TECHNISCHE UNIVERSITÄT
CHEMNITZ

Faculty of Electrical Engineering and Information Technology

Professorship of Communication Networks

Master Thesis

Autonomous security response orchestration with machine learning-based traffic prediction for programmable networks

for the fulfillment of the academic degree

M.Sc. in 'Information and Communication Systems'

Vibha R Goutham

Chemnitz, 1st Jan 2020

Supervisor : Prof.Dr.Ing-.Thomas Bauschert
Supervisor : Mr. Trung Phan Van

Declaration

I hereby declare that this Master's thesis titled "*Autonomous security response orchestration with machine learning-based traffic prediction for programmable networks*" is my own independent work. This work has not been, in part or in whole, presented or published elsewhere for academic assessment. Any form of content or information by other sources or authors that is used in this report is explicitly acknowledged or referred.

Chemnitz, 1st Jan 2020

_____

Vibha R Goutham

Acknowledgments

Add Acknowledgments later!

Abstract

The advent of Internet has impacted the world in an irreversible manner. The global network of interconnected computers opened up the virtual domain of cyberspace. The fast evolving domain of cyberspace, in the past few decades, has impacted people's lives in unimaginable ways while also making innumerable mundane tasks fairly simple. The speed at which the cyber world continues to advance marks a testament to the technological affiliations of the modern world but the malicious use of the same reminds the downside as is the case with most present day technologies. The nature and complexity of cyber-attacks is growing prominently as fast as the advance in Internet technologies itself making cybersecurity a vital area of research. This stresses on the need to tackle cyber-attacks and respond to them at speeds beyond human capabilities and necessitates for security functions to be automated. The focus of this thesis is to develop an automated security function using virtualization techniques to aide in dealing with a certain kind of cybersecurity threat. In the course of this research, a desired security function was designed, implemented, deployed and further analysis was performed using machine learning.

Contents

1. Introduction

1.1. Motivation

The Internet and the World Wide Web(WWW) are among the most successful inventions of the modern world and has been a medium to access information. People relying on the Internet has been a phenomenon like one seen never before. Internet is not only a source of entertainment but also a medium for the functioning of various domains such as secure banking and investment, healthcare, education while also critical for military and defence strategies. With the advent of 5G mobile technology the number of smartphone users is estimated to increase even further. This emphasizes on providing reliable service to end users and further stresses on the need for secure communication. Meanwhile, in the past several new age complex cyber-attacks have proved a serious threat by impacting critical businesses ranging from several minutes to few hours incurring huge financial losses. Cybersecurity, thus becomes an absolutely necessary field of research for these reasons. To deal with new age attacks, automating security functions with lower response times becomes essential to detect and deflect complex threats which in turn minimizes human intervention as much as possible. This thesis takes place in the context of NWO SARNET project on Security Autonomous Response with programmable NETworks [SARNET], a Dutch research collaboration project between research organizations including TNO, Netherlands. SARNET will be discussed in more detail in the next chapter. Further, traffic prediction and analysis for the cyber-threat was performed under the Chair of Communication Networks at TU Chemnitz, Germany.

1.2. Aim of the Thesis

Data ex-filtration is a type of cyber-attack type involving unauthorized copying, transfer or retrieval of data from a computer or server. The thesis aims at developing an automated security function focusing on providing a solution to a cyber-security threat of data exfiltration. In the course of this project, a model of an attacker-victim was set-up on a virtual switching domain. The aim at first is to detect the onset of the attack based on traffic volume prediction using machine learning techniques. For this purpose, customized malicious traffic data sets are used to train and test the machine learning algorithm in order to predict future malicious behavior in the network. Secondly, the aim is to mitigate the attack by transferring the infected traffic and/or infected node to a safe node over an encrypted secure medium. At the safe node, a provision for further inspection about the nature of attack or the attacker may be performed. The goal is also to achieve the transfer of the infected traffic or node to a safe node without providing any knowledge of the aforementioned actions to the attacker in question.

The automated security function is designed by combining the principles of SDN, NFV and security orchestration. The security function performs traffic prediction analysis which shall be discussed under attack detection. The design, implementation and deployment of the automated security function in the orchestrated cloud shall be discussed under mitigation. The detection and mitigation plan together forms the crux of this thesis and will be discussed in the subsequent chapters.

1.3. Structure of the Thesis

This report begins with the chapter of introduction that briefs about the motivation and aim of this thesis, presenting the reader the need for a security function to be developed. The second chapter covers an overview that presents definition for cyber attack and its types while also discussing about cyber security. Further in the third chapter, a theoretical detailing is provided on underlying concepts of SDN, NFV, security orchestration among other principles the course of the project relies on. Also here, the security function design and use cases are discussed in detail. Subsequently, in the next two chapters, the attack detection and mitigation is presented respectively. In the attack detection chapter, machine learning algorithm and techniques used to achieve traffic prediction are elaborated, while the attack mitigation chapter explains the implementation of the security function and offers automated mitigation plan in the form of security orchestrator playbooks. In the sixth chapter, results are presented with further statistical analysis and its evaluation is recorded. Finally, the seventh and the last chapter summarizes with a conclusion and outlook of the thesis.

2. An Overview on Cyber-security

2.1. A definition of cyber-attack

The foremost challenge in addressing a problem is to estimate the nature and scope while specifying its inclusions and exclusions leading to a formal definition. A variety of definitions have been applied to cyber-attacks as cyberspace pursuits deviate from traditional principles and classification. Although several definitions aim to convey a similar meaning, it is all the more important to define cyber-attacks so that a legal framework is formulated to deal with it. In this section, two definitions are presented below.

The UK based legal publishers, Practical Law Company (PLC) defines cyber-attack as 'an attack initiated from a computer against a website, computer system or individual computer (collectively, a computer) that compromises the confidentiality, integrity or availability of the computer or information stored on it' [1]. Deriving from this definition, the CIA triad model is formed comprising of Confidentiality, Integrity and Availability that steers information security policies for organizations. Confidentiality underlines the access of sensitive information by intended users and preventing access to ones it is not intended for. Integrity ensures prevention of data altering by un-authorized users and maintaining accuracy throughout the process cycle. Availability emphasizes on hardware and software resource maintenance and prompt recovery of faulty resources to ensure continued use of resources. Further, dealing with cyber-attacks is discussed in [1], which states that procedures for investigating and responding to a cyber-attack depends largely on the nature of the attack itself.

Cyber-attacks has been proposed in [2] as 'any form of assault or retreat operation engage by individuals or organizations that focus on computer information systems, infrastructures, computer networks, and/or personal computer devices by various means of malevolent acts usually originating from an unidentified source that either steals, alters, or destroys a specified target by hacking into a susceptible system'. This provides an insight to interpret activities of cyber-attack perpetrators and to begin understanding laws accorded by existing legal bodies. It is interesting to note that the Shangai Cooperation Organization comprising China, Russia and other South-Asian observing countries such as India, Iran, and Pakistan recognizes cyber-attacks in a different perspective from that of the U.S. National Research Council as reviewed in [2].

Cyber-crime is a computer-oriented crime where the computer or Internet is used to carry out a malicious activity or is the target in itself. Complaints such as human traffickers using the internet to lure victims, illicit drug business, and cyber-bullying are a few examples of identified cyber-crimes. A cybercriminal(s) is an individual or organization performing malice due to an ulterior motive to gain unauthorized access to or make unauthorized use of an asset. Cyber-attacks in the past have demonstrated that varied motives lead to execution of such

attacks. Large scale cyber-attacks carried out with specific political, military or commercial interest have been accused of being State-sponsored attacks. Hacktivist is an Internet coined word for individuals or group who hack the cyberspace to promote a social, political or religious agenda. An Insider threat is also a commonly observed attack type where an employee or third-party worker of an organization performs a deliberate malicious action but could also arise out of negligence or by accident. Countries across the world have dedicated cyber cells working in co-operation with police, prosecutors and judges to understand such crimes and punish perpetrators. However, the United Nations Organization (UNO) notes [3] that out of 194 member states of the United Nation Conference on Trade and Development (UNCTAD), 138 countries have enacted cyber-crime legislations and more than 30 countries have no legislation in place. On comparing E-Commerce legislations worldwide, the report states 79% of countries have adopted E-Transaction laws and 52% have consumer protection laws. Also, while 72% of countries have adopted legislation for cyber-crime, only 58% have data protection and privacy legislation in place.

2.1.1. Vulnerabilities and Threats

A system that adheres to the CIA triad model of Confidentiality, Integrity and Availability for data and resources is considered to be secure as stated in [1]. and noted earlier in this chapter. The systems that defy either one of the three components of the CIA triad model is said to be compromised [4]. Hence, such systems that fail to comply with the model are often easily subjected to possible risks which therefore are recognized to be classified as threats and vulnerabilities. The systems that come under threats result in being exploited. In table 2.1, commonly encountered threats are discussed. An existing shortcoming in systems often leads to threats discussed here. Further, the state of systems being exposed to weakness of being attacked or harmed is therefore termed as vulnerability. In table 2.2, vulnerabilities commonly found in electronic systems [4] are broadly differentiated and discussed.

Table 2.1.: Threats

Type	Description
Unstructured Threats	Developed by security administrators or developers to identify loopholes and to ensure robustness.
Structured Threats	Deliberate attempts by hackers using advanced techniques.
External Threats	Arising from people who are not indigenous to a network or system and are trying to gain access for the same.
Internal Threats	Originated from authorized or unauthorized access, but arising from within the system or network.

Table 2.2.: Vulnerabilties

Type	Description
Network Technology weakness	Arising out of unsecure network equipment such as routers, switches and firewalls; on badly designed or installed unpatched operating systems with security loopholes; on bug-laden applications or databases; lack of built-in security mechanism on the TCP/IP protocol such as HTTP, FTP, SNMP, etc.
Configuration weakness	Due to unsecure user accounts exposing critical account information; weakly configured DNS authentication systems; common or easy passwords that can be cracked; adhering to default settings of products ridden with vulnerable settings.
Security Policy weakness	Arising from poorly drafted security policies due to insufficient monitoring and auditing; insufficiently informed security administrators or end-users; missing disaster recovery action plan.

2.1.2. Types of cyber-attack

The chapter initially presented prevalent definitions of cyber-attacks and how consequences of the vulnerabilities of a system and threats encountered can lead to a cyber attack. It is important to also note the types of such existing attacks. With the advent of cyber security measures While various new ways of attacks are continuously being invented, some of the types of cyber-attacks are noted in [5]. Commonly observed attack types are listed below.

- Data exfiltration: A type of attack involving the malicious activity of copying, retrieving or transferring of data by an unauthorized user. Also, leakage of sensitive data to an unauthorized entity by an external threat leading to data theft can be noted here. Advanced Persistent Threats (APTs) can be mentioned here, which is an advanced attack type where the primary goal is data exfiltration and a continuous effort to steal restricted company or organizational data.
- Malware: A collective word used to describe an attack type comprising of notorious and rogue software that includes ransomware, spyware, viruses and worms is broadly called as malware. It encashes on an existing system vulnerability resulting in blocking access or disrupting parts of the network, transferring data from the hard drive of the computer or installing unnecessary, harmful software. Typically the attack can arise when a suspicious email attachment or a malicious link is clicked on.
- Phishing: An attack type that relies on fraudulent communication, mostly email, coaxing users to reveal personal information such as passwords or bank details and in turn gaining unauthorized access to the system for performing malicious activity.
- Man-in-the-middle attack: In simpler terms this attack type is known as eavesdropping, where an unauthorized person thrusts between an ongoing transaction between two

parties and aims to filter or steal data. This attack type can be observed commonly on unprotected public Wi-Fi networks. Session hijacking, IP spoofing are common examples.

- Denial-of-service attack: The attacker overloads the system with enormous number of valid requests such that system or network is flooded and in turn becomes incapable of fulfilling them. This attack exhausts resources and bandwidth making sure any future genuine request cannot be completed either. Distributed-denial-of-service (DDoS) is a popular attack type where several already compromised systems are used to deny service to users. Teardrop and Smurf attack are other examples.

- Zero-day exploit: When a network vulnerability is detected or announced usually, there exists a window of time where a patch or an update is implemented to rectify it. A zero-day exploit abuses this disclosed window to carry out an attack.

- Drive-by attack: Attackers identify insecure websites to position malicious script containing HTTP or PHP code. This attack type does not require an active action from the victim's end but on simply visiting such suspicious websites, malware gets installed on the user's system.

- Password attack: The most common form of authentication end users have are passwords. Stealing passwords can have systematic approaches such as sniffing insecure connection to track unencrypted passwords; using a random approach of using 'brute-force' to try and test possible passwords; using a dictionary to guess and gain access to a system called as the 'dictionary attack'.

2.2. Cyber-crimes in the past

Cyber incidents targeting Government institutions, defence and infrastructure companies, economic and technology companies have been repeatedly observed in the past. A study made by Centre for Strategic and International Studies (CSIR) records cyber incidents since 2006, [6] comparing countries being targets against them being victims as shown in Fig. 2.1.

Based on possible motivation and specified targets cyber-attacks have been further classified as listed in [7]. These attack types are listed below while also noting relevant incidents that have had impact and been popular in the past.

- Attacks aimed at Nation States:
  Countries are targeted specifically with cyber-attacks aiming to derange normal functioning. In 2007, Estonia was targeted by Distributed Denial of Service (DDoS) attack by the Russian Government when the former decided to remove a Soviet World War II memorial in Tallinn [8]. While commoners were unable to access the Internet or their bank accounts it was reported that the Estonian President and parliament websites, Government ministries, media houses were also the targets. Recently, in January 2019 Iran was alleged to be involved in a global DNS hijacking campaign in the North America,

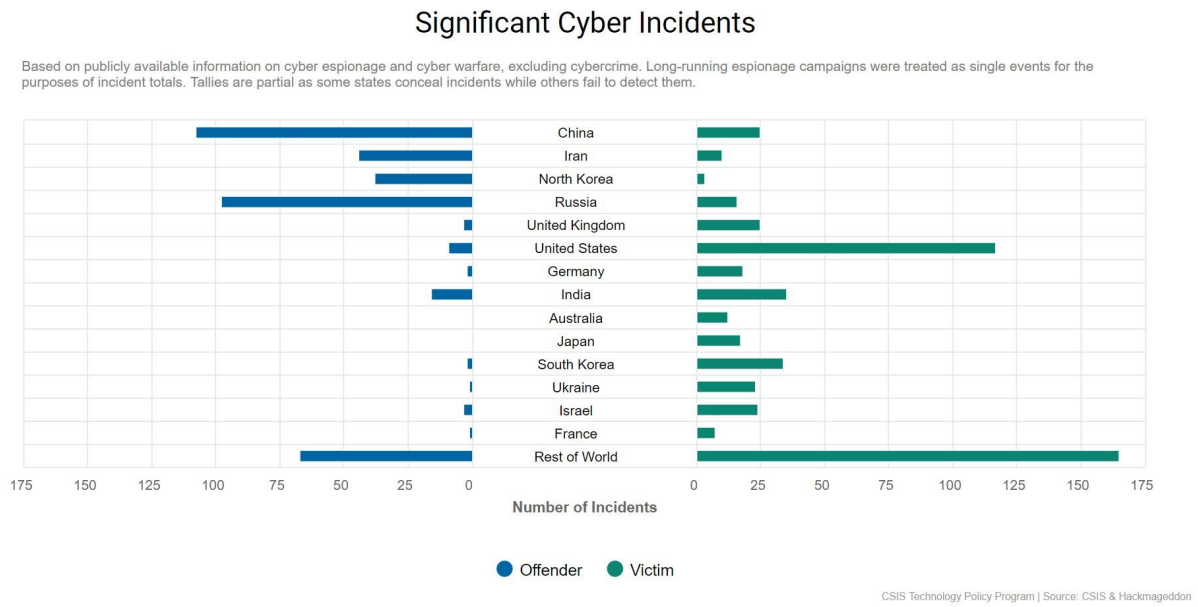

Figure 2.1.: Countries involved in cyber incidents as offender vs. as victim.

Europe and Middle East where target victims were Internet and telecommunications service providers and Government agencies [9].

- Attacks on National security:
  Cyber-attacks are reported on Government institutions, military and defence networks to procure information and to alter engagements that are political, strategic or military among other vital areas. In February 2019, Airbus, the European aerospace company reported theft of personal and IT access related information of European employees by Chinese hackers. Earlier in January 2019, a North Korean botnet was interrupted and taken down by the U.S.A following perpetrated attacks on its media houses, aerospace and financial sectors. Lockheed Martin, an American based global defence, aerospace and security company was under a cyber-attack in 2011 that brought one of its networks down for a week. Data from the SecurID, a RSA based security system was breached. The Department of Defence and the Homeland Security assisted in analysing and mitigating the attack [9].

- Attacks aimed at critical infrastructure:
  The network of systems and assets that are necessary for continued operation of a country to maintain its security, public safety and health, economy, oil and energy is termed by Governments as critical infrastructure. An incident in 2010 where a cyber-worm, Stuxnet was used to infiltrate a nuclear plant in Iran demonstrates this type of attack. The worm was undetected by security systems which made its way to machine controlling software. It targeted Centrifuges that spin material at high speeds and were isolating Uranium types used in nuclear weapons. As a result, infected machines were

disintegrated and further decommissioned by the Government of Iran [10].

- Attacks aimed at companies:
  Several organizations in the sectors of banking, finance, technology, communications and private sector companies are targeted by cyber-attacks from foreign Nations or from within one's own country. In 2018, Google experienced a data diversion attack that re-routed traffic via servers in Russia, China and Nigeria and was reported as a consequence of gateway protocol hijacking attack. As a result, access to a few Google services were impacted [11].

2.2.1. WannaCry: A worldwide cyber-attack

In May 2017, a ransomware crypto-worm targeted more than 230,000 computers in at least 150 countries [12]. The systems that became vulnerable to the malware were mostly running out-of-support Microsoft Windows OS and specially systems that had not installed the security patches that were released earlier in 2017. The Server Message Block(SMB) protocol uses TCP and UDP ports for file sharing and printing purposes. The SMB interface allows any code to be introduced into the system, which is an implementation vulnerability of the interface. WannaCry, upon entering vulnerable systems acquired different types of files such as document, image, video or audio files located either on the hard drive or on the network drive. The infected files were then encrypted while files got modified with a .WNCRY extension. Due to this, users were unable to access infected systems thereafter, while a message pop-up as shown in fig. 2.2, requested a crypto-currency ransom amount be paid to be able to free the system from the infection. The infected code generated individual bitcoin address to receive ransom from each affected system. However, this generation feature failed and the attackers were unable to recognize payments made was for which infected system.

Figure 2.2.: Message pop-up on systems affected by WannaCry

All over the world, several companies were affected from the attack as depicted in fig 2.3. National Health Service (NHS) hospitals in England and Scotland were one of the

worst affected. Around 61 NHS organisations were disrupted which included infected medical equipment. Among several other affected companies were Deutsche Bahn (Germany), Ministry of Internal Affairs of the Russian Federation, Nissan Motor Manufacturing (UK), O2 Telefonica (Germany), Renault (France), FedEx [13]. Before the attack infested widely, a 'kill switch' was discovered which acted as an emergency switch to prevent spread of the infection further. The attack was finally curbed when Microsoft released emergency security patches. The total losses claimed by different institutions was reported to be around billions of dollars.

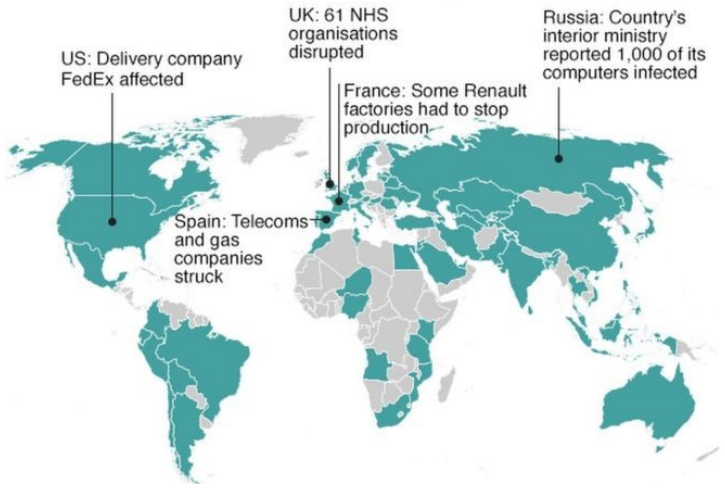

Figure 2.3.: Companies affected worldwide from WannaCry

2.3. Cyber Security and Robustness

The International Telecommunication Union (ITU) in [14] defines cybersecurity as a collection of various factors that forms a framework comprising of guidelines, tools, policies, security concepts, security safeguards, guidelines, risk management approaches, actions, training, best practices, assurance and technologies to safeguard the cyber environment, organization and user's assets. Cybersecurity also ensures in attaining and maintaining the security properties of assets and of the organization against pertinent cyber threats and risks in accordance to the CIA model of Confidentiality, Integrity and Availability as defined by the ITU in [15]. It is important to note that security in general and cybersecurity in particular is a continuous process and not an end state. Various standards have been developed and practiced by organizations to deal with cyber threats, implement security controls that have optimum cost benefits while also complying with legal and regulatory security guidelines. Cyber security standards define functional and assurance requirements within a product, system, process or technology environment [16]. The objective of these standards are to upgrade security infrastructure of systems and networks.

A survey by the European Union (EU) in 2017, reports that 80% of European companies had experienced at the least one cybersecurity incident in that year [17]. To equip the EU

organizations with counter-mechanism a wide range of security control measures have been undertaken by the Union. The European Union Agency for Network and Information Security (ENISA) is the expertise centre to assist Member States in dealing with cyber-attacks and the centre put forth the Cyber Security Strategy in 2013 [18]. Since 2009, ENISA is responsible for cyber-security standardization, record challenges, co-ordination between different countries and to work on EU initiatives in the direction of standardization as noted in [18]. A "Joint framework for EU Diplomatic Response to Malicious cyber activities" was proposed to strengthen international co-operation including relations between the EU and NATO along with a blueprint to respond quickly to large-scale cyber-attacks [17]. The industry standards are increasingly relying on the Cloud for application deployment, mobile and distributed services among several of its other applications. While the cloud is comparable to the term Internet, cloud computing aims at offering services such as storage, databases, networking, software over the Internet. The usage of Cloud paves way for flexible usage of resources and increased innovation while also simultaneously stressing on security due to the wide range of services it can offer. The implementation of this thesis also uses the Cloud as a service. The 'EU initiatives' as part of the security standardization proposed the 'EU Cloud Strategy' in 2012 as depicted in fig. 2.4[19], which focused on increased innovation, reduced costs while emphasizing on legal framework around Cloud computing.

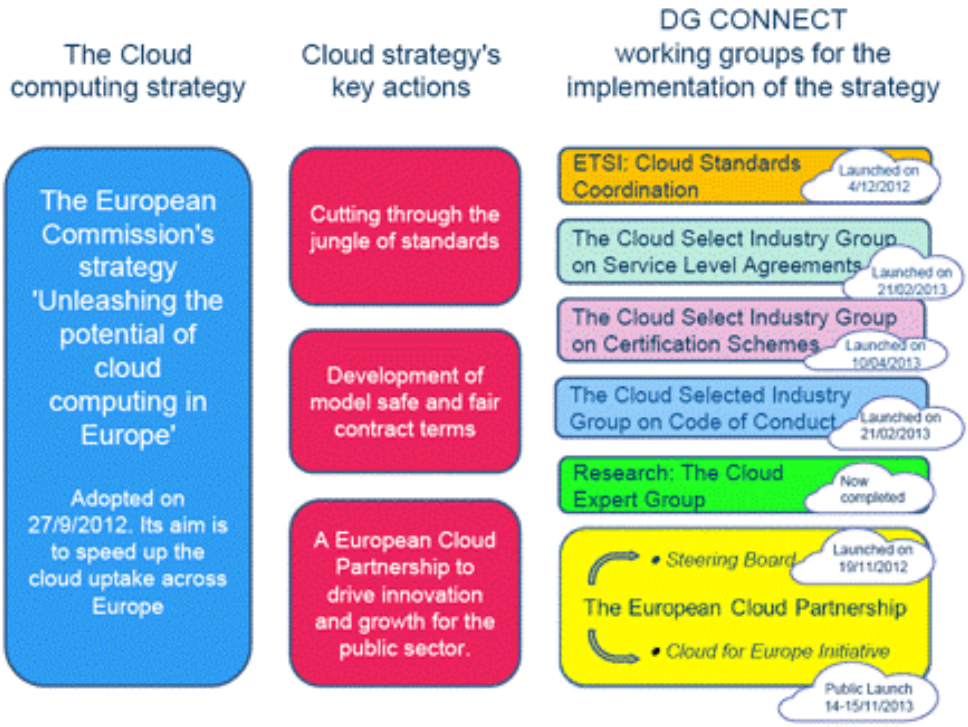

Figure 2.4.: The EU Cloud Strategy

Also, it works towards fostering adoption of security standards and to provide certification

schemes to improve trust between Government bodies, cloud providers and industry [18]. By the year 2020, a net gain of 2.5 million new jobs and a boost of 160 billion euro annual rise in the Gross Domestic Product(GDP) of the European Union was estimated as a result of employing this strategy.

2.4. Cyber Security and Robustness (CSR) at TNO

At TNO in the Netherlands, the Cyber Security and Robustness (CSR) department comprising a dedicated team of professionals works on continued secure and robust ICT networks and services [20]. It is responsible for developing innovative solutions for design, assessment and optimizing complex ICT infrastructures for improved cyber security, performance and resilience to failures and cyber threats. The Research and Development wing mainly focuses on the following four areas of Transaction Security- developing secure transaction designs by employing techniques such as cryptography and blockchain solutions; Security monitoring and detection- developing solutions for identifying and dealing with unknown attacks using anomaly-based techniques and network traffic analysis; Performance of Networks and Systems- designing reliable and robust ICT networks to control and optimize performance of networks and systems. This thesis is however a part of the Automated security group, developing quick automated response solutions to aid in security decision support. The execution of optimal responses to possible cyber-attacks and cyber threats, modelling potential cyber-attacks employing machine learning techniques are focus areas of the group. TNO handles projects for the Government in the area of defence. Customers are also in the field of banking, telecommunications and logistics and TNO values partnerships with Universities, research groups, knowledge institutes and product vendors.

2.5. SARNET

The abbreviated form for 'Security Autonomous Response NETwork' is called SARNET, a Dutch research project group headed by University of Amsterdam in collaboration with TNO along with other industry partners of the group Air France – KLM, COMMIT and CIENA [21]. The research group works on developing best ways of autonomous protection against various types of cyber-attacks using software defined, virtualized detection and defence mechanisms. The second sub-project of the SARNET group focusses on 'Creating a SARNET alliance', with the focus on organizing SARNET functionalities across multiple service provider and enterprise networks to build a trust based alliance to detect and mitigate cyber threats while authorizing each of the collaboration partners to be involved. The project structure of SARNET is organized at three levels. The Tactical level- determining best defence scenario against cyber-attacks by deploying functions [22] and analysing security state and KPI information [23]; the Strategic level- autonomous SARNET behaviours are modelled to identify risks and advantages for stakeholders; Operational level- designing functionalities to operate a SARNET using Software Defined Networking (SDN) and Network Function Virtualization (NFV), delivering security state and KPI information [21].

3. Designing the security function

3.1. Theory

3.1.1. Decision-making model

OODA Loop

OODA (Observe-Orient-Decide-Act) is a decision-making model given by military strategist John Boyd that was created from observing and examining fighter pilots in aerial combat[24]. The principle was later adopted by strategists, businesses and military services for operations research. The OODA loop facilitates in decision-making under changing circumstances and helps to thrive in a volatile environment of ambiguity and uncertainty of the outside world, comprising four processes as depicted in fig 3.1 and as described below[24].

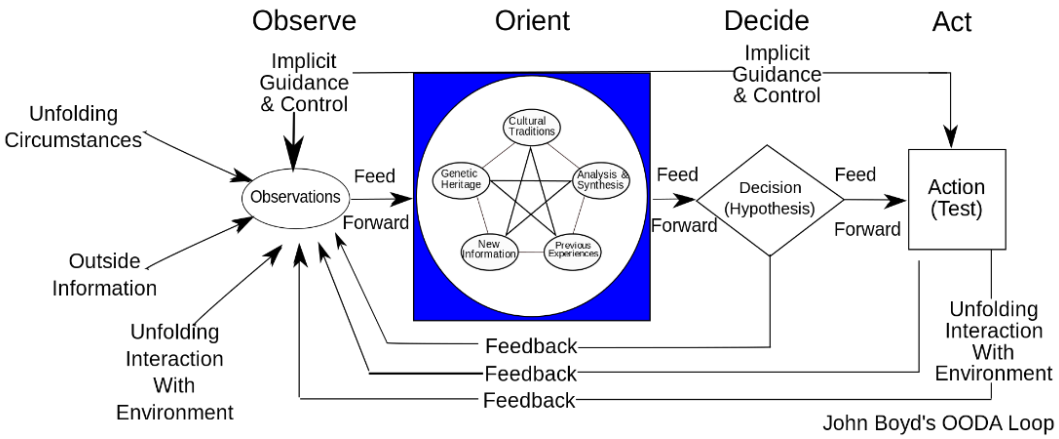

Figure 3.1.: OODA Loop

- Observe: The process of acquiring information about the environment by interacting, sensing, or receiving information about it. It is also guided and controlled by the Orient process along with the feedback from the Decide and Act processes.
- Decide: The process of choosing from a number of hypotheses about the situation and its consequential responses. It is also guided by internal feed-forward from Orient and provides internal feedback to Observe process.
- Orient: The interactive process of implicit projections, permutations, correlations and rejections that is shaped by previous experiences/instances and unfolding circumstances.

- Act: The process of testing selected hypotheses and its interaction with the environment. It receives internal guidance and control from the Orient process as well as feed-forward from Decide. It provides internal feedback to Observe.

SARNET Loop

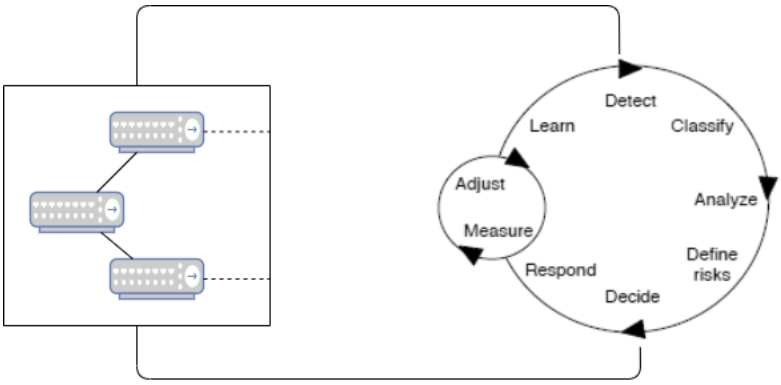

Figure 3.2.: Sarnet Loop

The Secure Autonomous Response Network (SARNET) framework provides autonomous response across multiple domains to network attacks by exploiting underlying SDNs functionalities and virtualised network functions[25]. The SARNET control loop in fig 3.2, is similar to the OODA when successfully applied to cyber security and has an additional step which increases the granularity. In this newly added learn phase, data is collected and stored to improve response times for future attacks[26]. It aims at providing a system that can react autonomously to attacks by utilizing a knowledge pool of tactics tailored to the strategies defined by the businesses that use the system. SARNET loop monitors and continuously evaluates a number of security observables in the state of the network and services. Detection of change in values of observables and any violation of the expected state initiates this control loop. The classify, analyse and risk-defining is the recognition phase, after which SARNET autonomously decides the appropriate response to restore the network to an acceptable security state. Adjustments and re-measuring is performed upon not returning to the desired state after response. SARNET reprograms the network flows, redefines location of the virtualised network functions, and possibly move the location of computing and storage services[25]. Adopting from the OODA and SARNET Loop, the course of the thesis designs a security function that builds an automated response system. The detection phase is handled by machine learning-based classification algorithm to detect traffic volumes based maliciousness. The ML module assists in classifying between good and malicious traffic, analysing and defining threshold for risks. The Decide process receives a culminated input to classify traffic and provides a feedback to the other planes of the network stack to prepare for a response. The Respond process is handled by the mitigation plan achieved through an automated security orchestration. The Learn phase is also handled by the mitigation that fetches and stores information for any further detailed analysis.

3.1.2. Software Defined Networking (SDN)

In traditional communication networks, a host of a number of network devices are utilized to perform varied tasks such as monitoring and management, ensuring security and reliability, switching, routing, etc. In order to respond to the challenges faced by the network, complex network algorithms and protocols are implemented on the network devices comprising them. Maintenance and management of the network is often challenging as network devices are implemented on particular vendor-based configurations and varying interfaces in the form of closed code. This also makes standardization of network interfaces and protocols that increase the degree of inter-operability a complex process. Research and innovation in the last two decades have led to the evolution of active network, where a packet carries the program instead of raw data[27]. On receiving a smart packet, network devices execute the program and the different actions it specifies, thus responding to what the packet carries instead of a passive transmission of packet payload from one node to another.

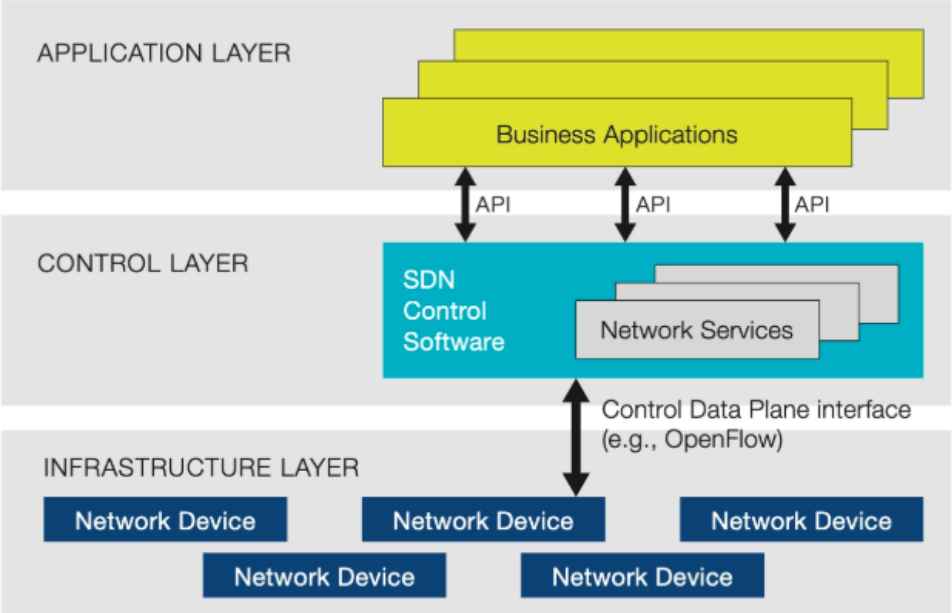

Figure 3.3.: SDN Architecture

Software Defined Networking (SDN) presents network programmability through which the process of developing network applications, network integration and rapid prototyping for managing complex networking systems is simplified. Thus, SDN techniques also drastically change the process of designing, building, testing, and operating networks. As presented in fig 3.3, the architecture of SDN involves three planes: application, control, and data planes[27]. The network intelligence is logically centralized to the Control Plane and in certain cases is extended to the Application layer through customized applications. The application plane executes network applications whereas the control plane regulates the rules for the entire network based on the requests generated by these applications. The controller configures the

network devices in the data plane (or infrastructure plane) based on the set rules. As a result, the network devices in the data plane forward packets to different nodes based on the instructions given by the controller. In between the control and data plane, flow management follows two approaches, proactive- where the controller is able to install flow entries permanently and in particular before they are actually needed; reactive- where flows are installed on demand. In fig 3.4, reactive flow management is presented where a new packet on the data plane is forwarded based on the rules set by controller[28]. SDN decouples the control and data planes allowing both the planes to evolve independently in programmability, automation, and network control thus enabling them to build highly flexible and programmable networks. In SDN, the data plane and the control plane can interact within a closed control loop. The control plane receives network events from the data plane, which then computes network operations based on the events and the resource information of the network. The data plane in return executes the operations, which can change the network states[29].

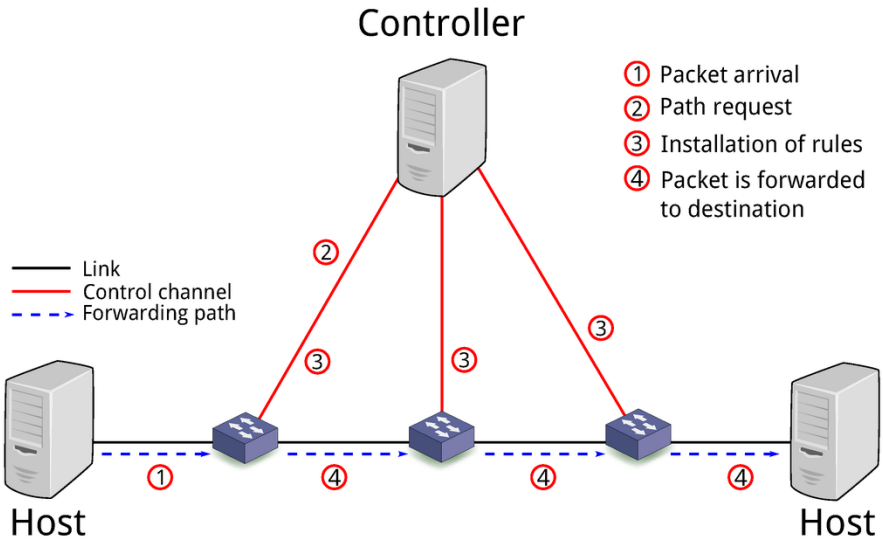

Figure 3.4.: SDN reactive flow management

In the southbound direction, the security controller interacts with data plane network devices through southbound APIs. The information such as task parameters, running status of the resources, etc. are transferred from the control plane to the data plane. A feedback of result parameters, resource logs are returned from the data plane to the control plane. OpenFlow Protocol is widely used in SDN which is responsible for the communication through a secure channel between an OpenFlow Controller that manages a flow table containing match field, instructions regarding the flow and an OpenFlow Switch that manages its own flow table given by the OpenFlow Controller. In the northbound direction, the controller interacts with network applications via northbound APIs through which network policies are issued to the controller and the implementation results are returned[30].

3.1.3. Network Function Virtualization(NFV)

The traditional network systems relied on independent and dedicated devices with specialized equipment such as switches, routers, firewalls, load balancers, etc. to accomplish individual tasks involving packet processing. This owed to growing infrastructure with increased operational and maintenance costs. It also resulted in scalability and compatibility challenges when incorporating multi-operator devices. These limitations led to evolution of Network Function Virtualization (NFV) where functions of network elements was logically virtualized leading to its separation from hardware infrastructure as depicted in fig 3.5[31]. SDN facilitates dynamically controlling the network and the provisioning of networks as a service. On the other hand, NFV offers to manage and orchestrate the virtualization of resources for the provisioning of network functions and building into higher-layer network services. Hence, this makes SDN and NFV complementary and also co-dependent.

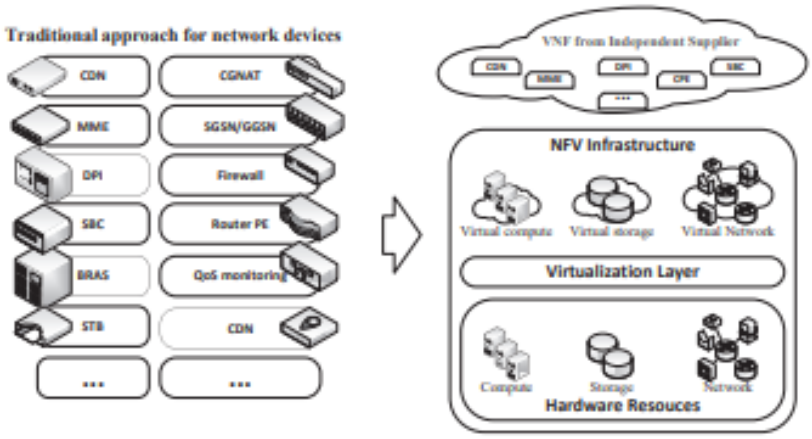

Figure 3.5.: Traditional approach vs. NFV approach

As a result, an amalgamation of network devices can be used for functions such as computing, data storage, network and providing bandwidth which are dynamically allocated to software Virtual Network Functions (VNFs). This approach lowers the overall costs for provisioning and maintenance of network devices while also allows flexible placement and optimisation of VNFs in the infrastructure depending on the demand for service. The ETSI NFV Industry Specification Group (ISG) is responsible for standardisation and the NFV-SEC working group focuses on establishing trust and security in the NFV Infrastructure (NFVI). The specifications are updated through a two-year phase release management process and in [32] a discussion on NFV release 2 and open issues are presented. In its release 3, during 2017-18, support for contemporary technologies such as edge computing and network slicing, advances in acceleration technologies virtualization among a set of other features were discussed[33]. In its latest 2019-20 NFV Release 4, the group shall work on optimizing networking integration, optimizing NFV-MANO (Management and Orchestration) framework's capability and usage, support to lightweight virtualization technologies amongst a host of additional features[33].

3.1.4. Security Orchestration

Network Function Virtualization (NFV) aims at the virtualization of network devices and intends to replace specialized network equipment using standard IT virtualization technologies. With growing networks, secure communication is often stressed upon and in the NFV approach, secure virtual networking and security management thus becomes an important requirement. The ETSI NFV Management and Orchestration (MANO) working group that has defined the ETSI NFV Reference Architecture further aims at controlling the NFV environment through orchestration and automation, thus introducing a Security Orchestrator to handle this requirement[34]. Security orchestration is the process of integrating various mutli-vendor security and non-security products, automating tasks through workflows while also creating provision for human interaction and end user oversight. Security Orchestration, Automation, and Response (SOAR) technology as depicted in fig 3.6, is a Network Service Orchestration combined with the automation of detection and incident response, which through the help of intelligent service feed organizes and coordinates network services to achieve the requirements of end users.

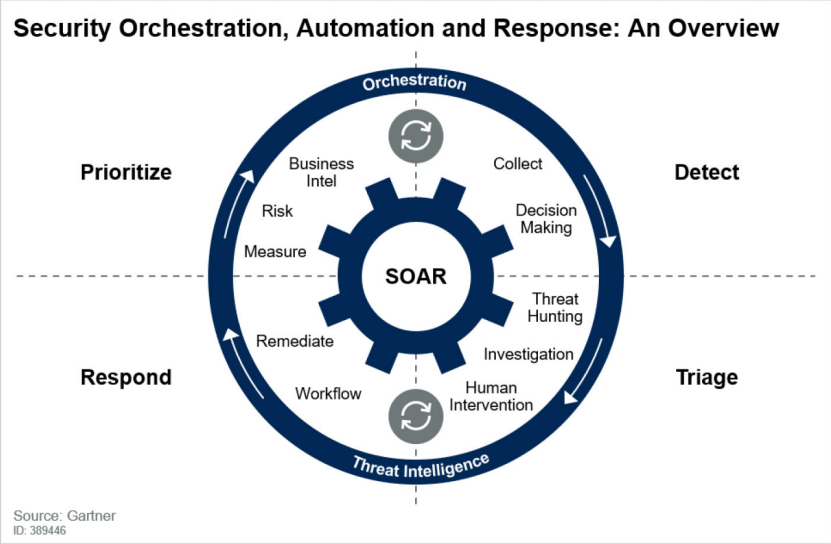

Figure 3.6.: Security Orchestration, Automation, and Response (SOAR)[35]

The security orchestrator operates from the Application layer with a set of customized applications dictating security policies for the underlying control plane and in return the data plane. Automating repetitive tasks to force multiply efforts and focus on mission-critical decisions, reducing dwell times with automated investigations and faster response times, integrating existing security infrastructure together are some of the key functions. A hybrid network of virtual and physical elements also uses security orchestration as described in **[[34]]**, which is responsible for managing security services provided via a Security Service catalog that triggers the lifecycle management of the Security Service, monitoring status, collecting Security Service related Key Performance Indicators (KPIs) and decision making.

3.1.5. Data Encryption

Data encryption is the process of converting plain text into an unreadable cipher text format which helps in protecting the confidentiality of digital data either stored on computer systems or transmitted through a network such as the internet. While a symmetric encryption uses a single password to encrypt and decrypt data, asymmetric encryption uses two keys for encryption and decryption- firstly the public key shared among users that encrypts the data; secondly the unshared private key that decrypts the data. Virtual Private Network (VPN) exploits the public network infrastructure such as the Internet to send and receive data but also ensures secure communication path for reliable data transmission between the sender and receiver. VPN protocols provides additional security layer for data transferred on communication networks.

IPSec protocol

IPSEC is a layer-3 VPN protocol for secure communication ensuring end-to-end security of data transfer across the network through privacy and authentication services. IPSec encapsulates packets by protecting the payload with encryption algorithms along with which the protocol group consists of an encapsulating security payload (ESP), an authentication header (AH) and a key management model. The CIA triad model was presented in [**1**], and the IPSec protocol complies with it by providing Integrity and Authentication through the IP AH and Confidentiality through the ESP. Access to the system is controlled by application-layer key management scheme including both public and private key-based systems, where the key exchange is based on the IKE (Internet Key Exchange) that facilitates the management and maintenance of resources in IPSec[36]. The architecture comprises of two IP header constructs, the authentication header (AH) and the encapsulation security payload (ESP), along with the Security association (SA)[37].

- AH: The AH performs integrity checks by using a hash function and a secret shared key in the algorithm header.
- ESP: The ESP brings authenticity through source authentication, data integrity through hash functions and confidentiality through encryption protection for IP packets.
- SA: Before exchanging data, IPsec protocols use SA for communicating established shared security attributes such as algorithms to be used for encrypting the IP packet; hash function to ensure the integrity of the data between parties using the Internet Security Association and Key Management Protocol (ISAKMP).

The establishment of an IPsec connection takes place in two phases. In Phase 1, the two endpoints authenticate one another and negotiate keying material resulting in an encrypted tunnel for negotiating the ESP security associations. In Phase 2, the two endpoints use the secure tunnel created to negotiate ESP SAs that are used to encrypt the actual data passed between the two endpoints.

GRE tunneling

Generic Routing Encapsulation (GRE) is a layer-3 VPN tunneling protocol and encapsulation standard that is used to mainly encapsulate IP packets for transmission. The GRE provides point-to-point private connection creating reliable and secure communication path where original data packets are encapsulated inside GRE header that eliminates vulnerability[38] during packet transmission over unsafe public networks. On the near end (sender) of the tunnel, data packet is received by the GRE endpoint routers which encapsulates with the GRE header along with the destination address of the tunnel. On the far end (receiver) of the tunnel, the receiver end routers de-capsulates the packet and delivers it to the desired destination. The tunnel can be used for the encapsulation with any OSI layer-3 protocol.

GRE-IPSec tunneling

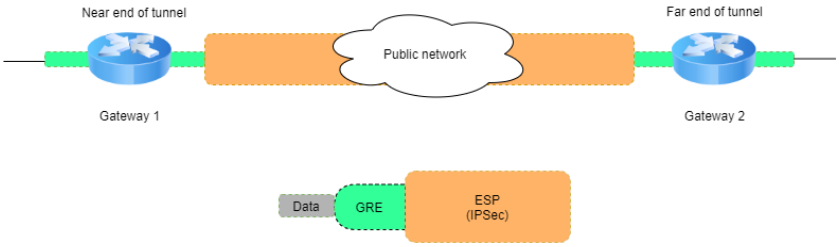

Figure 3.7.: GRE-IPSec tunneling

GRE is a powerful point-to-point tunneling technique however it does not provide strong security features like encryption, authentication, and sequencing as in the case of IPSec. In [38], an analysis of site-to-site and remote access VPN protocols are presented. The performance of the protocols in terms of throughput, RTT, Jitter and security mechanism is recorded and the results show that GRE is suitable for time sensitive and bandwidth sensitive application whereas IPSec is better suited for security sensitive applications. A combination of GRE and IPSec forms a safe data encryption method for transferring data packets over public networks. As depicted in fig 3.7, GRE over IPSec tunneling encrypts both data payload and routing protocols that is transferred between site-to-site gateways. In fig 3.8, data packet encapsulated first with a GRE header and further with the AH and ESP header is depicted.

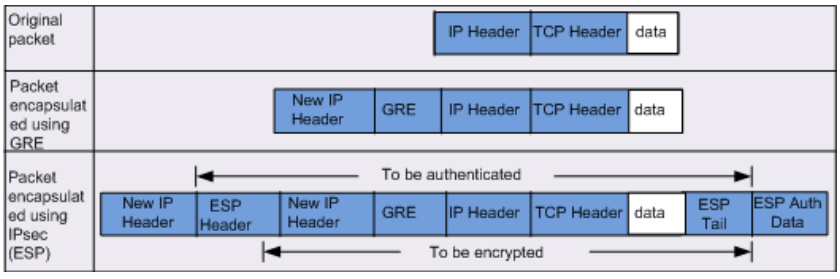

Figure 3.8.: GRE-IPSec header encapsulation on data packet

3.2. Network Topology Design

3.2.1. Traditional infrastructure vs. Cloud computing

Traditional IT infrastructure constituting dedicated hardware resources connected via a network through an on-premises server was the preferred choice for business companies and organisations, until a decade ago. The method required multiple hardware equipment to be installed, maintained and monitored for tasks, owing to the growing space and cost of physical infrastructure. On the other hand, the novel approach of cloud computing aims at sharing of resources that proved to be a breakthrough and several organisations adopted this method through the last decade. Typical IT infrastructure deploys dedicated resources such as servers, network, storage and operating systems on individual systems while cloud computing furnishes resources as an on-demand service to the end user. Further, based on technological needs requirements can be met as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) giving more flexibility to businesses. The computing models of private cloud offers services aimed at only members belonging to an organization, public cloud enables accessibility to common people from any location whereas hybrid cloud is a combination of both the private and public cloud. Thus with its varied features of computing models and a continuous effort towards improving cloud security standards to ensure disaster recovery, companies are offered greater degree of flexibility with respect to accessibility and sharing policies. The most striking feature of cloud computing lies in its efficiency envisaging a continuous backing up of data that ensures a hassle free way of data restore to end users as well as enterprises. The very idea of cloud conforms to accessibility of data and applications from any internet enabled device from anywhere in the world. It supports usage of diverse and multi-vendor hardware resources along with a selection of pre-built tools. Another distinguishing feature of the cloud from the traditional approach is its scalability where more resources can be effortlessly added on top of existing infrastructure and is cost-efficient practice for oscillating and unpredictable workloads.

The scope of this thesis is primarily based on the principles of SDN and NFV that gives rise to an agile, responsive and flexible network. Hence in this case, using the cloud to host the hardware and software resources is the suitable choice of infrastructure to harness the power of virtual computing. There are several choices available in the marketplace such as Microsoft, IBM, Amazon, etc. that offer public cloud solutions. Private cloud can be implemented using infrastructure and software from companies such as OpenStack, CloudStack, VMWare, etc. A comparative study between OpenStack and CloudStack in [39] presents a similar general architecture for both the middlewares. Although it mentions higher complexity in installation of OpenStack, a better overall stability as recorded in the performance analysis. Further, a comparison between OpenStack and VMWare in [40] presents a better scheduler performance owing to the Distributed Resource Scheduler (DRS) in the latter case. However, private cloud and OpenStack in particular, discussed in detail in the next sub-section, is well suited in the scope of this thesis due to features such as end user access and image management.

3.2.2. OpenStack and services

OpenStack is an open-source cloud operating system controlling large volumes of computing, networking and storage resources from a datacentre. In 2010, it started as a joint project between NASA and Rackspace hosting to handle large data volumes. As a cloud middleware, it helps with setting up the environment and handles middle ground of provisioning resources between the cloud resources and client. OpenStack comes with multi-tenant architecture that can support multiple tenants- several users or organisations from a single instance of a running software or hardware. It is a package of different components each of which is primarily responsible for interrelated services such as computing, networking, storage among others and combining such different instances each tenant can spawn several Virtual Machines (VMs) for dedicated tasks. The message queue server, RabbitMQ handles communication between the components and overall integration is performed by Application Programming Interface (API) as presented in [41]. In the scope of this thesis, the designed security function is hosted in the OpenStack environment. The detailed description of each service of the platform is beyond the scope of this work but an overview [42] of the most prominent services used for the design is depicted in fig. 3.9 and briefed below. At the onset, the end user is first presented with the OpenStack graphical interface, its dashboard called *Horizon*. Components are available as API to administrators but the web based GUI makes it a simple customized framework that provides access, manage, automate functions to the cloud resources and also allows third party tools as plug-ins. Apart from the dashboard, the main features used while developing the security function are the compute service– *Nova*, the networking service– *Neutron* and the image service– *Glance*.

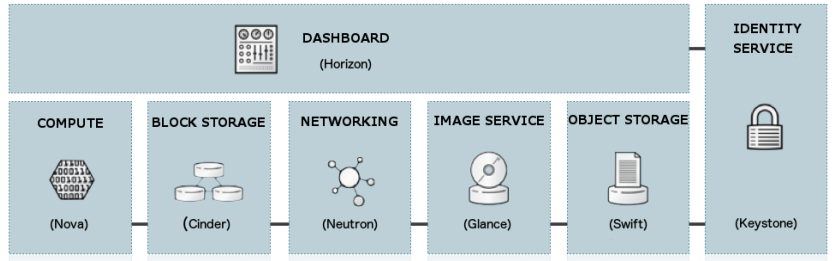

Figure 3.9.: OpenStack service overview

Nova is the core component of the OpenStack project to handle deploy, manage and automate large number of virtual machines and instances for computing tasks. It uses existing hypervisors based on a shared-nothing architecture [43] and has no virtualization software of its own. Using hypervisor HyperV, > 6 VMs were used in the thesis for designing the security function. *Glance* is the store for images with the function of managing virtual copies of hard disks which are used as start-up copies to deploy new VMs on the cloud. Images can be stored in any available format on OpenStack and for the security function Ubuntu 16.04.6 LTS and CentOS7 was uploaded and used in the QCOW2 format. *Neutron* is the networking component for managing networks, IP addresses and to ensure a congestion free environment. For the security function, both public and private networks with specific

sub-nets were used. Also, security rules such as IP protocol, port range and direction of traffic were set to streamline network traffic. *Keystone* is the identity service used to manage user profiles, service catalogues and endpoints with different authentication mechanisms such as token-based, password type. During this implementation, a user account was created on OpenStack and the service was also used for migration of VMs between different *Nova* hosts. Apart from these features, *Cinder*- the block storage component aims at managing the creation, attaching and deletion of volumes to instances whereas *Swift*- the object storage component takes care of save and retrieval of data, mainly static, in the cloud environment.

Research Cloud

The available in-house OpenStack platform together hosts 10 physical servers, top-of-the rack switches and several SDN switches as depicted in fig. 3.10, on which the designed security function resides as a Virtual network Function (VNF). The control node is the heart of OpenStack hosting nova, neutron-server and glance which are respectively responsible for the basic functionality of computing, networking and storage. Compute node hosts hypervisor that is responsible for provisioning Virtual Machines including a Nova compute agent and a Layer-2 agent (Open V-Switch (OVS) in this case). The network node is another core component taking care of connectivity. It hosts a Layer-3 agent for inter-network routing, handles the Layer-2 agent while also taking care of communication with external VMs and with different compute nodes through tunnelling. The cloud is managed by a Virtual Infrastructure Manager (VIM) including a management network which all the nodes are connected to and provides for internal communication; a tunnel network for tunnelling between networks and compute nodes; an external network for traffic meant for the outside world. Along with Infrastructure as a Service (IaaS) functionality, OpenStack provides orchestration, service and fault management.

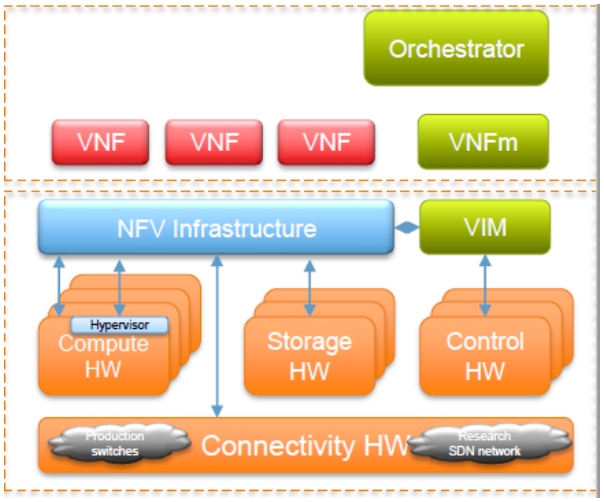

Figure 3.10.: Layout of the in-house cloud

3.3. Design choices

3.3.1. SDN Switch: Open vSwitch

In the previous chapter the research cloud was presented. For design of the SDN-based security function to be hosted in this cloud environment, the rational choice would be to opt for a virtual switch. Open vSwitch(OVS) is a multi-platform, open source software switch designed to be used as a vswitch (virtual switch) in virtualized server environments that forwards traffic between different virtual machines (VMs) on the same physical host while also forwards traffic between VMs and the physical network. Closed source virtual switches mostly operate in a single environment whereas a user chooses an operating system distribution and hypervisor for Open vSwitch environment thus making a design to be modular and portable. The vSwitch uses OpenFlow and the OVSDB (Open vSwitch Database) management protocol making it extensible by re-programmable[44] flow programming model where a controller responding to traffic installs microflows that supports OpenFlow instruction fields. OpenFlow switch consists of one or more flow tables, a group table[43] to perform packet lookups and forwarding along with an OpenFlow channel to an external controller as depicted in fig 3.11(a). The switch communicates with the controller and the controller manages the switch via the OpenFlow protocol.

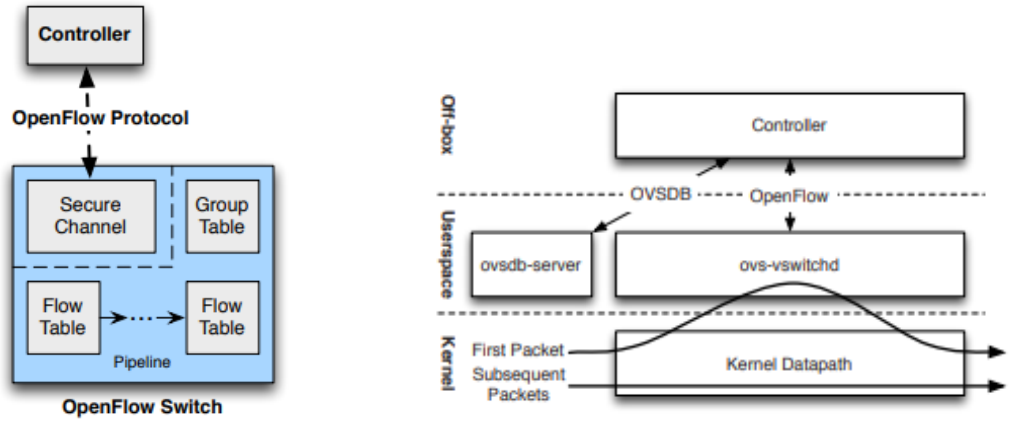

Figure 3.11.: (a) and (b): Open vSwitch components and interface

As presented in [45] and fig 3.11(b), two main OVS components module direct packet forwarding, a userspace daemon ovs-vswitchd and a datapath kernel. The ovs-vswitchd module instructs datapath on how to handle packets of a particular type which then simply follows instructions called actions, given by ovs-vswitchd that specifies physical ports or tunnels on which to transmit the packet. Actions may also specify packet modifications, packet sampling, or instructions to drop the packet. When the datapath module has not been instructed what to do with the packet, it delivers it to ovs-vswitchd which determines how the packet should be handled and also informs the datapath to cache the actions for handling similar future packets. OpenFlow allows a controller to add, remove, update,

monitor, and obtain statistics on flow tables and their flows. It also diverts selected packets to the controller and injects packets from the controller into the switch. OpenFlow and version 1.3 in particular is used for our design. Southbound APIs are issued by the controller to fetch switch statistics. The ovs-vswitchd module receives OpenFlow flow tables from an SDN controller, matches any packets received from the datapath module against the tables, gathers applied actions and caches the result in the kernel datapath. The OVSDB management protocol is responsible for interacting with OVS database for the purposes of managing and configuring OVS instances. The following OVS ultilties are used to configure and monitor switches in the security function design.

- ovs-vsctl: The ovs-vsctl program is a utility for querying and configuring ovs-vswitchd and has been extensively used in our design. It connects to an ovsdb-server process that maintains an OVS configuration database and based on input commands it queries and applies changes to the database . If there are applied changes, by default it waits for ovs-vswitchd to finish re-configuration.
- ovs-ofctl: The ovs-ofctl program is a command line tool for monitoring and administering OpenFlow switches. Also, it can show the current state of OpenFlow switches, features, configuration and table entries. This program has been used in our design to install and alter OVS flow rules.

The flow table entry is identified by match fields and priority which when considered together identify a unique flow entry in the table. Flow entry that doesn't match all fields and has priority equal to 0 is regarded as a table-miss flow entry. A flow is installed for handling similar future packets and packet matching is as explained in fig 3.12[44].

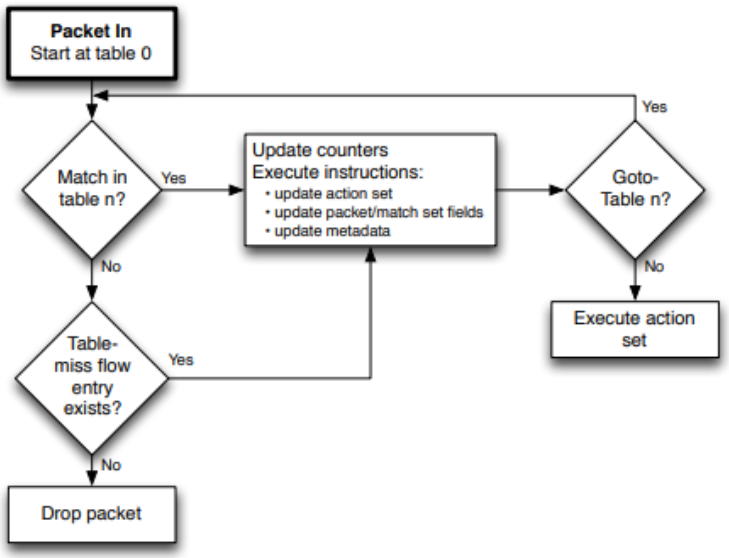

Figure 3.12.: Packet flow through an OpenFlow switch

3.3.2. SDN Controller: Ryu

There are many different SDN controllers available today that can efficiently act as a strategic control point in the SDN network. Ryu, a component-based, open source SDN framework programmed in Python acts as the brain in our security function. Ryu's software components with well defined API makes it easy to create new network control and management applications. It is a well supported, targeted controller with a concise technical documentation. More importantly, it has full OpenFlow Support and its Southbound interfaces allow communication with SDN switches such as OVS. Ryu also supports multiple southbound protocols for managing devices such as OpenFlow, NETCONF and OF-Config. The Application layer can deploy network policies to data planes via well-defined northbound APIs such as REST. The controller provides simple supporting infrastructure to deploy code that can utilise the platform as desired and also well-defined API to change the way components are managed and configured.

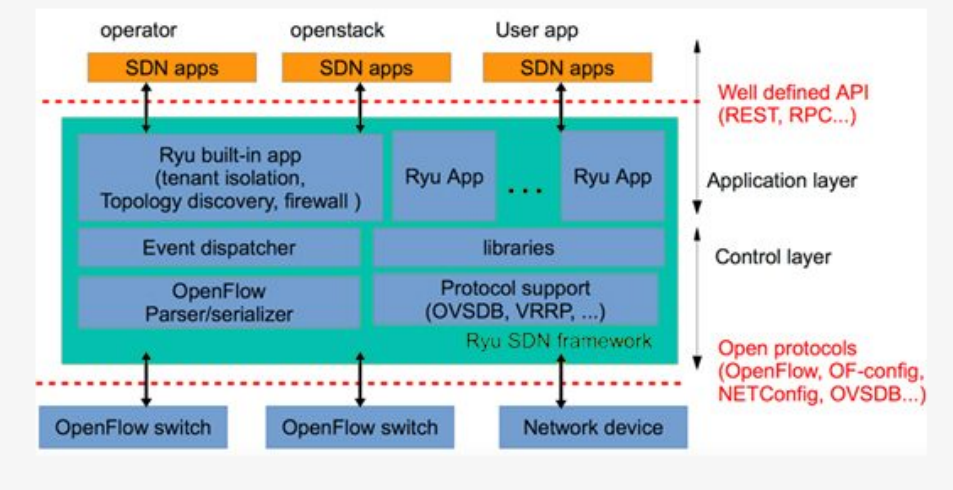

Figure 3.13.: Components of Ryu SDN controller

The components of Ryu are as depicted in fig. 3.13. The main executable is the bin/ryu-manager[46]. The central management of Ryu are the base components that loads built-in applications and route messages among other Ryu applications while also deploying new applications. The OpenFlow controller is a main component that handles connections from switches while also generates and routes events to appropriate Ryu entities. A decorated Python method in Ryu acts as an event handler and RyuApps receives instances from its event class[47]. The method's dispatcher argument specifies one or more of the following negotiation phases and accordingly events are generated for the Ryu event handler.

- Handshake Dispatcher: Sending and waiting for hello message
- Config Dispatcher: Version negotiation and send features-request message
- Main Dispatcher: Receive switch-features message and send set-config message

- Dead Dispatcher: Disconnect from the peer or disconnecting due to unrecoverable errors.

Ryu hosts a Packet Library that allows to parse and build various protocol packets. Ryu OVSDB Manager library allows code to interact with devices speaking the OVSDB protocol that enables it to perform remote management of devices and react to topology changes on them. OVSDB library initiates connections from controller side. Ryu messages and structures are supported by OpenFlow version 1.3.

- Controller-to-Switch Messages: The controller sends a handshake feature request to the switch upon session establishment, a set config request message to set configuration parameters, message to configure table state and modify the flow table among other messages.
- Asynchronous Messages: The switch notifies controller with asynchronous messages when it sends a received packet to the controller, when flow entries are deleted, upon change of ports or during errors.
- Symmetric Messages: Messages exchanged between a switch and a controller when connection starts including the hello message, the echo request-reply message.
- Description of a port: Messages return details such as Port number, MAC address, state and current features of ports within the switch.
- Flow Match Structure: Matches keyword arguments with a compose/query API to build flows.
- Action Structures: Action structures indicates output of a packet to the particular switch output port or set it to group action, queue action among other things.

In the implementation of the security function, a Ryu controller app was customized. The app comprises event classes with functions such as a packet-in handler- to handle incoming packets; switch features handler- to install table-miss flow entries and an add-flow function. Together the functions perform actions such as GET Datapath ID to identify OpenFlow switches, install a flow to avoid packet-in next time, construct packet-out message and send it. Also, the Ofctl REST Service Ryu application was used in the implementation which provides REST APIs for retrieving and updating the switch stats. Retrieve API such as GET all switches, GET all flows stats and update API such as Add a flow entry among others were used.

3.3.3. Security Orchestrator: Phantom

Security Operation centers (SOCs) have several cybersecurity tools to prevent, detect and mitigate threats. Security orchestration is the process of integrating dissimilar SOC tools and workflows to deliver quicker, effective responses through task automation. Commercially the market offers several varied security orchestrator options to choose from. Swimlane, ThreatConnect, Fireye among many others provide Security Orchestration, Automation and Response (SOAR) platforms to develop incident response solutions. Orchestrators vary from one to many number of data sources for ingesting the incoming feed of security data while also including disparate tools to control incidents and to offer quicker response times. Also, for automating response tasks, tools offer playbook solutions that codifies mitigation plans while automatically pulling alerts from Security Information and Event Management (SIEM) environments and intelligence feeds. Graphical interface editors offer the chance of building customized playbooks while it also provides drag-and-drop functionality for certain pre-built actions. We have chosen Phantom, a popular security orchestrator available in the market to automate our designed security function.

Phantom enables SOCs to carry out investigate, degrade, contain and remediate threats in complex enterprise environments, enpoints and assets[48]. Security response analysts can view, act on incidents and various attributes, manage related information through the life-cycle of an incident, develop the next generation of automation strategies. Primary component of the platform is the Visual Playbook Editor (VPE) that allows to construct simple but sophisticated playbooks offered in Python making it easier to choose other components of the security function. The VPE generates behind the scenes python code in real-time as the tool administrator builds playbooks graphically. The workflow in Phantom is as depicted in fig. 3.14.

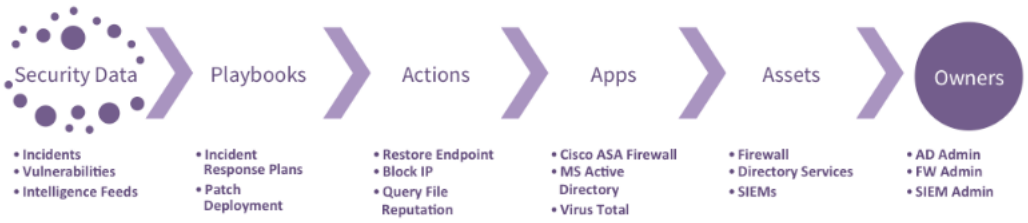

Figure 3.14.: Workflow in security orchestrator: Phantom [47]

The implementation of our automated security function is named *sarnet* and a detailed presentation will be made in Chapter 5. Playbooks are used for automated decision making and executing various kinds of actions as a response to an incident. Following the work flow as in fig 3.14., Phantom needs a source of security events to function as a fully automatic systems and this can be in the form of SIEMs, threat intelligence services, email, ticketing systems, the sample generator, or through a REST API. This input data is used to trigger our mitigation plan designed in the form of python based playbooks which basically automates running several actions. Containers are the top level data structure that playbooks operate

Table 3.1.: Workflow in Phantom

Layers	Description
Security Data	Phantom absorbs input data as incidents, vulnerabilities and intelligence feeds. Splunk is the primary data ingestion source and also accepts python input feeds.
Playbooks	Security operations plan codified as python scripts that implements logic to make decisions and call necessary actions.
Actions	Pre-defined generic and higher level primitives offered and available as code blocks including "start", "end".
Apps	Python modules that are extensions of the platform to communicate between vendor-specific product or service. Apps publish list of executable Actions and the supported type of Asset for it.
Assets	Physical or virtual infrastructure devices that execute actions upon interaction with Apps.
Owners	Designated as approvers on Assets. When Actions are performed on Assets owners receive approval requests.

on in Phantom and they are used to group objects together that are related to a particular event that can include actions, playbook runs, results and files among other information. On triggering a playbook, containers are required for tracking what the action belongs to, along with saving output and results returned from running the action. Artifacts are individual items of a container and as associated objects serve as evidence related to the Container. In case of *sarnet*, the Phantom security data type 'incidents' is ingested as a REST API and this input is fed to an existing container 'event' to perform actions such as verification of node under attack, traffic cloning and traffic encryption among other actions.

Phantom installation comes shipped with several pre-installed apps written for specific products. Every action of an App specifies regular expression for a version of the product that it supports incorporating many popular hardware and software-based network devices on to the orchestrator while it also provides options to build apps for unsupported and new devices. For implementing *sarnet*, few pre-installed apps were used while a new app *Ryu* was also built for automated network communication between the Phantom platform and the third party SDN-controller RYU. The Phantom architecture and components are depicted in fig 3.15. With the help of a python-based connector module the app sends orchestration control signals to the SDN control plane for performing underlying tasks. Once action is invoked, the Phantom Action Daemon invokes an executable that imports the appropriate module via an IPC connection as shown in fig 3.16. Further, the workflow needs Assets which is a service or piece of equipment that Phantom can communicate with or control and provides apps the functionality to manage and operate on devices. The configured instance assets provide with the information needed to communicate with apps such as IP address,

username and password. For configuration of an input asset, an existing container type or a new one can be added and items coming in via that asset will be labelled with the container label.

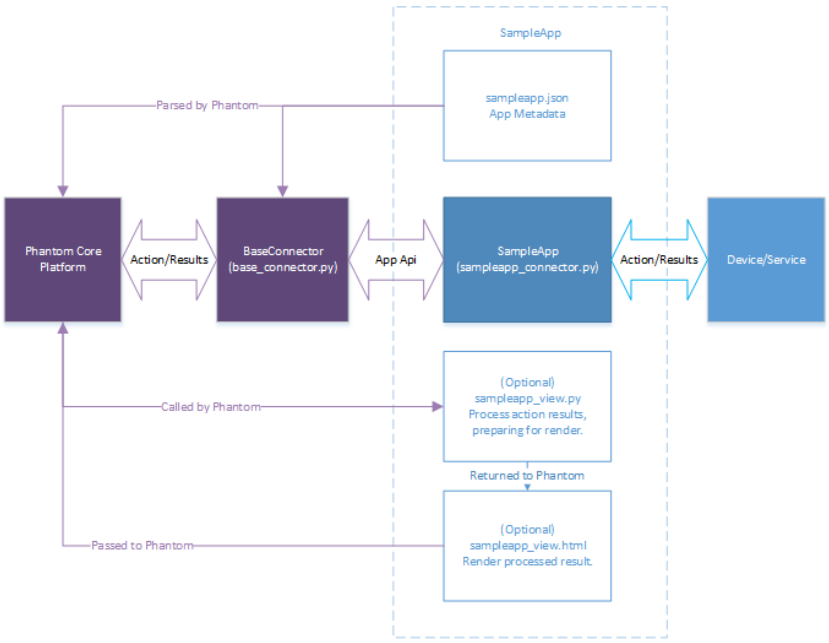

Figure 3.15.: Phantom architecture and components

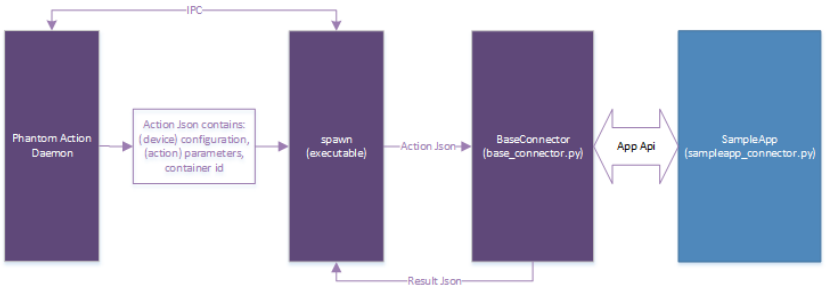

Figure 3.16.: Phantom connector module

After the configuration, existing playbooks or customized playbooks are built and incorporated. Phantom provides built-in functions as filter or actions to schematize a incident response plan as depicted in fig 3.17 and its actions are briefed in table 3.2. To run a playbook, the playbook settings have to be configured specifying type of events, container labels, logging, etc. and it has to be saved. Phantom also provides a GUI based python editor and for generic actions it comes with a skeleton code block. Saved and active playbooks can be run in the GUI based debugger that logs parameters sent for each action and also the returned message metadata between between subsequent playbook blocks.

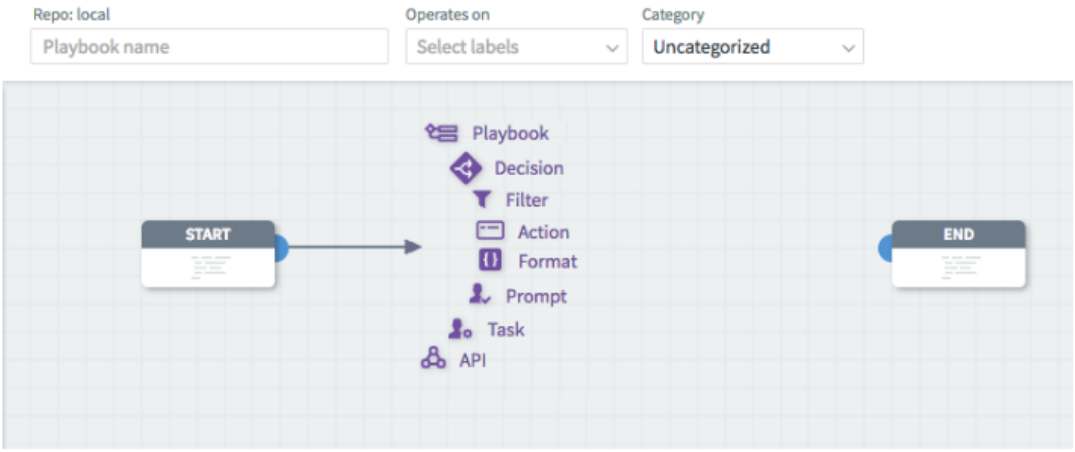

Figure 3.17.: Phantom Visual Playbook Editor

Table 3.2.: Visual Playbook Editor actions

Layers	Description
API	Utility action to set parameters of the container based on the information fed to the playbook. For example: set severity, set status etc. for a container.
Task	Sends a message to a Phantom user or group from a existing list and wait for acknowledgment.
Prompt	Adds human interaction to playbook and assigns prompts to individual users where actions can be completed, delegated or let to be timed out.
Format	Allows to craft custom strings and messages from intermediary blocks to form coherent text. For example: create ticket, draft an email contents.
Action	Can be investigate, contain, correct or generic actions that uses configuration settings from assets and apps. For example: GET datapath ID of switch, lookup IP address.
Filter	Filters output data of the intermediate, predecessor block to narrow down cases for the further part of incident response.
Decision	Blocks control program flow based on comparison of artifact data, container properties, date functions and action results of the if/then logic.
Playbook	Several playbooks can be chained together by calling other or generic response playbooks from within a playbook.

3.4. Use Cases

In a typical SDN network, the control plane issues control signals to the data plane for facilitating automated network traffic flow. At the heart of our set-up is the combination of SDN controller, Ryu and the SDN switch, Open vSwitch respectively forming the control plane–data plane. The communication between the devices are handled by APIs of OpenFlow and Open vSwitch Database (OVSDB) Management Protocol. On initiating Ryu, the OVSDB manager library spawns a server on the controller side listening on the designated port number *6640* but does not commence any active connection. To connect to Ryu, the OVS switch using its service ovs-vsctl then connects on its dedicated port number *6633*.

3.4.1. Normal traffic scenario

To depict a normal scenario, communication between a web based server-client is emulated on the data plane in the experimental setup, for which a python implementation of Simple-HTTPServer[28] is used on port number *80*. For demonstration, Linux network namespaces and netns in particular are used [29]. The network namespace is a logical copy of the parent network stack and in our case the namespaces exists only when the OVS bridge exists. But namespaces come with its own interfaces, routing rules and firewall rules. Hence, in principle, communication with network namespace is similar to that with any another device within the network. The web based server-client running on two different namespaces is installed on the data layer and along with its control layer is as depicted in fig 3.18. Upon establishing connection between the RYU and the OVS switch, the combination of ovs-ofctl service and southbound APIs such as OpenFlow, OVSDB install flows on the switch side. In the expected normal scenario, the communication between the client and server is handled by the flows installed on the forwarding table of the Open vSwitch as instructed by the Ryu controller.

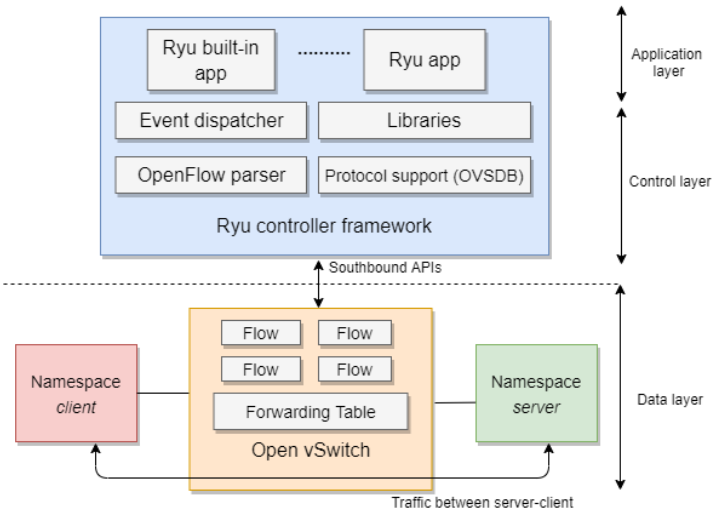

Figure 3.18.: Experimental setup in normal scenario

3.4.2. Expected attack scenario

For the design of the security function a specific cyber-attack type has been assumed. Data ex-filtration attack was defined in Chapter 2 and is known to be a type of attack involving the malicious activity of copying, retrieving or transferring of data by an unauthorized user. At the onset of the attack, the person intending information theft may possess stolen valid credentials and pose as an authorized user. When not in possession of required credentials, the attacker resorts to continuous effort to steal restricted data which is regarded as Advanced Persistent Threats (APTs). In both the cases, there are possibilities for the system in question to respond and release sensitive information as per the made request without the knowledge of an ongoing attack. In the scope of the thesis, it has been assumed that such an attack has already taken place. As APTs involve continuous efforts to break into a system, high traffic volumes can be encountered which is rather unexpected within our network and is hence deduced as malicious activity. In the detection phase of our security function, this has been addressed using machine learning algorithms with the help of high volumes of custom network traffic and is presented in the next chapter. After detection, in the subsequent phase the aim of the desired security function is to respond to the malicious activity by commencing mitigation without providing a hint of it to the attacker. This means that while the mitigation activity takes place in the background, the attacker in our model continues sending malicious requests to the system. Due to this continued malicious activity, the desired security function aims at cloning the malicious traffic and diverting it to another secure node within the system for further investigation. However, the destination safe node as depicted in fig. 3.19, may reside well within the same network or in an external network. In organizations operating from multiple locations, for the benefit of cybersecurity teams, the malicious traffic may have to be forwarded to secure nodes across continents. In the course of the path taken by the malicious traffic, assuming that public networks maybe encountered the mitigation plan also aims at encrypting the already infected traffic so as to avoid any further compromise. At the secure node, advanced facilities to further investigate malicious traffic such as deep packet inspection, sandbox or quarantine can be provisioned, however this detailed inspection shall be beyond the scope of this thesis work.

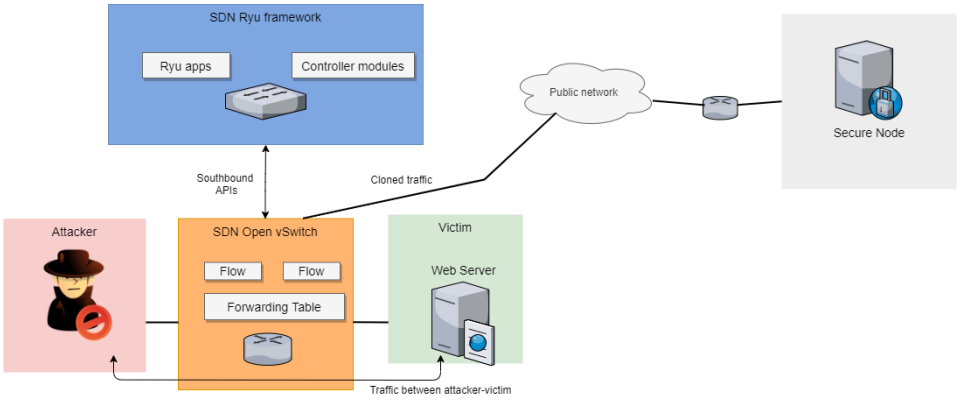

Figure 3.19.: Experimental setup in expected attack scenario

1. Use Case: Traffic cloning and encryption

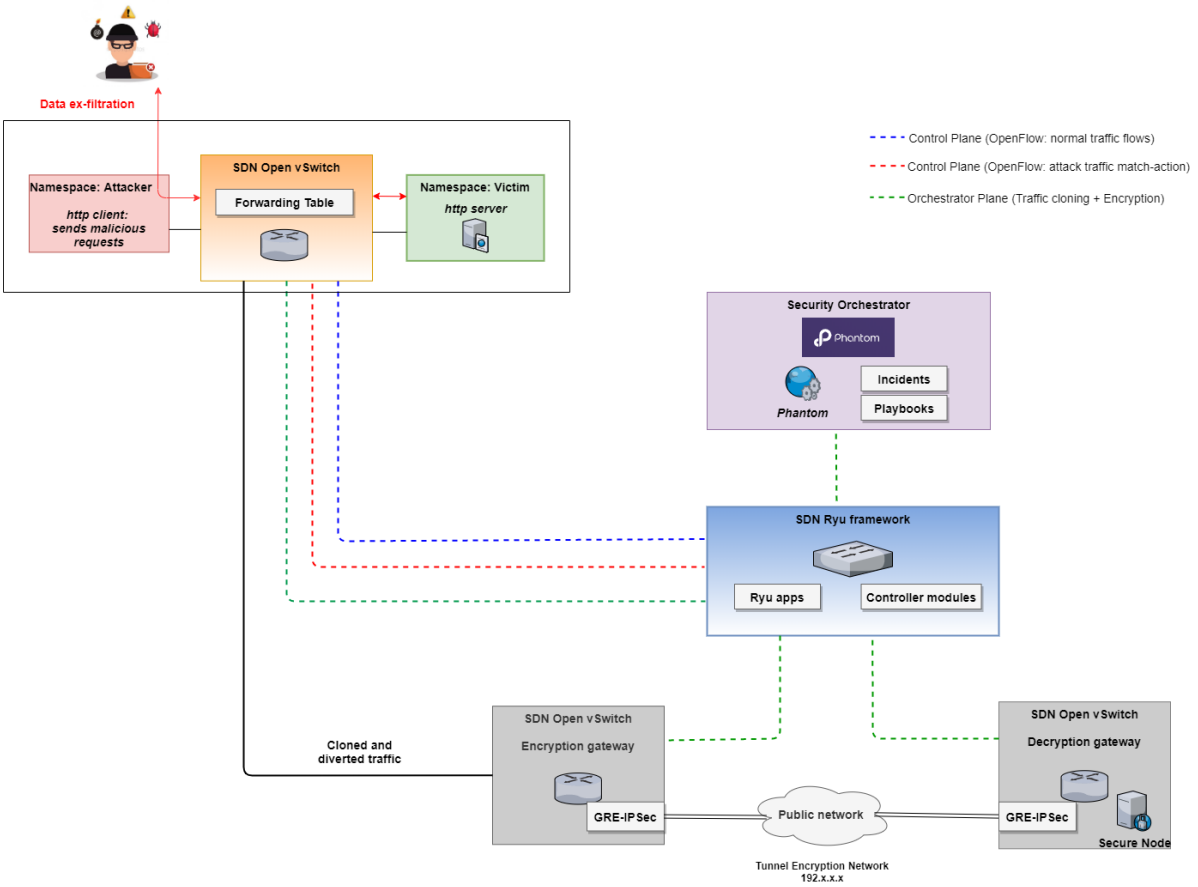

Figure 3.20.: Use Case 1: Network topology

The first use case, is a rather simple one where the aim of the security function is to clone the malicious traffic and to divert it to a secure node for further analysis. Considering that this traffic on its path from the infected node to the secure node maybe routed over public unsafe network, the aim is also to encrypt the malicious traffic to avoid any possible further compromise before it can be analysed. As a pre-requisite for the initiation of the security function, an active connection between the data plane and the control plane respectively represented by the Open vSwitch and Ryu controller as shown in fig 3.20 is established involving successful exchange of handshake messages. After this step, the control signal (blue line from fig 3.20) from Ryu is expected to install two table-miss flow entries with different priorities for traffic routing on the Open vSwitch. The first and the lowest priority instruction matches all entering packets and its output action is specified to the controller port. This instruction directs for further installation of flows from Ryu when normal packets are encountered on the data plane. The attacker-victim namespace or the http server-client connected to the switch, transmits http i.e., TCP traffic between them. For the purpose

of demonstration, TCP traffic is assumed to be non-indigenous within the network and hence is regarded as malicious. The second instruction has the highest priority so as to handle malicious traffic. With match-action rules for TCP packets set on Ryu, the control plane installs flows (red line from fig 3.20) on the data plane upon encountering malicious traffic. This flow is responsible for normal flow of traffic between the attacker-victim as well as for traffic cloning and diversion. Further, on separate instances of OpenStack VMs encryption and decryption gateways are set. The gateways maintain active connection with the Ryu which in turn install flow rules for traffic routing between their respective network interfaces. The traffic forwarding between OpenStack VMs brings cloned traffic to the first network interface of encryption gateway and on the second network interface the near-end of GRE-IPSec is established. The encrypted malicious traffic that is to be sent over public network is demonstrated using a dedicated tunneling network from OpenStack. On the first network interface of the decryption gateway is the far-end of the GRE-IPSec tunnel and on the second network interface packet collection facility is enabled. The essence of this work is to automate all the above mentioned actions with minimal human interaction. For this purpose the security orchestrator, Phantom is introduced. The Ryu communicates with the Phantom applications using REST-based northbound API (green line in fig 3.20). The security function thus automates functions of verifying the system under attack, setting up encryption and decryption gateways, cloning malicious traffic and diverting it over encrypted tunnels. This automation is achieved with the help of Phantom playbooks and is presented in detail under chapter 5, mitigation.

2. Use Case: VM Live migration with traffic cloning and encryption

Use case 2 is similar to the earlier use case but includes an additional function making it slightly more complex. The traffic cloning, diversion and encryption are also present in this case along with a live migration of the VM hosting the SDN Open vSwitch that is attached to the attacker and victim namespace. For further understanding, Linux network namespaces is depicted in fig 3.21, that is shown attached to an Open vSwitch.

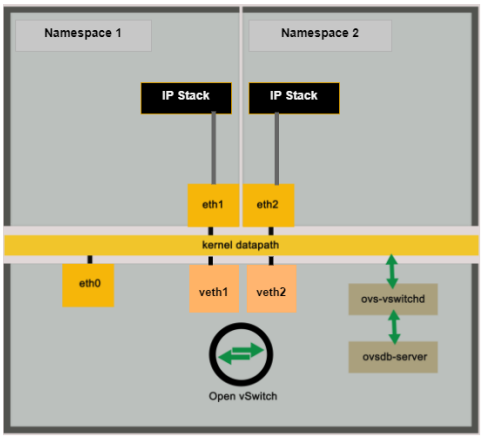

Figure 3.21.: Linux network namespaces

The OVS vSwitch daemon connects the ovsdb-server to the Linux kernel datapath which is also attached to the network interface eth0. The attacker and victim namespace are represented by custom namespaces, Namespace 1 and namespace 2 which are logical copies of the network stack with its own IP routing rules[29] and exists only when the OVS bridge and its interfaces exists. The network interfaces of the namespaces, eth1 and eth2 are attached to virtual interfaces veth1 and veth2 via the kernel datapath. Similarly, we represent the attacker-victim model in our demonstration and the motivation for such a setup is to perform live migration of the VM it exists on. The goal of this exercise is to obtain a virtual copy of the active VM and transfer it to the secure node placed behind the decryption gateway with facilities for deeper analysis of the attacker behavior. For this purpose, the Virtual Infrastructure manager (VIM) which in this case is the OpenStack controller module is introduced as presented in fig. 3.22. Further, aiming to automation this function, the mitigation plan triggered from Phantom facilitates a snapshot function that creates and transfers live VM snapshot of the switch under attack to the secure node. As a result, this plan yields the security administrator a secure virtual copy of the live infected VM along with a collected dump of malicious traffic.

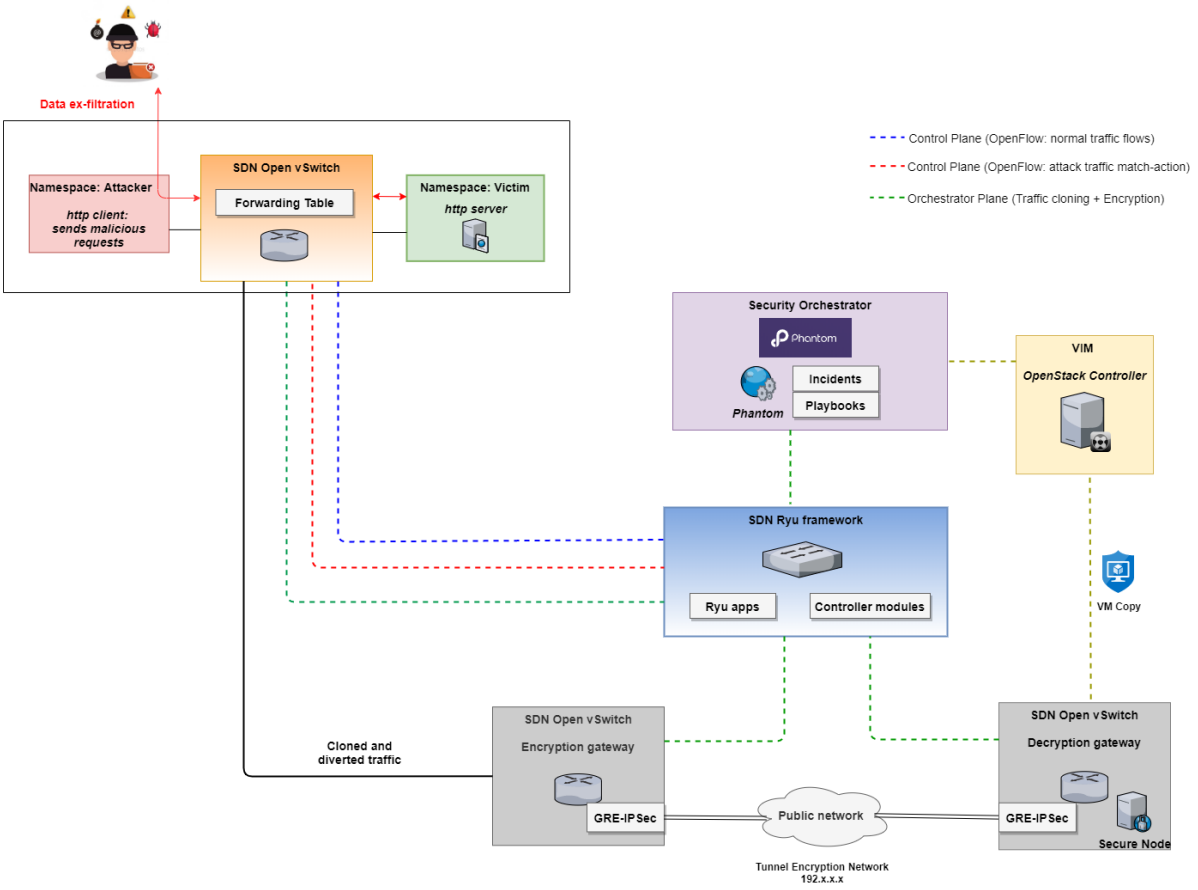

Figure 3.22.: Use Case 2: Network topology

4. Mitigation: Implementing security function

4.1. Security Orchestrator: Phantom

4.1.1. Configuring Phantom workflow

Phantom is shipped as an OVA file that is suitable for using it as a VMWare instance. The security function in our case, is however implemented within OpenStack (Rocky) environment and such a file is not sufficiently supported. Hence the Phantom image was extracted from the OVF package and installed on a CentOS Linux release 7.5.1804 based VM with an ephemeral back-end. Further, a python environment was prepared using the Phantom install script and an user account was configured. Many or all of the the supported apps can be installed together along with Phantom installation. A snapshot of the VM was converted as a Glance image to be used for any future use of Phantom instance on openStack. Phantom provides the web-based GUI playbook editor and for accessing it via a web browser, the NGINX 16.1 web server was configured as reverse proxy on the OpenStack Phantom instance. The Phantom workflow for mitigation plan design is configured as listed below.

- Data sources: The orchestrator is fed input data through Bash scripts and Python scripts incoming from the SDN controller. Additionally, to trigger playbooks REST APIs are used.
- Playbooks: Python playbooks are built using the Visual editor for each of the two defined use cases while also different playbooks are chained together. A detailed presentation is made in the next sections.
- Add container: A new container is added to the events queue with the label 'incidents' specifying artifacts such as ID number, name, description, created time and its status is set to 'open'. Sensitivity and severity are used to classify containers to denote the impact of incoming expected data and are set to 'amber' and 'medium' respectively. Actions are performed within Containers.
- Add actions: Actions are taken against an Asset to enable it to perform a task or to retrieve information from. 'Investigate' actions such as geo-locate IP, fetch hostname and 'generic' actions such as send email are used in the playbooks.
- Apps: The platform offers built-in apps. SMTP, ipinfo.io were used off the shelf whereas a customized app was built to support SDN controller Ryu.
- Add assets: Asset can be dynamically added to the system and Phantom Apps execute an action on a user-defined asset. In our case, user credentials, SSL port and server

name(for configuring an SMTP server), URL of the SDN controller(to fetch datapath ID of switch) are examples for user-defined assets.

- Owners: Owners are responsible for the management of assets within the organization and receive an approval request to execute a particular action on a specified asset. This facility is not used in this implementation.

4.1.2. Building a Phantom app: Ryu

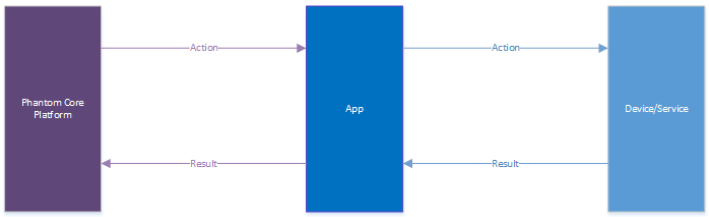

Figure 4.1.: Phantom App architecture

Phantom Apps provide a mechanism to extend the platform capability by adding connectivity to third party technologies to execute actions. As depicted in fig 4.1, the app is instructed by the platform to perform an 'action' which then translates it into set of specific commands comprehensible by the device or service and vice-versa. The SDN controller, Ryu App was built as listed below.

- Using the App wizard- name, description and logo for the new app was updated. A URL asset corresponding to the IP intsance of Ryu was configured as *http://IP instance:port number*
- Actions to test connectivity and *GET dpid ID* to fetch datapath ID of the switches connected to Ryu was defined. A corresponding Action JSON, containing asset configuration and action parameters is created.
- Phantom Apps use JSON schema to define configuration and the App wizard generates a list of files including connector modules, JSON parameter files among other things. The base connector module was further customized using python and its member function 'handle action' is parsed which loads parameters from the Action JSON file.
- The configuration and parameter files was validated. The Phantom connector module scheme was shown fig 3.16 where the connector result JSON was created and sent back to spawn in case of an error to re-generate the correct executable file.
- Upon a successful compile, the action result JSON object is returned. App data paths pass down information about the data that needs to be presented to the UI layer.
- The assets and actions are made available for the visual playbook editor to incorporate into playbooks. Depending upon the action and the asset to execute on, one or more Apps can be chosen on the editor.

4.2. Use Case 1

In Chapter 3, Use case 1 has been defined with the aim to achieve traffic cloning and encryption in response to malicious activity on the Open vSwitch. The motivation behind traffic cloning is to duplicate malicious traffic which is to be further analysed at a secure node. Traffic encryption is performed to reduce risk of further compromise and safe transmission of cloned traffic to the secure node. At the onset, a connection between data plane and control plane is established with the exchange of handshake messages. At this stage, Ryu installs two table-miss flows on the Open vSwitch as shown in fig 4.2, for handling future incoming traffic. The lower priority (0) flow is the default packet-in flow and the switch feature handler of the Ryu installs this flow. As a special case, the highest priority(10) flow is installed to deal with malicious traffic. The switch feature handler's match-action looks for match fields of- packet type: tcp, packet destination: 100 and the output action for outgoing traffic is set to port 2 while traffic is also set onto port 5 thus achieving traffic cloning.

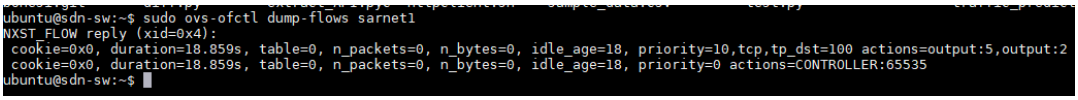

Figure 4.2.: Table-miss flows installed on the data plane

After the table-miss flows are installed, incoming packets are taken care by the Ryu packet-in handler. In the event of an incoming packet, the function parses the packet header and matches it for the tcp packet. If the packet type and destination matches, the traffic is treated as malicious and cloned traffic is forwarded to encryption gateway via port 5. Normal communication with port 2 continues as our aim is to keep the attacker uninformed about the mitigation actions. Ryu installs priority (1) flows for traffic intended between ports 1 and 2 as shown in fig 4.3. In case of any packet other than tcp, traffic flows normally within the switch-namespace environment.

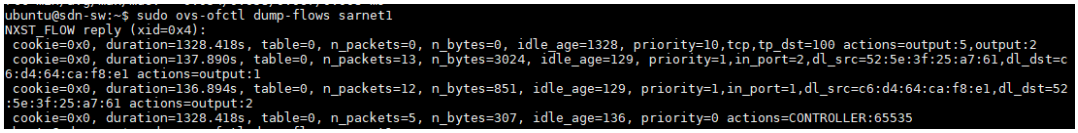

Figure 4.3.: Traffic flows for all incoming packets

Further, this cloned traffic is encrypted at the next subsequent node in our system using IPSec-GRE. However, the aim of this thesis is it to automate the mitigation action. For this reason, we introduce our mitigation plan as python-based security playbooks hosted on the security orchestrator, Phantom. The SDN controller communicates with the application layer via northbound-APIs which then triggers playbook at the onset of malicious activity thereby automating mitigation functions, traffic cloning and traffic encryption.

4.2.1. Playbook for automated mitigation

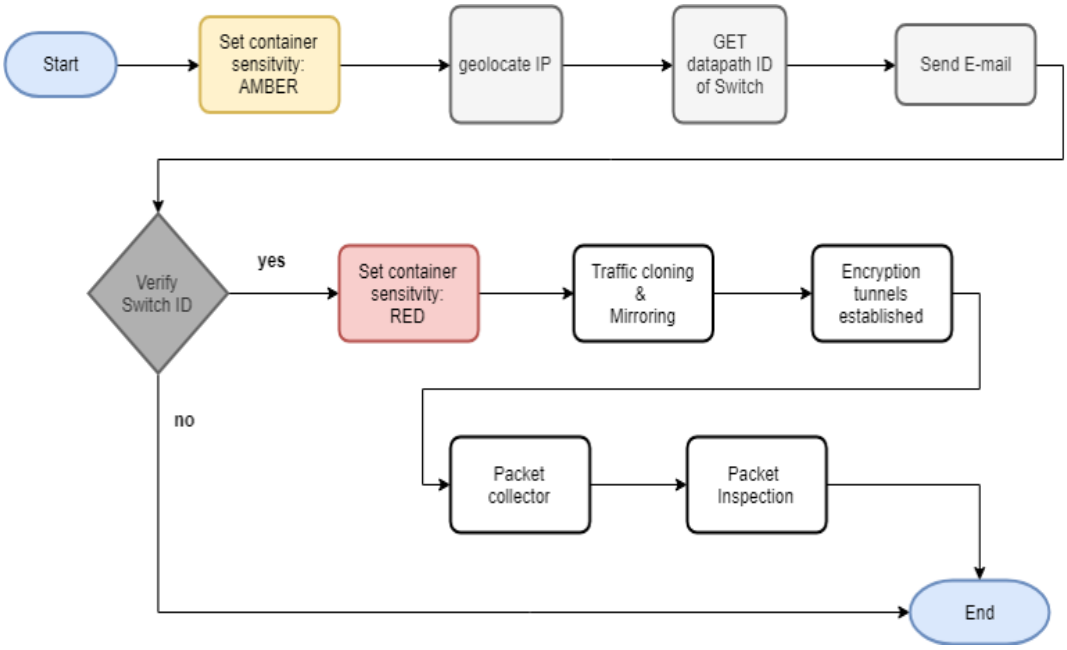

Figure 4.4.: Use case 1: Playbook flowchart

The scheme for the first playbook is drawn in fig. 4.4 and listed in detail below.

- Phantom platform supports RESTful APIs to trigger execution of playbooks. A HTTP POST from the control plane 'starts' the playbook run.
- Sensitivity - Phantom API block action is used to 'set sensitivity' of the container from 'white' to 'amber' at the start of the execution. This is done to set the sensitivity higher so as to notify the security admin monitoring the events queue of a possible attack.
- Geolocate IP- *Asset URL: http://ipinfo.io*
  The IP information web service URL fetches geo-location details such as co-ordinates, city, country of the possible attack inflictor. This app has been configured for the demonstration of this function only. It may fail in case of an IP spoof attack.
- GET dpid- *Asset URL: VM instance of SDN controller*
  The Ryu app is configured to fetch and store datapath IDs of all the switches connected to the controller. The Ryu ofctl service responds to northbound REST API used with the URI- /stats/switches.
- Send e-mail- *Asset: SMTP server, user credentials, SSL port*
  The Phantom SMTP app is configured to Gmail server and at this step triggers an email notification to the security admin informing of a possible attack.

- Verify switch ID- Before the execution of mitigation action, the Phantom decision block matches the datapath IDs of the three connected switches present in our network to the ones fetched by the 'GET dpid' function. If they do not match the playbook is terminated and only upon verification the mitigation plan is activated.
- Sensitivity changed- After the verification step, malicious activity in the system is confirmed and the sensitivity of the container is changed using the 'set sensitivity' API from 'amber' to 'red' for denoting the highest sensitivity in the event queue.
- Traffic cloning: The duplicating of and mirroring traffic onto an adjacent port of the switch is automated with the signaling to the Ryu ofctl service. The northbound API with the URI- /stats/flowentry/add is used to install the new flow on the Open vSwitch.
- Traffic encryption: The encryption of traffic is achieved by enabling traffic flows via GRE-IPSec enabled ports and creating a tunnel between the encryption and decryption gateway. The pre-shared key is set to secret.
- Packet collector: At the secure node located on the decryption gateway, affected traffic packets are collected using Wireshark and saved as a PCAP file.
- Packet inspection: The Wireshark file is uploaded to the vault of the Phantom container and available as an attachment. This collected malicious traffic file shall be used for any further analysis. This marks the end of mitigation and the playbook is terminated.

The flow of action and control signals between the data plane, control plane and the orchestrator plane is depicted in fig 4.5.

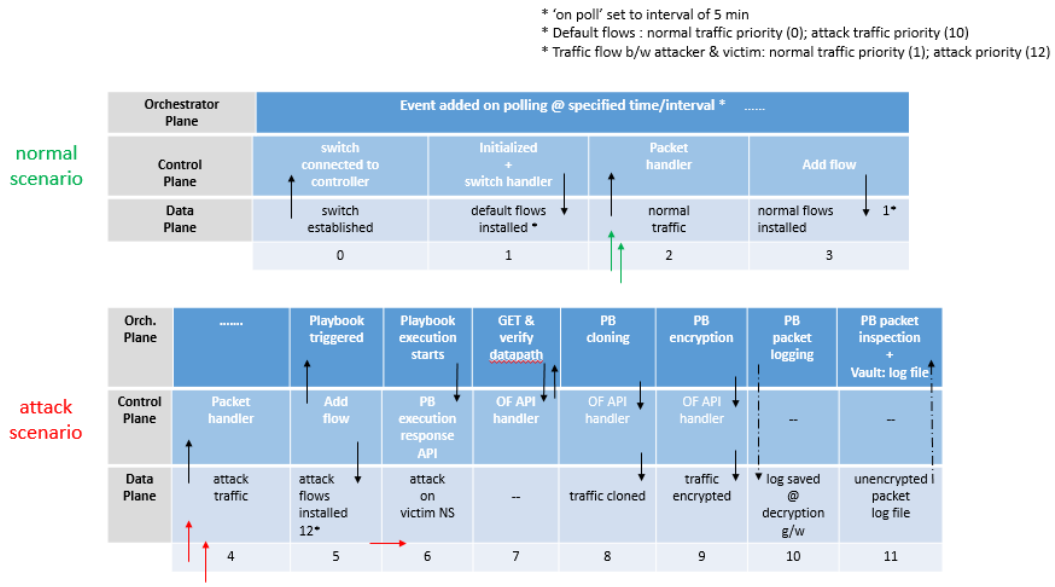

Figure 4.5.: Use case 1: Flow of action between Data, Control and Orchestrator plane

The Phantom app 'polling' is incorporated to add containers for playbook execution and the characteristic feature of the app is 'on poll'. The App has a provision to 'set interval' and in our case it is set to 5 minutes, meaning a poll is conducted once every 5 minutes to add a new container to the events queue. This feature uses the REST APIs to add new containers and is most beneficial for running a series of playbooks at specified intervals on different machines. In the fig 4.5, steps 0-3 depict action flow in a normal scenario. The steps 4-12 depict the deducing of malicious activity and the corresponding playbook being triggered. In fig 4.6, the playbook layout as drawn in the Phantom Visual Playbook Editor is presented.

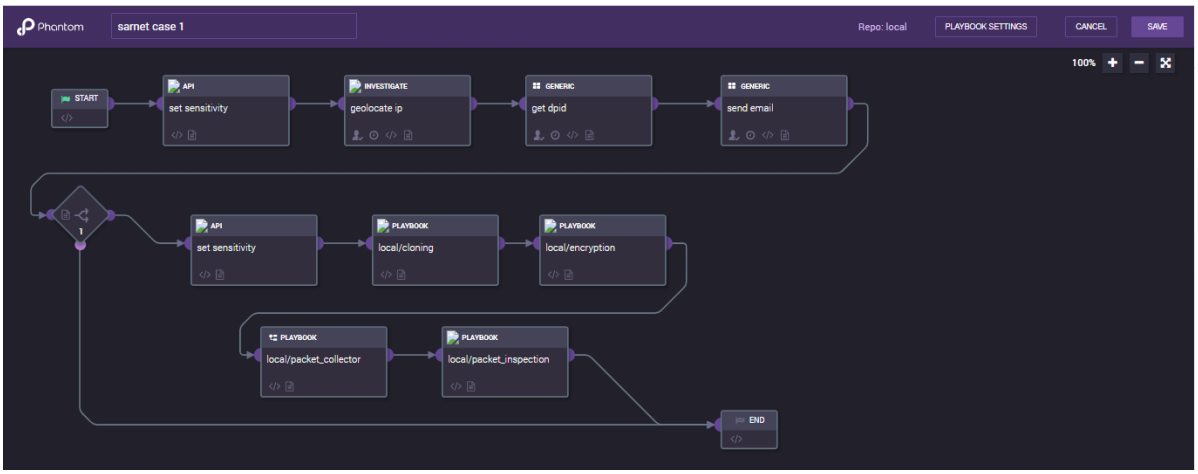

Figure 4.6.: Use case 1: Phantom playbook - sarnet

4.3. Use Case 2

In Chapter 3, the Use case 2 was defined which aims at performing a VM Live migration along with traffic cloning and traffic encryption. The switch is attached to the attacker namespace, through which the possible data ex-filtration is triggered by the attacker. The idea of a live migration of VM is to transfer a copy of the VM under attack to the secure node where further analysis on the VM and steps such as VM quarantine can be performed. For this use case, the steps followed from establishing a connection between the switch and controller, installing of flows on the switch, traffic cloning and traffic encryption are similar to the steps followed for the Use case 1. The subsequent steps are to automate these functions via a new playbook as explained in the following section.

4.3.1. Playbook for automated mitigation

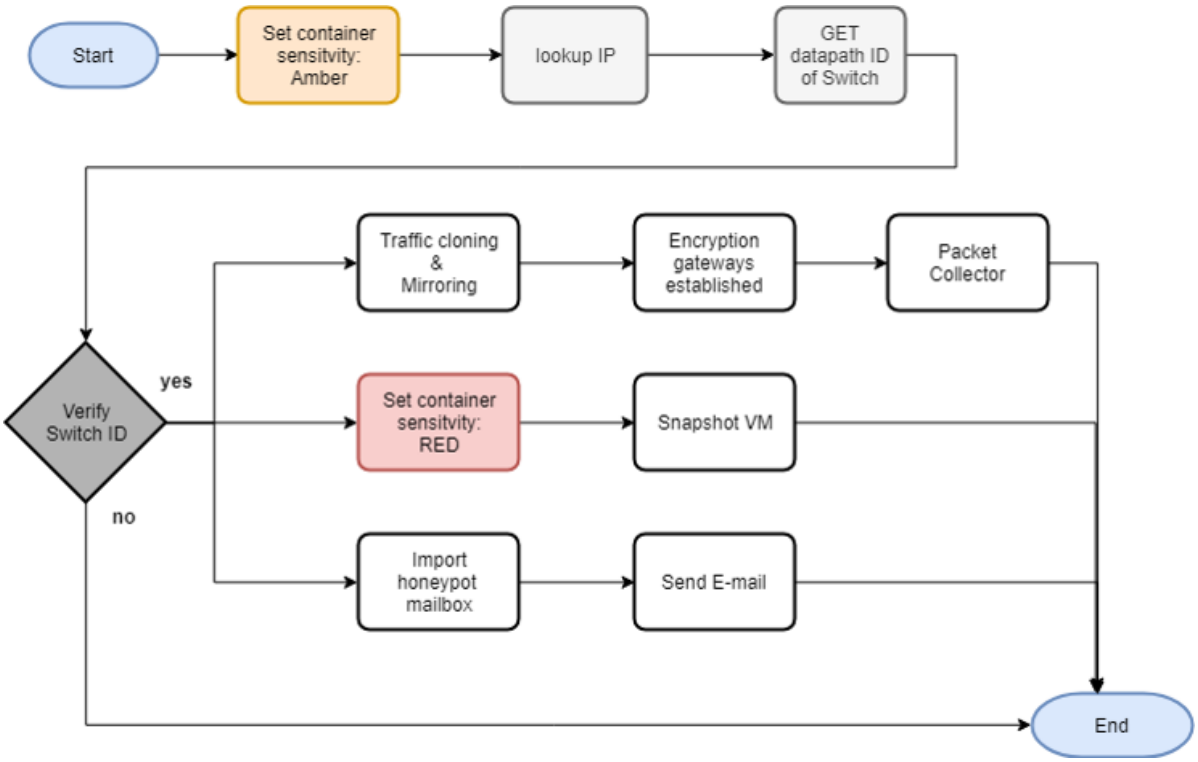

Figure 4.7.: Use case 2: Playbook flowchart

The scheme for the playbook is drawn in fig. 4.7 and further listed in detail below.

- A HTTP POST from the control plane- Ryu triggers the playbook run with 'start'.
- Sensitivity- The Phantom API block action 'set sensitivity' labels the container from 'white' to colour 'amber' that notifies higher sensitivity and the possibility of an attack to the security admin monitoring the events queue.

- lookup IP- *Asset URL: http://ipinfo.io*
  The IP information web service URL fetches hostname of the system where the possible attack is originating. This function is similar to the 'geo-locate IP' from Use Case 1 and has been configured for the purpose of this demonstration.

- GET dpid- *Asset URL: VM instance of SDN controller*
  The Ryu app is configured to fetch and store datapath IDs of all the switches connected to the controller. The Ryu ofctl service responds to northbound REST API with the URI- /stats/switches sent from the orchestrator.

- Verify switch ID- Before the execution of mitigation action, the Phantom decision block matches the datapath IDs of the three connected switches present in our network to the ones fetched by the 'GET dpid' function. If they do not match the playbook is terminated and upon verification the next mitigation steps are activated.

- Sensitivity- After verification, maliciousness in the system is confirmed and the sensitivity of the container is changed using the 'set sensitivity' API from 'amber to color 'red' for monitoring purposes and denoting the highest sensitivity in the event queue.

- Traffic cloning and encryption- Duplicating and mirroring traffic onto an adjacent port of the switch is automated using the Ryu ofctl service. Encryption of malicious traffic is achieved by enabling traffic flows via GRE-IPSec enabled ports as done in Case 1.

- Packet collector- At the secure node located on the decryption gateway, affected traffic packets are collected using Wireshark and saved as a PCAP file.

- Snapshot VM- The OpenStack controller module, is the cloud's Virtual Infrastructure Manager (VIM) and Keystone service is used for its authentication. With the OpenStack service Nova, a snapshot of the infected VM containing the switch and attached namepsaces is made on the OpenStack controller. Using OpenStack Glance, the VM copy is downloaded in QCOW2 format at the secure node. An automated live migration of the VM image from the secure node to the orchestrator is performed and this image is available in the vault of container whhich runs this playbook.

- Import mailbox- At the affected Open vSwitch node a simple research-based python honeypot is executed which presents the attacker with the login screen and prompts for user credentials at the onset of malicious activity. Attack related details such as 'From' and 'To' email address of the attacker, time stamps of login are recorded into a mailbox which is then uploaded to the container vault. This function is executed only for the demonstration of this security function and may fail during a complex attack that tries to surpass the login screen prompt.

- Send e-mail- *Asset: SMTP server, user credentials, SSL port*
  The Phantom SMTP app is configured to Gmail server and at this step triggers an email notification informing of a possible attack and the hostname of VM generating the attack to the security admin. This terminates the end of playbook execution.

The container required for executing playbook is configured differently than in Use case 1, which used 'polling' function to add containers. Here, a new event (or container) is added via REST API from the control plane when an attack is confirmed. Playbook is marked 'active' and on addition of a new container active playbooks get automatically triggered on the orchestrator. Hence, a playbook is not directly triggered from the Ryu controller as in Case 1. At the end of playbook execution a PCAP file generated by the packet logger function, an imported mailbox with attacker related information and a live migrated snapshot of the infected VM is available on the vault of the playbook container. The flow of action between data plane, control plane and orchestrator plane is depicted in fig 4.8., where steps 0-3 present action flow in a normal scenario. The steps 4-12 depict the mitigation actions in response to the malicious activity. In fig 4.6, the playbook layout as drawn in the Phantom Visual Playbook Editor is presented.

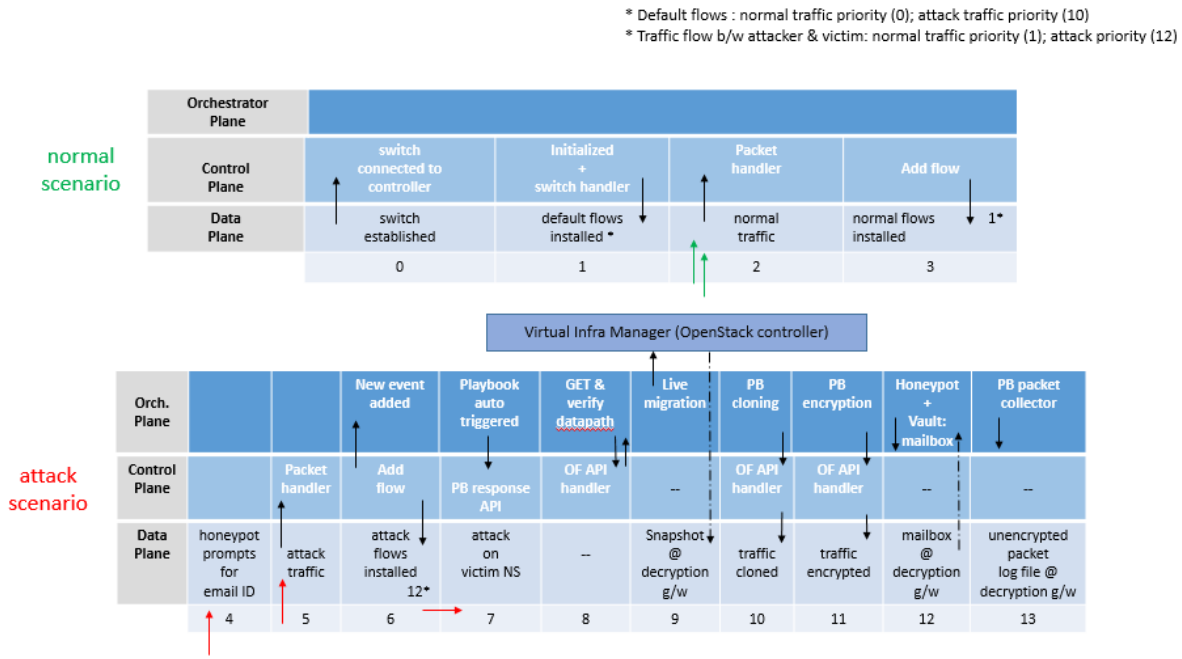

Figure 4.8.: Use case 2: Flow of action between Data, Control and Orchestrator plane

figures/pb2_phantom.PNG

Figure 4.9.: Use case 2: Phantom playbook - sarnet (**To be Added**)

4.4. REST APIs

Application Programming Interface (API), is an interface or protocol for communication between different applications or parts of a computer program. Upon combining API with the HTTP protocol methods it can be used to access parts of the web. REST stands for Representational State Transfer and is a standardized architecture style for creating a Web Service API. SDN decouples control signals from data signals aiming at a flexible and programmable network that is different from legacy network architectures where data plane, control plane and management plane are integrated into network devices. Due to this decoupling, the programmability of the SDN network is concentrated via a lower level- south-bound API to the data plane and higher level- north-bound API with the application layer making it a complex distributed system whose structure, function, resource, and behavior can change dynamically. As discussed in **[[28]]** and presented in fig 4.10, the controller maintains a global topology of the network and provides a configuration API to manage the flow tables in the underlying OpenFlow devices in the data plane.

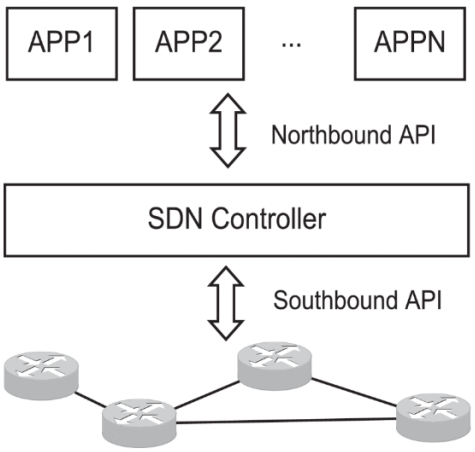

Figure 4.10.: REST APIs in OpenFlow-based SDN network

The SDN controller provides a programmable interface for applications to observe and change the network states dynamically called the Northbound API. REST API can be highly extensible and maintainable for managing distributed data networking resources by providing different functions at the same time while also making certain changes to those functions over time without breaking its clients. In case of SDN, it includes services from both data and control planes, such as switches, routers, subnets, networks, NAT devices, and controllers. REST API is characterized by design rules such as client-server communication with a clear separation and uniform interface for communication between them; stateless where the server does not store any information of a request to re-use it and that the client requests must carry all the information required by the server to carry out a request. Parameters involved in the API design is listed below [49].

- The URL pattern *resource/identifier/resource* is used for communication between resources.
- HTTP methods (HTTP/1.1) GET, POST, PUT, DELETE are used to operate on resources.
- HTTP response status codes represent the outcome of operations on resources.
- Payload format encoding via HTTP headers (e.g. Content-Type: and Accept:); GET parameters (e.g. format=json) and resource label (e.g. /foo.json).

The OpenStack IP instance of the SDN component or orchestrator is the resource in our case. The HTTP methods and response status codes of south-bound API, North-bound APIs and Phantom RESt APIs for communication within our security function is recorded in the following section. POSTMAN was used to design Phantom APIs interactively. The HTTP uses Authorization header for authentication and the Accept header to initiate the right behavior by API and was set on POSTMAN.

4.4.1. APIs of security function

South-bound APIs

- APIs to relay information to the switches and routers at the lower level to data plane.
- SDN Controllers communicate with the switches/routers and commonly use OpenFlow, OVSDB as is in our case.

North-bound APIs

- APIs used to communicate between the SDN Controller and the services and applications running from the Application layer.
- A WSGI web server is used to create the REST APIs in Ryu. Using the ofctl service, flow programming and Statistics retrieval is performed for our case.

```
GET switch statistics

ubuntu@ryu:~$ curl -X GET http://10.20.5.39:8080/stats/switches
[1, 2, 3]


GET switch flow stats

ubuntu@ryu:~$ curl -X GET http://10.20.5.39:8080/stats/flow/1
{"1": [{"actions": ["OUTPUT:5", "OUTPUT:2"], "idle_timeout": 0, "cookie": 0,
"packet_count": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 148,
"duration_nsec": 763000000, "priority": 10, "length": 112, "flags": 0,
"table_id": 0, "match": {"dl_type": 2048, "nw_proto": 6, "tp_dst": 100}},
{"actions": ["OUTPUT:CONTROLLER"], "idle_timeout": 0, "cookie": 0,
"packet_count": 149, "hard_timeout": 0, "byte_count": 6258, "duration_sec": 148,
"duration_nsec": 763000000, "priority": 0, "length": 80, "flags": 0, "table_id":
0, "match": {}}]}
```

Figure 4.11.: Northbound API using GET method

- Switch statistics such as datapath ID, flow stats are retrieved using *GET*. New flows are added to the data plane from the orchestrator via RYU controller, using python-based *POST* method. These methods are used to achieve traffic cloning and traffic encryption.

```
POST new flow to switch

import subprocess
import requests
import array

url = "http://10.20.5.3:8080/stats/flowentry/add"

#cloning flow
payload = '{\"dpid\": 0000000000000001, \"priority\": 30, \"flags\": 1, \"match\":
{\"eth_type\": 2048, "nw_proto\": 6, "tp_dst\": 100}, \"type\": \"OUTPUT\",
"port\": 5} ] }'
headers = {
    'cache-control': "no-cache",
    'Postman-Token': "98befe48-d360-45c7-a6a3-2326bf21f798"
    }
response = requests.request("POST", url, data=payload, headers=None)
```

Figure 4.12.: Northbound API using POST method

Phantom REST APIs

- On detection of malicious activity, playbooks are triggered from the SDN controller plane using POST method.

```
Trigger a playbook with playbook ID, container ID

curl -X POST \
  https://admin:*****@10.20.5.34/rest/playbook_run \
  -H 'Content-Type: application/json' \
  -H 'G-TOKEN: 95e3bcff-bfca-454d-b59e-768da6280c38' \
  -H 'Postman-Token: 0dfea112-ed58-4bcd-b030-d14f0d5202c9' \
  -H 'cache-control: no-cache' \
  -d '{
  "container_id": 287,
  "playbook_id": "local/sarnet case 1",
  "scope": "new",
  "run": true
}
```

Figure 4.13.: Phantom API using POST to trigger playbook

- A new event is added to the Phantom event queue using POST method to automatically trigger 'active' playbooks, upon detection of malicious activity.
- USing GET method, details of playbook run event is fetched from phantom using REST.

```
Add a new container into Phantom event queue

curl -X POST \
  https://admin:*****@10.20.5.34/rest/container \
  -H 'Content-Type: application/json' \
  -H 'Postman-Token: 3d6990da-67b9-4d7c-9ef4-ba339de534ed' \
  -H 'cache-control: no-cache' \
  -d '{
  "id": 355,
  "version": "1",
  "label": "events",
  "name": "my_event",
  "description": "test event for sarnet security function",
  "playbook_run_id": 413,
  "status": "open",
  "sensitivity": "amber",
  "severity": "medium",
  "kill_chain": "",
  "data": {},
  "artifact_count": 1
}'
```

Figure 4.14.: Phantom API using POST to add new event

```
GET details of a playbook_run from Phantom

curl -X GET \
  https://admin:*****@10.20.5.34/rest/playbook_run \
  -H 'Postman-Token: ae80816a-d273-4c96-ac67-3d45adbfca40' \
  -H 'cache-control: no-cache' \
  -d '{
  "id": 355,
  "version": "1",
  "label": "events",
  "name": "my_event",
  "source_data_identifier": "64c2a9a4-d6ef-4da8-ad6f-982d785f14b2",
  "description": "test event for sarnet security function",
  "status": "open",
  "sensitivity": "amber",
  "severity": "medium",
  "create_time": "2019-08-16 07:18:46.631897+00",
  "start_time": "2019-08-16 07:18:46.636966+00",
  "end_time": "",
  "due_time": "2019-08-16 19:18:00+00",
  "close_time": "",
  "kill_chain": "",
  "owner": "admin",
  "hash": "52d277ed6eba51d86190cd72405df749",
  "tags": [""],
  "asset_name": "",
  "artifact_update_time": "2019-08-16 07:18:46.631875+00",
  "container_update_time": "2019-08-16 07:19:12.359376+00",
  "ingest_app_id": "",
  "data": {},
  "artifact_count": 8
}'
```

Figure 4.15.: Phantom API using GET to fetch playbook run details

A. General Addenda

A.1. Detailed Addition

to be added

List of Figures

List of Tables

Bibliography

[1] V. Farhat, B. McCarthy, and R. Raysman. "Cyber Attacks: Prevention and Proactive Responses". In: (2011).

[2] J. B. Marshall and M. A. Saulawa. "International Journal of International Law: ISSN: 2394-2622 (Volume 1 Issue 2)". In: ().

[3] U. Nations. "Conference on Trade and Development". In: (). DOI: https://unctad.org/en/Pages/DTL/STI_and_ICTs/ICT4D-Legislation/eCom-Cybercrime-Laws.aspx.

[4] A. Singh and M. Singh. "An Empirical Study on Automotive Cyber Attacks". In: (2018).

[5] Cisco. "Types of cyber-attacks". In: (). DOI: https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html.

[6] C. for Strategic International Studies. "Significant cyber incidents". In: (). DOI: https://www.csis.org/programs/cybersecurity-and-governance/technology-policy-program/other-projects-cybersecurity.

[7] T. Vaidya. "Survey and Analysis of Major Cyberattacks". In: (). DOI: https://security.cs.georgetown.edu/~tavish/cyberattacks_report.pdf.

[8] F. Policy. "10 years After the Landmark Attack on Estonia, Is the World better prepared for Cyber Threats?" In: (). DOI: https://foreignpolicy.com/2017/04/27/10-years-after-the-landmark-attack-on-estonia-is-the-world-better-prepared-for-cyber-threats/.

[9] C. for Strategic International Studies. "Significant Cyber Incidents since 2006". In: (). DOI: https://csis-prod.s3.amazonaws.com/s3fs-public/190211_Significant_Cyber_Events_List.pdf.

[10] BBC. "Timeline: How Stuxnet attacked a nuclear plant". In: (). DOI: https://www.bbc.com/timelines/zc6fbk7.

[11] T. Sun. "CYBER WARFARE? Google 'hit by WORST EVER cyberattack' with traffic 'hijacked' and routed through Russia and China in 'war games"". In: (). DOI: https://www.thesun.co.uk/news/7726913/google-cyber-attack-traffic-highjacked-russia-china/.

[12] W. Smart. "Lessons learned review of the WannaCry Ranswomware Cyber Attack". In: (2018).

[13] B. News: "Ransomware cyber-attack: Who has been hardest hit?" In: (). DOI: https://www.bbc.com/news/world-39919249.

[14] I. T. Union: "Definition of cybersecurity". In: (). DOI: https://www.itu.int/en/ITU-T/studygroups/com17/Pages/cybersecurity.aspx.

[15] NIST. "Cyber Security Standards". In: (). DOI: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=152153.

[16] E. Commission. "State of the Union 2017 – Cybersecurity: Commission scales up EU's response to cyber-attacks". In: (). DOI: http://europa.eu/rapid/press-release_IP-17-3193_en.htm.

[17] D. S. Purser. "Best practices in Computer Network Defense: Incident Detection and Response". In: (2014).

[18] E. Commission: "European Cloud Strategy 2012". In: (). DOI: https://ec.europa.eu/digital-single-market/en/european-cloud-computing-strategy.

[19] K. Donert. "Creating A Cloud Computing in Education Strategy for Europe". In: (). DOI: https://www.slideshare.net/eurogeo/creating-a-cloud-computing-in-education-strategy-for-europe.

[20] TNO. "Cyber Security Robustness". In: (). DOI: https://www.tno.nl/en/focus-areas/information-communication-technology/expertise-groups/cyber-security-robustness/.

[21] SARNET. "Security Autonomous Response with programmable NETworks". In: (). DOI: https://delaat.net/sarnet/index.html.

[22] B. G. e. a. Leon Gommans John Vollbrecht. "The Service Provider Group Framework: A framework for arranging trust and power to facilitate authorization of network services". In: (2014).

[23] L. Gommans. "Multi-Domain Authorization for e-Infrastructures". In: (2014).

[24] M. Revay and M. Liska. "OODA loop in command control systems". In: Oct. 2017, pp. 1–4. DOI: 10.23919/KIT.2017.8109463.

[25] R. Koning, B. Graaff, C. Laat, R. Meijer, and P. Grosso. "Interactive analysis of SDN-driven defence against distributed denial of service attacks". In: June 2016, pp. 483–488. DOI: 10.1109/NETSOFT.2016.7502489.

[26] R. Koning, B. Graaff, G. Polevoy, R. Meijer, C. Laat, and P. Grosso. "Measuring the efficiency of SDN mitigations against attacks on computer infrastructures". In: *Future Generation Computer Systems* 91 (Aug. 2018). DOI: 10.1016/j.future.2018.08.011.

[27] E. Kaljic, A. Maric, P. Begovic, and M. Hadzialic. "A Survey on Data Plane Flexibility and Programmability in Software-Defined Networking". In: *IEEE Access* 7 (Apr. 2019), pp. 47804–47840. DOI: 10.1109/ACCESS.2019.2910140.

[28] W. Braun and M. Menth. "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices". In: *Future Internet* 6 (May 2014), pp. 302–336. DOI: 10.3390/fi6020302.

[29] L. Li, W. Chou, W. Zhou, and M. Luo. "Design Patterns and Extensibility of REST API for Networking Applications". In: *IEEE Transactions on Network and Service Management* 13 (Apr. 2016), pp. 154–167. DOI: 10.1109/TNSM.2016.2516946.

[30] I. Gde, M. Muthohar, A. Prayuda, and C. Deokjai. "Time-based DDoS Detection and Mitigation for SDN Controller". In: Aug. 2015. DOI: 10.1109/APNOMS.2015.7275389.

[31] D. Ageyev, O. Bondarenko, W. Alfroukh, and T. Radivilova. "Provision security in SDN/NFV". In: Feb. 2018, pp. 506–509. DOI: 10.1109/TCSET.2018.8336252.

[32] M. De Benedictis and A. Lioy. "On the establishment of trust in the cloud-based ETSI NFV framework". In: Nov. 2017. DOI: 10.1109/NFV-SDN.2017.8169864.

[33] E. ISG. "NFV in ETSI". In: (). DOI: https://www.etsi.org/technologies/nfv.

[34] B. Jaeger. "Security Orchestrator: Introducing a Security Orchestrator in the Context of the ETSI NFV Reference Architecture". In: Aug. 2015, pp. 1255–1260. DOI: 10.1109/Trustcom.2015.514.

[35] G. Research. "Gartner Market Guide for Security Orchestration, Automation, and Response Solutions". In: (). DOI: https://www.rapid7.com/info/gartner-market-guide-soar/.

[36] K. Ogudo. A. "Analyzing Generic Routing Encapsulation (GRE) and IP Security (IPSec) Tunneling Protocols for Secured Communication over Public Networks". In: Aug. 2019, pp. 1–9. DOI: 10.1109/ICABCD.2019.8851004.

[37] B. Stackpole. "An introduction to IPSec". In: Jan. 2007, pp. 2093–2101.

[38] S. Jahan and S. Saha. "Application Specific Tunneling Protocol Selection for Virtual Private Networks". In: Jan. 2017. DOI: 10.1109/NSysS.2017.7885799.

[39] J. Mulerikkal and Y. Sastri. "A Comparative Study of OpenStack and CloudStack". In: Sept. 2015, pp. 81–84. DOI: 10.1109/ICACC.2015.110.

[40] S. Sahasrabudhe and S. Sonawani. "Comparing openstack and VMware". In: Oct. 2014, pp. 1–4. DOI: 10.1109/ICAECC.2014.7002392.

[41] S. Makhsous, A. Gulenko, O. Kao, and F. Liu. "High available deployment of cloud-based virtualized network functions". In: July 2016, pp. 468–475. DOI: 10.1109/HPCSim.2016.7568372.

[42] OpenStack. "OpenStack service overview". In: (). DOI: https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html.

[43] P. Kacsuk and N. Podhorszki. "Dataflow parallel database systems and LOGFLOW". In: Feb. 1998, pp. 382–388. ISBN: 0-8186-8332-5. DOI: 10.1109/EMPDP.1998.647223.

[44] O. N. Foundation. "OpenFlow Switch Specification". In: (). DOI: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf.

[45] T. K. e. a. B Pfaff J Pettit. "The Design and Implementation of Open vSwitch". In: (2015).

[46] Ryu. "Components of Ryu". In: (). DOI: https://ryu.readthedocs.io/en/latest/components.html.

[47] Ryu. "Ryu application API". In: (). DOI: https://ryu.readthedocs.io/en/latest/ryu_app_api.html.

[48] Phantom. "Overview: Introduction". In: (). DOI: https://my.phantom.us/3.5/docs/automation/overview.

[49] AusDTO. "API Design Guide Documentation, Release 0.1". In: (). DOI: https://readthedocs.org/projects/apiguide/downloads/pdf/latest/.