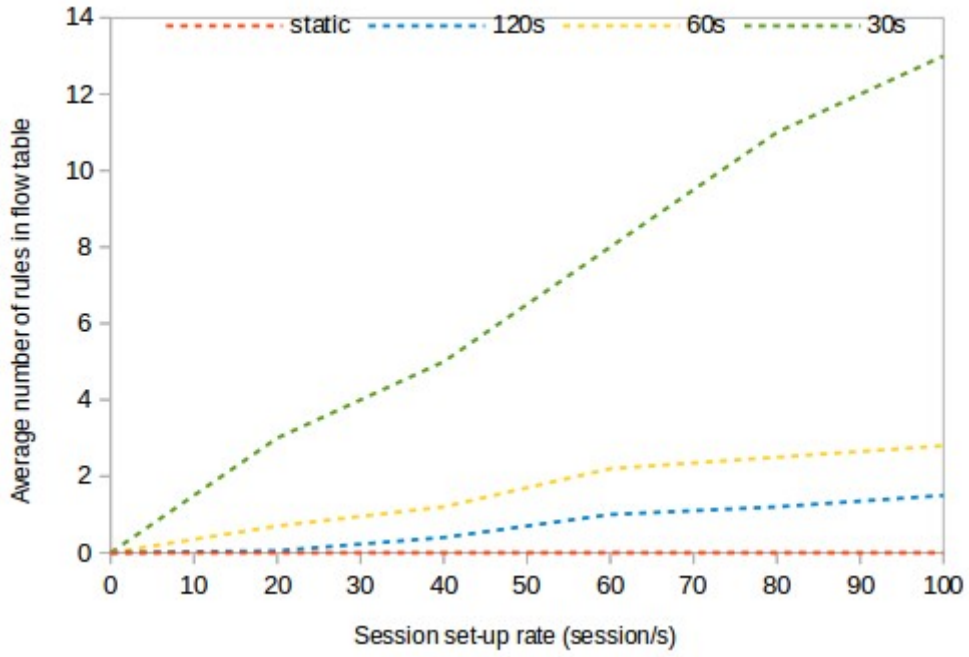


Figure 5. 4 Length of flow-table per mutation interval

5.3.3Analysis of Results

The proposed network reconfiguration solution for multi-tenancy NFV deployment reduces the chances of correctly identifying IP addresses of the VMs hosting tenant VNFs and their location within the NFV cloud. The results above demonstrate the effectiveness of this moving target defence (MTD) technique. Figure 5.2 shows the success of each individual attack between nodes for different mutation intervals. For all the mutation intervals, the success rate fluctuates for the initially small number of samples and then becomes more stable as the number of attacks increases. This figure shows that as mutation interval is reduced (mutation frequency increased), the individual attack success ratio also decreases, as can be observed from the figure success rate is reduced by 40% for the mutation interval of 30s compared to a static network. This implies that as the mutation frequency increases an attacker needs to perform more scans than in the static network. It can therefore be concluded that the deception ratios increase as the mutation frequency increases.

Figure 5.3 also clearly shows the effect of the proposed MTD technique. When the mutation interval is reduced, the success rate of correctly identifying the VMs decreases. This experimentation was conducted for 3000 seconds. When the configuration is static, the number of completed attacks is 240 out of 1000, while for the mutation interval of 120s the number is reduced to 50 and an adaptation interval of 30s allows only 5 successful attacks. This figure also explicitly shows that as the mutation frequency increase more time and number of attacks are required to

successfully identify VNFs therefore it can also be concluded that the deterrence and deception ratios increase as the frequency increases.

To achieve better deception and deterrence ratios higher mutation frequency should be used. However, this will require a network with large block size and unused addresses. A network with largest unused address space can be assigned non-repetitive vIP address in adjacent mutation intervals causing more confusion to an attacker. The size of network block therefore directly affects effectiveness of the reconfiguration solution.

Although it has been observed that increase in the frequency of mutation increases complexity and difficulty for attacker, this on the other hand increases the length of flow tables in the OVS switches and also causes some jitter. Figure 5.4 illustrate that for increase in number of sessions per second results in longer flow table with increase of mutation frequency.

5.4. CHAPTER SUMMARY

This chapter presents experimentations that were carried out as proof-of-concept for the proposed multi-tenancy NFV and the proposed security solutions. An analysis was made to evaluate the effectiveness of the logical separation implemented by the VTN manager to isolation VTNs deployed in one physical compute node. The isolation was analysed by the metrics: traffic, control, address space and performance. The given test scenario illustrates that the OVSs use the combination of IP address and VLAN Id to filter traffic between tenant networks. The VLAN technology enables logical isolation of the networks residing in one compute node and therefore tenants are only aware of their networks and cannot in anyway affect other tenants. The network reconfiguration concept is also analysed and evaluated based on four metrics: deception, deterrence and flow table length against scanning tools that are normally used by attackers to study network topology and its vulnerabilities. From the presented test scenario it has been observed that increasing the frequency of mutation increase the deception and deterrence ratio hence making it harder for an attacker to correctly identify the VNFs location and IP address. The results also implies that an increase in IP address block enables non-repetitive use of of the vIP and also brings more confusion and deception to an attacker. However shorter mutation intervals results in larger flow table size in the OVSs.

CHAPTER 6

CONCLUSION

6.1 Introduction

In this chapter, a brief summary of the content of this dissertation report that have been reported in the preceding chapters is presented here. The chapter also discusses an extension that can be made of this work in the future in order to improve and evaluate in the proposed security solutions in this dissertation.

6.2. Conclusion

Network Function Virtualisation (NFV) and Software Defined Networks (SDN) are two independent technologies that promise network flexibility, increased network and service agility, and support service-driven virtual networks using concepts of virtualisation and softwarisation [2]. Integration of these technologies enables cloud operators to provide network-as-a-service to multiple tenants in a data-centre deployment. NFV defines network work functions as software instance that can be run on any compute node, whereas SDN proposes a centralised architecture that decouples control plane from data plane and introduces network programmability, which makes it possible to dynamically control, change, and manage network behaviour through software, providing programmable interfaces into the network.

Despite the benefits brought by these technologies, they also bring along security challenges that need to be addressed and managed to ensure successful deployment and encourage faster adoption in industry. This dissertation illustrated the benefit of using the two technologies to complement each other by designing, implementing and analysing security solutions focused on tenant isolation, network access control (NAC) and network reconfiguration for an operator offering network-as-a-service (NaaS) to multiple tenants in a data-centre.

The isolation is achieved by implementing logical separation of resources residing in the same hardware to ensure tenant privacy. On the other hand, NAC can be achieved by implementing authentication, authorisation and accounting of tenants to access only their respectively allocated resources. The network reconfiguration is implemented by randomly mutating the IP addresses and virtual interfaces of the network functions. These solutions have been implemented as SDN applications making it possible to intelligently and flexibly control traffic flows to achieve best

network throughput, perform traffic monitoring and analysis, and smart intrusion detection using customised user algorithms [2].

This dissertation reviewed the NFV and SDN concepts and their respective architectures. Existing security solution proposed for multi-tenancy NFV are revised and their strengths and weaknesses analysed. SDN based security solutions proposed for traditional networks and cloud architectures are reviewed and used to model the NAC and security zoning solution for multi-tenancy NFV. The moving target defence solutions proposed in the past are also reviewed and a technique of network reconfiguration is extended to the NFV cloud.

An SDN based framework with data plane, control plane and application layer is proposed for implementation of the multi-tenancy NFV deployment. The data plane consists of the cloud environment comprising of virtual resources that can be used to host tenant VTNs. The control plane is built of a cloud orchestrator and an SDN controller. The cloud orchestrator manages the virtual resources within the cloud platform hosting multiple virtual networks, while an SDN controller simplifies monitoring, management and manipulation of traffic in this environment. SDN applications are the software defined network policies that enable a cloud operator to protect the VTNs hosted in the cloud environment.

A proof of concept (PoC) implementation of the proposed SDN based NaaS architecture on an Openstack cloud environment has been demonstrated. The PoC is based on integration of two open-source platforms—OpenDaylight (ODL) project and OpenStack. The NFV cloud environment is implemented using OpenStack that virtualises the physical compute node resource into independent virtual machines (VMs) that are used to host the VNFs for different tenants. The ODL project is a consolidation of components including a pluggable controller, interfaces, protocol plug-ins and applications that offer ready-to-install network solutions [43]. An ODL controller and the VTN application from ODL are used to orchestrate security solutions for the NFV cloud environment. The VTN manager uses REST commands to the OVS switches in the compute node.

Experimental set-ups were implemented using the tools proposed for the PoC and the proposed security solutions for multi-tenancy NFV deployment were evaluated and analysed. An analysis was made to verify the logical isolation aspect proposed between virtual tenant networks (VTNs) residing in one compute node. From this analysis, an observation was made that member VMs assigned to individual VTNs can only communicate within their home VTN and are not aware of existence of other virtual networks. The network reconfiguration concept is also analysed and evaluated based on three metrics: deception, deterrence and flow table length against scanning tools that are normally used by attackers to study network and its vulnerabilities. An analysis made shows that increase of mutation frequency increases the deception and deference ratio hence

making it harder for an attacker to correctly hit on target VNFs. The results also implies that an increase in IP address block enables non-repetitive use of a single virtual IP address (vIP) and also brings more confusion and deception to an attacker. However, it has also been illustrated that shorter mutation intervals results in larger flow table size in the OVSs.

This dissertation designs, implements and evaluates security solution use-cases that illustrates the benefits of integrating an SDN controller into the NFV architecture to simplify implementation and management of security in the dynamic NFV environment. These SDN based solutions allow automation of security policies hence enabling scalability, programmability, network and service agility at the same time reducing CAPEX. The controller can be used to modify or upgrade security rules based on newly discovered threats or tenant requirements.

6.3. Future Work

6.3.1 Network access control (NAC) solution

Because of constraint in resources, the evaluation of the NAC solution proposed presented in section 4.3.2 for the multi-tenancy NFV environment has been added to future work. The efficiency of this solution can be analysed by studying the strength of the security certificates and the limitation to accessing cloud environment. Tenant administrators should only be authorised to access their respective allocated resources in the multi-tenancy cloud environment.

6.3.2 Network Reconfiguration

The Overhead caused by the proposed reconfiguration solution can be studied in detail in the future. The delay experienced on active sessions, when the first packet gets to the OVS after mutation, while the controller is updating the flow tables can be studied and its impact on transparent end-to-end communication can be analysed. The DNS delay implications that can be expected by nodes when an IP address change occurs during reconfiguration can also be investigated in the future.

A security detection solution can be proposed to work with the network reconfiguration algorithm presented in this work to reactively respond to an attacks by mutating interfaces of VNFs that have been compromised to different OVS interfaces. In this dissertation the solution only pro-actively protect VTNs against potential attacks but does not consider cases when the attack has been successful.

6.4. Chapter Summary

This chapter has given an overview of the work presented in this dissertation report. It gives

highlights on the design and implementation of the proposed SDN based NaaS framework and security solutions. It also provides an insight on the effectiveness of the proposed framework and security solutions. Future analyses that can be done on the proposed NAC and reconfiguration solutions were also presented.

REFERENCES

[1] W. Wang et al., "BalanceFlow: Controller load balancing for OpenFlow networks," Cloud Computing and Intelligent Systems (CCIS), IEEE 2nd International Conference, vol. 2, pp. 780-785, 2012.

[2] ONF Solution Brief, ‘‘OpenFlow-enabled SDN and network functions virtualization,’’ Open Netw. Found., 2014.

[3] ETSI, “Network Functions Virtualisation (NFV): Architectural Framework,” ETSI GS NFV, vol. 2, p. V1, 2013.

[4] ETSI (2013). *Network Functions Virtualisation (NFV); Use cases.* ETSI NFV ISG Draft Documents. [online] Available at: http://dsocbox.etsi.org/ISG/NFV/Open/Latest_Draft/ [Accesses 10 Apr. 2016].

[5] ETSI. (2014). *Network Functions Virtualisation (NFV); NFV Security; Security and Trust Guidance.* ETSI NFV ISG Draft Documents. [online] Available at: http://dsocbox.etsi.org/ISG/NFV/Open/Latest_Draft/[Accesses 10 Apr. 2016].

[6] OPNFV. (2016) *An Open Platform to Accelerate NFV.* [online] Available at: https://www.opnfv.org/sites/opnfv/files/pages/files/opnfv_whitepaper_092914.pdf [Accessed 09 May 2016].

[7] J. Matias et al, "FlowNAC: Flow-based network access control,"Software Defined Networks (EWSDN), 2014 Third European Workshop on. IEEE, 2014.

[8] Open Network Foundation, “Software-Defined Networking: The New Norm for Networks”, O N F White Paper, 2012.

[9] D. Kreutz et al., “Software-Defined Networking: A Comprehensive Survey,” Proceedings of the IEEE,Vol. 103, No. 1, January 2015.

[10] ETSI (2015). *Network Operator Perspectives on Industry Progress, ETSI.* [online] Available at: http://portal. etsi. or/NFV/NFV_White_Paper2. Pdf [Accesses 28 Apr. 2016].

[11] M. Monshizadeh et al., "Cloudification and security implications of TaaS," Computer Networks and Information Security (WSCNIS), 2015 World Symposium, 2015.

[12] W. Braun and M. Menth, “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,” Future Internet, vol 6, pp 302-336, 2014.

[13] C. J. Chung et al., "SeReNe: on establishing secure and resilient networking services for an SDN-based multi-tenant datacenter environment," IEEE International Conference on Dependable Systems and Networks Workshops. IEEE, 2015.

[14] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012.

[15] W. C. Moody, H. Hu, and A. Apon. "Defensive maneuver cyber platform modeling with

Stochastic Petri Nets," Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference, pp. 531-538, 2014.

[16] P. Kampanakis, H. Perros, and T. Beyene., "SDN-based solutions for Moving Target Defense network protection," World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium, pp. 1-6, 2014.

[17] Y. Han, W. Lu and S. Xu, "Characterizing the power of moving target defense via cyber epidemic dynamics," Proceedings of the 2014 Symposium and Bootcamp on the Science of Security, ACM, 2014.

[18] J. Y. Cai et al., "An attacker-defender game for honeynets," International Computing and Combinatorics Conference. Springer Berlin Heidelberg, 2009.

[19] B. Jaeger, "Security Orchestrator: Introducing a Security Orchestrator in the Context of the ETSI NFV Reference Architecture," Trustcom/BigDataSE/ISPA, IEEE. Vol. 1. IEEE, 2015.

[20] B. Cataldo et al., "A novel approach for integrating security policy enforcement with dynamic network virtualization," Network Softwarization (NetSoft), 2015 1st IEEE Conference, 2015.

[21] A. Lemek. *How to manage security inn NFV environments* [online]. Available: https://techzine.alcatel-lucent.com/how-manage-security-nfv-environments.

[22] R. Mijumbi et al., "Management and orchestration challenges in network functions virtualization," Communications Magazine, IEEE 54.1, pp 98-105. 2016.

[23] D. Zissis and D. Lekkas , "Addressing cloud computing securityissues," Future Generation computer systems 28, no. 3, pp 583-592, 2012.

[24] T. Wood, A. Gerber, K. Ramakrishnan, and J. van der Merwe, “The case for enterprise-ready virtual private clouds,” Workshop on Hot Topics in Cloud Computing (HotCloud), 2009.

[25] M. S. Dayananda and A. Kumar, "Architecture for inter-cloud services using IPsec VPN," Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference, 2012.

[26] H. Hu et al., "FLOWGUARD: building robust firewalls for software-defined networks," Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014.

[27] Vmware white paper. (2016). *VMware vCloud Networking and Security Overview.* [online] Available at: http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/products/vcns/vmware-vcloud-networking-and-security-overview-whitepaper.pdf [Accessed 09

May 2016].
[28] B. Pfaff et al., “The design and implementation of open vswitch,” 12th USENIX symposium on networked systems design and implementation (NSDI 15), pp. 117-130, 2015.
[29] F. Webber et al., “Applications that participate in their own defense (APOD),” BBN Technologies Cambridge MA, 2003.
[30] S. K. Xu, Y. N. LI, and Y. W. FU, "A Study on SAR Jamming Technique Based on Deceptive Moving Target [J]," Modern Radar7025, 2008.
[31] S. Antonatos et al., “Defending against hitlist worms using network address space randomization,” Computer Networks, vol. 51, no. 12, pp. 3471–3490, 2007.
[32] J. Yackoski et al., “A self-shielding dynamic network architecture,” Military Communication Conference, 2011 - MILCOM 2011, pp. 1381 –1386, Nov. 2011.
[33]] E. Al-Shaer, Q. Duan, and J. H. Jafarian, “Random host mutation for moving target defense,” SecureComm, vol. 106, pp.310–327, 2012.
[34] S. Robertson et al., "CINDAM: Customized Information Networks for Deception and Attack Mitigation," Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2015 IEEE International Conference, 2015.
[35] W. Han, et al., "HoneyMix: Toward SDN-based Intelligent Honeynet," Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, ACM, 2016.
[36] Y. Jarraya et al., “Multistage OCDO: Scalable Security Provisioning Optimization in SDN-based Cloud, IEEE 8th International Conference on Cloud Computing, pp. 572-579, June 2015.
[37] D.V. Bernardo and B. B. Chua, "Introduction and analysis of SDN and NFV security architecture (SN-SECA)," IEEE 29th International Conference on Advanced Information Networking and Applications. IEEE, 2015.
[38] Verizon. (2016). *SDN-NFV Reference Architecture.* [online] Available at: http://innovation.verizon.com/content/dam/vic/PDF/Verizon_SDN-NFV_Reference_Architecture.pdf [Accesses 20 May 2016].
[39] Cisco. (2016). *Cisco Secure Data Center Solution.* [online] Available at: http://www.cisco.com/c/en/us/solutions/enterprise-networks/secure-data-center-solution/index.html [Accesses 28 Jun. 2016].
[40] P. Yasrebi et al., “Security function virtualization in software defined infrastructure,” IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 778-781, May 2015.
[41] X. Wen et al., “Comparison of open-source cloud management platforms: OpenStack and OpenNebula,” In Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference, pp. 2457-2461, May 2012.
[42] R. Muñoz, et al. "SDN/NFV orchestration for dynamic deployment of virtual SDN

controllers as VNF for multi-tenant optical networks," Optical Fiber Communication Conference, Optical Society of America, 2015.

[43] Opendaylight.org. (2016). *Platform Overview | Opendaylight*. [online] Available at: http://www.opendaylight.org/software [Accesses 03 May 2016].

[44] Wiki.opendaylight.org. (2016). *OpenDaylight Virtual Tenant Network (VTN):Overview – Opendaylight Project.* [online] Available at: https://wiki.opendaylight.org/view/OpenDaylight_Virtual_Tenant_Network_%28VTN%29:Overview [Accessed 27 May 2016].

[45] OpenStack. (2016). *Software & raquo; Openstack Open Source Cloud Computing Software.* [online] available at: http://www.openstack.org/software [Accessed 27 May 2016].

[46] P. Ben and B Davie. (2013). *The Open vSwitch Database Management Protocol.* [online] Available at: https://tools.ietf.org/html/rfc7047 [Accessed 10 Jun. 2016]..

[47] Wireshark.org. (2016). *Wireshark . Go Deep*. [online] Available at: https://www.wireshark.org/[Accessed 27 Aug. 2016].

[48] Iperf.fr. (2016). *iPerf – The TCP, UDP and SCTP network bandwith measurement tool*. [online] Available at: https://iperf.fr/ [Accesses 15 Aug. 2016].

Appendix A: Experimentation

A.1 Set-up and configuration

This section describes in detail how the actual components used to implement the multi-tenancy NFV cloud environment that is managed by an SDN controller are configured and integrated to work with each other. The goal of this part is to guide the reader through the installation and configuration steps of the controller and the cloud environment.

A.1.1. System Requirements

The ODL controller Java virtual machine was implemented in a computer with the following specification: 6 Cores CPU, 6 GB RAM, 300 GB Hard drive and Ubuntu 14.04 64 bit Operating System. The Openstack was configured using Devstack in Virtual machine—Oracle Virtualbox environment with the following properties: 4 Cores CPU, 6 GB RAM, 32 GB Hard drive, 3 virtual network adapters—NAT, Host-only and Bridged—and Ubuntu trusty server 64 bit Operating System.

A.1.2. CLOUD Platform Set-up

Devstack Mitaka was used to provision an OpenStack based cloud environment using the latest versions downloaded from a git master downloaded from [47]. This deployment was installed in an Ubuntu trusty 14.04 server residing in a virtualbox environment. Deployment of the Devstack scripts enabled one Openstack controller node and one compute node. When stacking the devstack script, neutron was enabled to communicate with OpenDaylight. This was achieved by adding the code extract below to the local configuration file—localr. This file contains all custom settings for Devstack.

```
[ml2_odl]
url=http://localhost:8080/controller/nb/v2/neutron
username=admin
password=admin
```

The "./stack" command was executed to run the devstack scripts and bring up the Openstack cloud environment. The code extract below shows command used to set ODL controller as manager for the OVS in the Openstack compute node.

```
# ovs-vsctl set-manager tcp:$localhost:6640
[devstack ~]# ovs-vsctl show
 9f3b38cb-eefc-4bc7-828b-084b1f66fbfd
  Manager "tcp:localhost:6640"
     is_connected: true
  Bridge br-int
     Controller "tcp:localhost:6633"
     fail_mode: secure
     Port br-int
        Interface br-int
  ovs_version: "2.1.3"
```

The Openstack dashboard shown in figure 7.1 below was accessible on the browser in the local machine, it can be used to create virtual networks using Open vSwitch kernel switches and to display existing ones. The internal network addresses were chosen from class A network ranges. To simulate external hosts, we designated one of these subnets as external subnet.

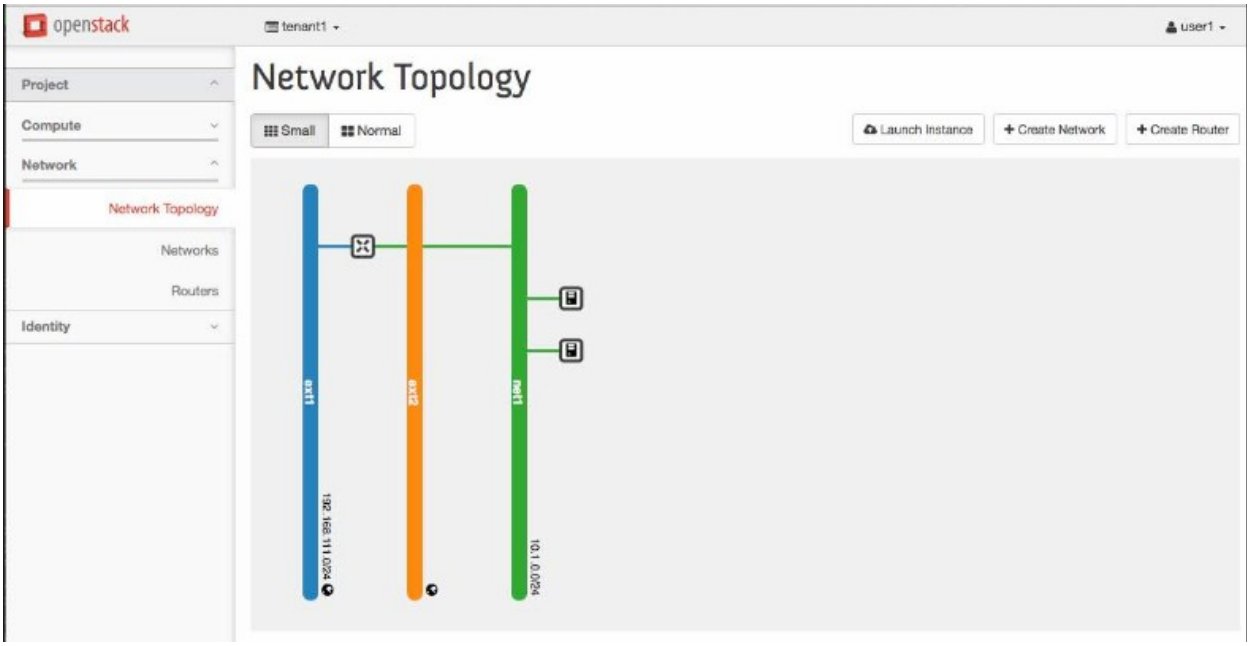


Figure 7. 1 Networks created in OpenStack

A.1.3. OpenDaylight

On a Ubuntu 14.04 host machine, JAVA and Postgre packages and dependencies, needed to compile the VTN application, were installed using the command below:

```
“aptget  install”: “gcc  make g++  maven  libboostdev libcurl4opensssldev
git pkgconfiglibjson0dev libssldev unixodbcdev ant xmlstar-let”.

“post-gresql 9.3  postgresql client 9.3 postgresql client
common postgresql contrib 9.3 odbc postgresql”
```

Berryllium OpenDaylight project .tar file was downloaded from [43] and the controller can be configured to run in a Java virtual machine.

```
cd distribution-karaf-0.4.0-Beryllium
 ./bin/karaf
```

The VTN Manager, which is a built-in application within the controller, was installed using the command below. This enables the Manager to use neutron networking to communicate with OpenStack through the rest API.

```
root> feature:install odl-vtn-manager-neutron odl-vtn-manager-rest
```

The odl-ovsdb-openstack feature was installed in the controller to enable use of the OVSBD protocol for communication with the OVS in the OpenStack environment. The odl-dlux installs the controller Web portal. This portal is available from on the host machine on URL: http: localhost: 8088 and it can be used to view the switches connection and network connection in the Openstack environment. Figure 7.2 displays sample of the network topology in the Openstack environment display in the ODL user interface. Below is the list of features that were enabled in the controller to allow integration of Openstack and OpenDaylight.

```
root>feature:install odl-base-all odl-aaa-authn odl-restconf odl-nsf-all
   odl-adsal-northbound odl-mdsal-apidocs odl-ovsdb-openstack odl-ovsdb-northbound
                              odl-dlux-core
```

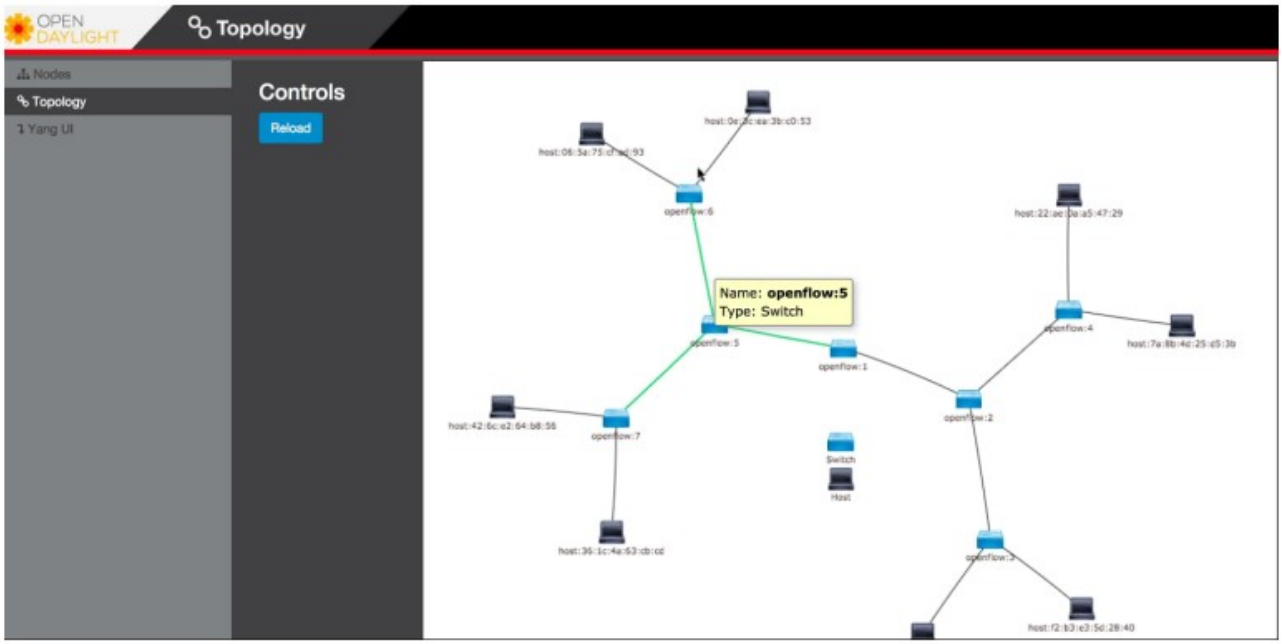


Figure 7. 2 Network Topology observed in Openstack

A.1.4. VTN Manager and OpenStack

Virtual Tenant Network (VTN) manager application residing within the controller was used in order to control and manage the networking. VTN creates a virtual networking environment, in which each network inside is a different VTN and is managed as an independent network. Below is a list of REST commands (POST, PUT, GET and DELETE) that were used to create tenants and map the VTN to the Openstack environment.

A.1.4.1 Create VTN1

curl --user "admin":"admin" -H "Content-type: application/json" -X POST \http://localhost:8181/restconf/operations/vtn:update-vtn \ -d '{"input":{"tenant-name":"vtn1","update-mode": "CREATE","operation":"SET","description":"creating vtn","idle-timeout":300,"hard-timeout":0}}'

A.1.4.2 Create vBridge for VTN1

curl --user "admin":"admin" -H "Content-type: application/json" -X POSThttp://localhost:8181/restconf/operations/vtn-vbridge:update-vbridge \ -d '{"input":{"update-mode":"CREATE", "operation":"SET","description":"creating vbr","tenant-name":"vtn1","bridge-name":"vbr1"}}'

A.1.4.3 Create first interface for VTN1

curl --user "admin":"admin" -H "Content-type: application/json" -X POST\ restconf/operations/vtn-vinterface:update-vinterface \ -d '{"input":{"update-mode":"CREATE", "operation": "SET","description":"Creating vbrif1 interface","tenant-name":"vtn1","bridge-name":"vbr1","interface-

name":"if1"}}'

A.1.4.4 Map VTN1 to physical interface

curl --user "admin":"admin" -H "Content-type: application/json" -X POST \ restconf/operations/vtn-port-map:set-port-map \ -d '{"input":{"vlan-id":0,"tenant-name":"vtn1", "bridge-name": "vbr1","interface-name":"if1", "node":"openflow:1","port-name":"s2-eth1"}}'

Appendix B: Publication

The conference presentation and paper below have resulted from this dissertation:

1. R. Lejaha and J. Mwangama, “SDN based security solution for Multi-Tenancy NFV,” South African telecommunication network and application conference (SATNAC), September, 2016.

EBE Faculty: Assessment of Ethics in Research Projects

Any person planning to undertake research in the Faculty of Engineering and the Built Environment at the University of Cape Town is required to complete this form before collecting or analysing data. When completed it should be submitted to the supervisor (where applicable) and from there to the Head of Department. If any of the questions below have been answered YES, and the applicant is NOT a fourth year student, the Head should forward this form for approval by the Faculty EIR committee: submit to Ms Zakiya Chikte (Zakiya.chikte@uct.ac.za); New EBE Building, Ph 021 650 5739).

Please note – It is important to keep a signed copy of this form as students must include a copy of the completed form with the dissertation/thesis when it is submitted for examination.

Name of Principal Researcher/Student: RETSELISITSOE S. LEJAHA **Department:** ELECTRICAL ENGINEERING

If a Student: **Degree:** M.ENG IN TELECOMMUNICATIONS **Supervisor:** JOYCE MWANGAMA

If a Research Contract indicate source of funding/sponsorship: SELF-SPONSOR

Research Project Title: SDN BASED SECURITY POLICY FOR MULTITENANCY NFV

Overview of ethics issues in your research project:

Question 1: Is there a possibility that your research could cause harm to a third party (i.e. a person not involved in your project)?	~~YES~~	NO
Question 2: Is your research making use of human subjects as sources of data? If your answer is YES, please complete Addendum 2.	~~YES~~	NO
Question 3: Does your research involve the participation of or provision of services to communities? If your answer is YES, please complete Addendum 3.	~~YES~~	NO
Question 4: If your research is sponsored, is there any potential for conflicts of interest? If your answer is YES, please complete Addendum 4.	~~YES~~	NO

If you have answered YES to any of the above questions, please append a copy of your research proposal, as well as any interview schedules or questionnaires (Addendum 1) and please complete further addenda as appropriate.

I hereby undertake to carry out my research in such a way that

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

Signed by:

	Full name and signature	Date
Principal Researcher/Student:	RETSELISITSOE SYLVIA LEJAHA	26/04/16

This application is approved by:

Supervisor (if applicable):		26/04/2016
HOD (or delegated nominee): Final authority for all assessments with NO to all questions and for all undergraduate research.		29/4/16
Chair : Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the above questions.		