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

396093

Advisor : Prof. Dr. Ing-. Thomas Bauschert
Supervisor : Mr. Trung Phan Van
Submission Date : 10th June 2020

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, 10th June 2020

_____

Vibha R Goutham

Acknowledgments

This thesis is submitted for fulfillment of the requirements for the academic degree, Master of Science in 'Information and Communication Systems' at Technische Universität Chemnitz.

Foremost, I would like to express my sincere gratitude to Prof. Dr.-Ing. Thomas Bauschert at the Chair of Communication Networks, Technische Universität Chemnitz for his invaluable support, for his patience throughout the course of the project and for providing me this opportunity to work at their department.

I am incredibly grateful to my thesis supervisor M.Sc. Trung Phan Van for his excellent guidance and motivation throughout the project. His supervision helped me steer this research in the right direction.

I would also like to thank Dr. Piotr Zuraniewski at TNO, Netherlands for his insightful comments and for helping me conceptualise this project in its early days. I extend my thanks to Ir. Frank Fransen at TNO, Netherlands for providing an opportunity to join their team as an intern and access the research infrastructure at their company.

Finally, I express my profound gratitude to my parents and all my friends for providing me with unfailing support and continuous encouragement through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

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

List of Figures

List of Tables

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 Shanghai 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 legislation and more than 30 countries have no legislation in place. On comparing E-Commerce legislation 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 commonly encountered

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.: Vulnerabilities commonly found

Type	Description
Network Technology weakness	Arising out of un-secure 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 un-secure 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 ex-filtration 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,

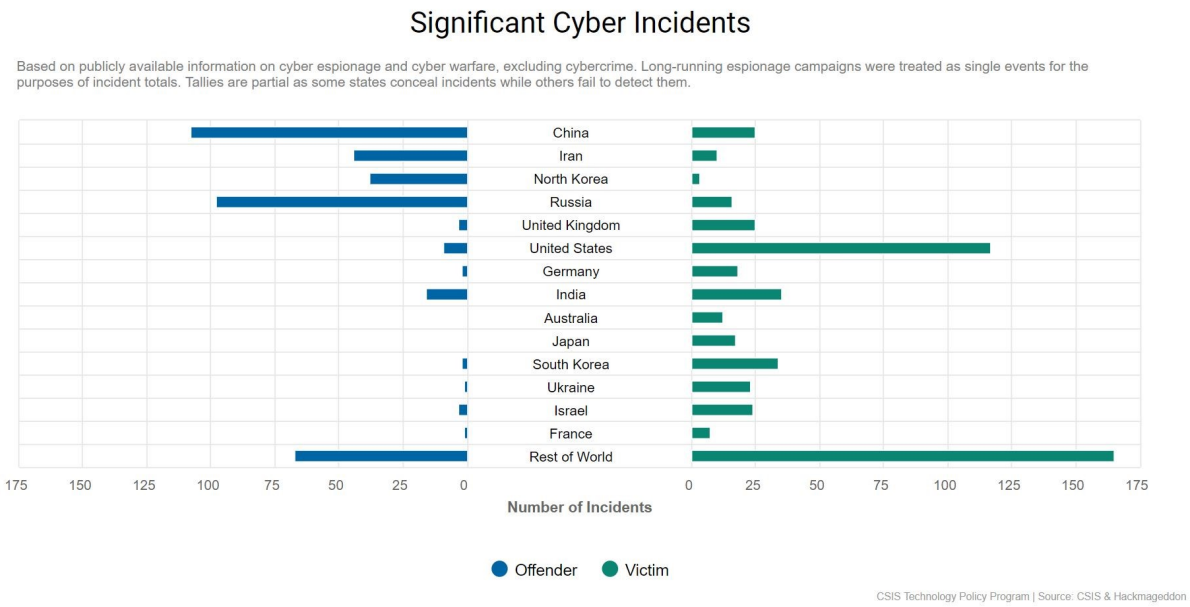

Figure 2.1.: Countries involved in cyber incidents as offender vs. as victim.

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.

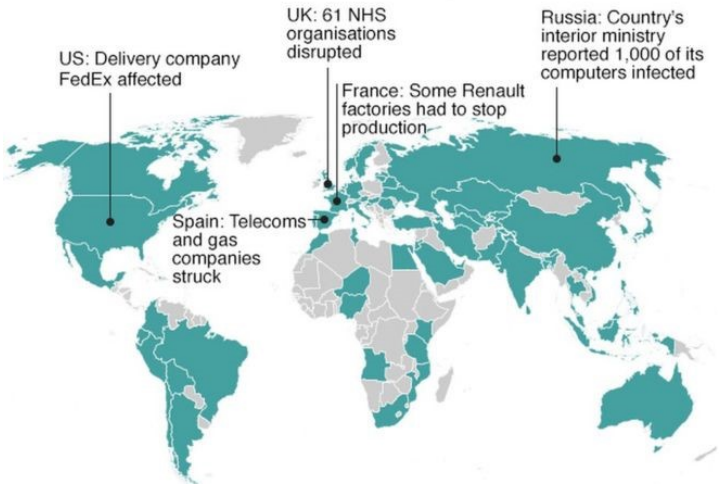

Figure 2.3.: Companies affected worldwide from WannaCry

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 [14]. 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 [15] [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 other things. While the cloud is comparable to the term Internet, cloud computing offers services such as storage, databases, networking, software over the Internet. The usage of cloud paves way for flexible usage of resources and increased innovation. 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 [18], which focused on increased innovation, reduced costs while emphasizing on legal framework around cloud computing.

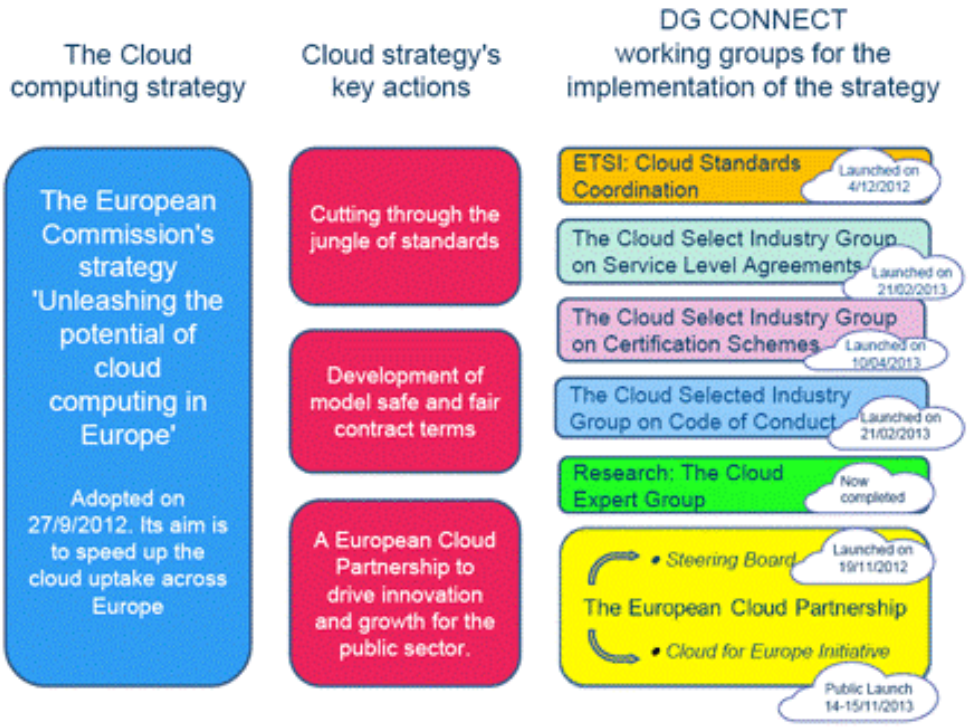

Figure 2.4.: The EU Cloud Strategy

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 [19]. 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 Dutch Ministry of Defence and its other customers are in the field of banking, telecommunications and logistics. 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 [20]. 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 other sub-project of the SARNET group focuses 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. The project structure of SARNET is organized at three levels. The Tactical level- determining best defence scenario against cyber-attacks by deploying functions [21] and analysing security state and KPI information [22]; 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 [20].

This project was conceptualized in its initial days at TNO in the context of NWO SARNET.

3. Designing the Security Function

3.1. Fundamentals

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 [23]. 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 [23], comprising four processes as depicted in fig 3.1 and as described below.

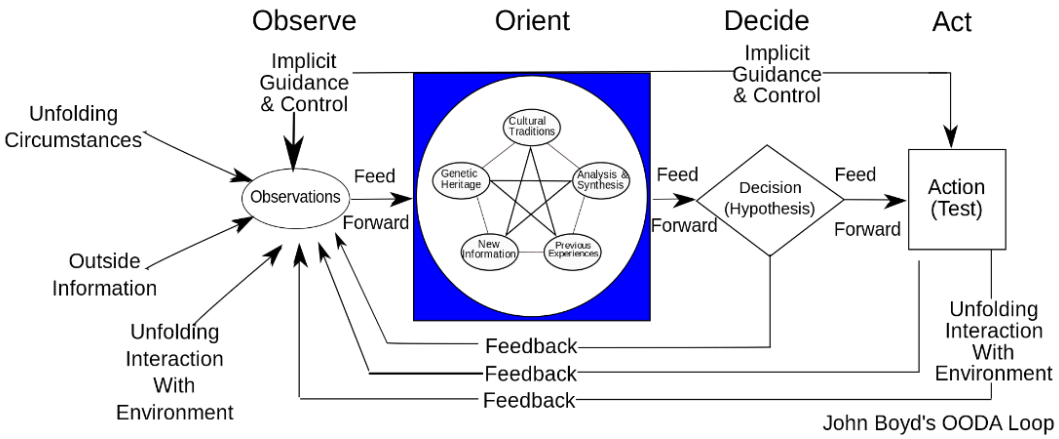

Figure 3.1.: OODA Loop

- 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

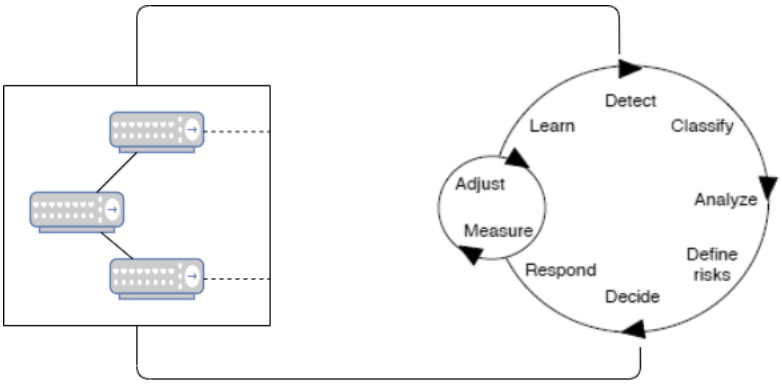

Figure 3.2.: Sarnet Loop

The Secure Autonomous Response Network (SARNET) framework provides autonomous response across multiple domains to network attacks by exploiting underlying SDNs functionalities and virtualized network functions [24]. 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 [25]. 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 observable in the state of the network and services. Detection of change in values of observable 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 re-programs the network flows, redefines location of the virtualized network functions, and possibly move the location of computing and storage services [24]. 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 [26]. 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.

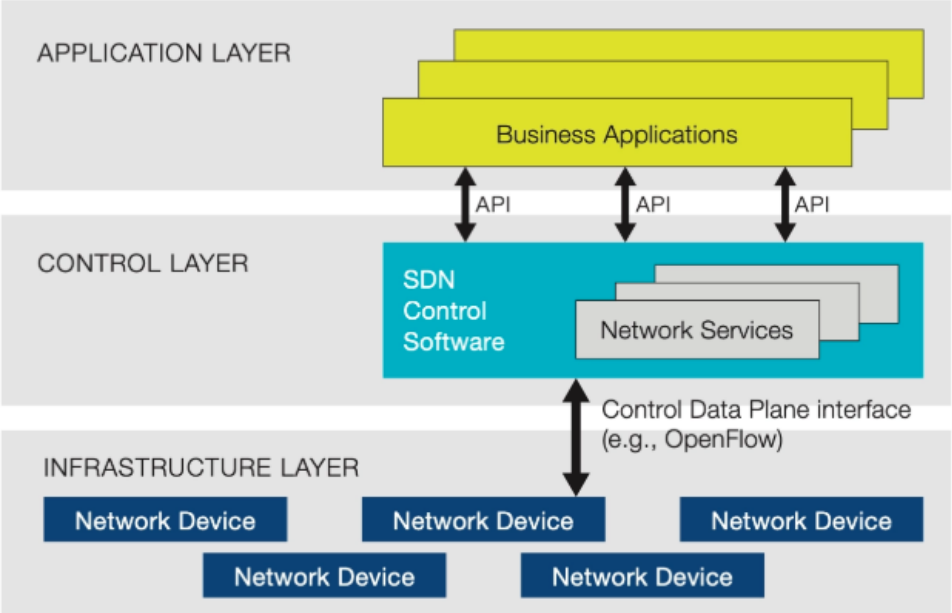

Figure 3.3.: SDN Architecture

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 [26]. 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 [27]. 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 [28].

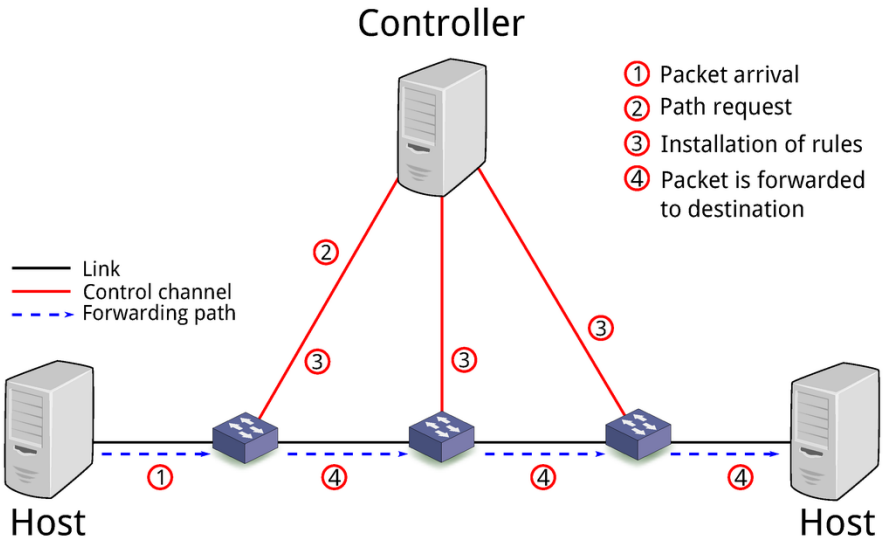

Figure 3.4.: SDN reactive flow management

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 [29].

3.1.3. Network Function Virtualization(NFV)

Traditional networks 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; scalability and compatibility challenges when incorporating multi-operator devices. These limitations led to evolution of Network Function Virtualization (NFV) where functions of network elements are logically virtualized separating it from hardware infrastructure as depicted in fig 3.5 [30]. SDN facilitates dynamically controlling the network and provisioning of networks as a service whereas NFV offers to manage and orchestrate virtualization of resources for provisioning of network functions and building higher-layer network services. Hence, SDN and NFV are complementary and also co-dependent.

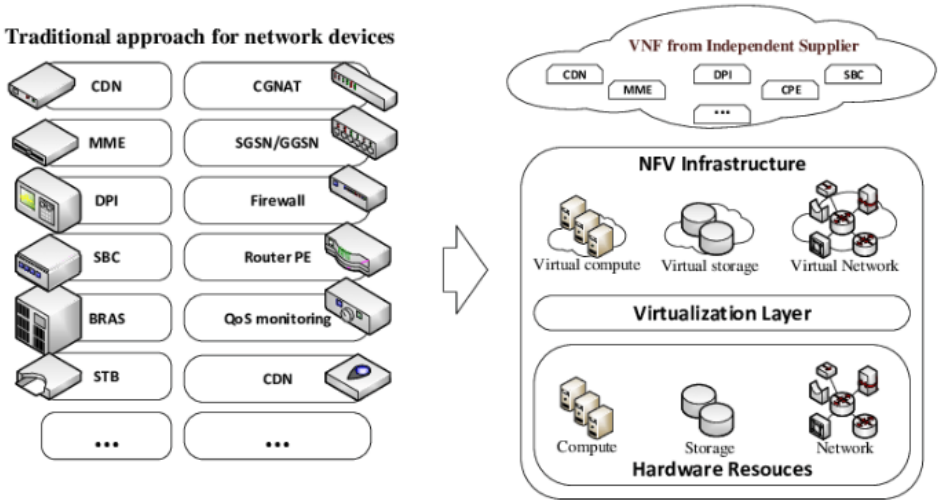

Figure 3.5.: Traditional approach vs. NFV approach

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 overall costs for provisioning and maintenance of network devices, allowing flexible placement and optimisation of VNFs in the infrastructure based on 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. In [31] NFV release 2 and open issues are presented. In its release 3 from 2017-18, support for contemporary technologies such as edge computing and network slicing, advances in acceleration technologies virtualization among were mainly discussed [32]. 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 [32].

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 [33]. 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 [34], 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.

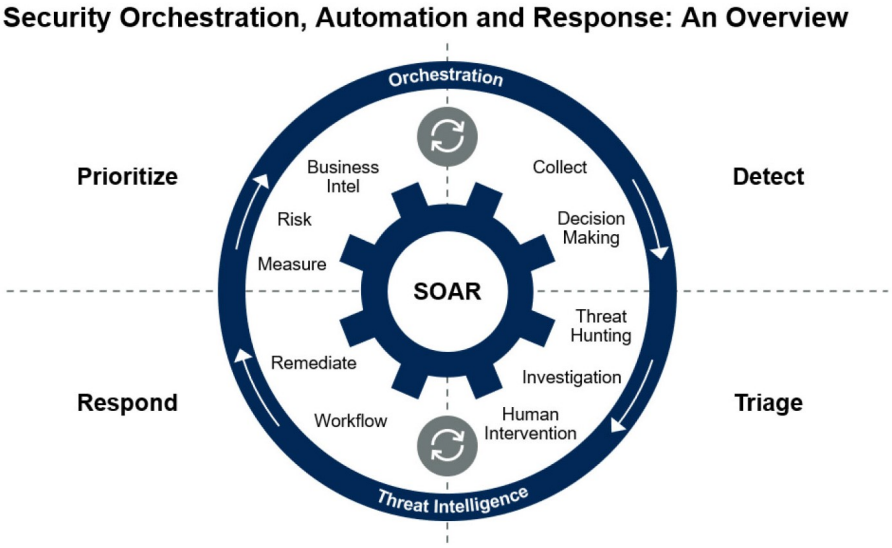

Figure 3.6.: Security Orchestration, Automation, and Response (SOAR)

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 [33], which is responsible for managing security services provided via a Security Service catalog that triggers the life-cycle 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 [35]. The architecture comprises of two IP header constructs, the authentication header (AH) and the encapsulation security payload (ESP), along with the Security association (SA) [36].

- 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 [37] 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

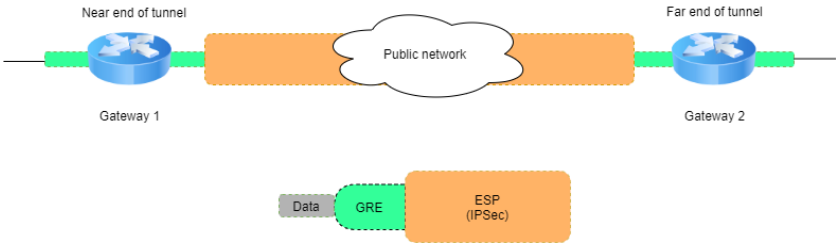

Figure 3.7.: GRE-IPSec tunneling

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 [37], 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.

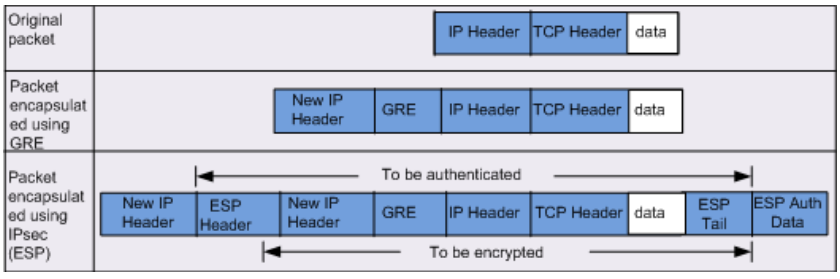

Figure 3.8.: GRE-IPSec header encapsulation on data packet

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 [38] 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 [39] 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 [40]. 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 [41] 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*.

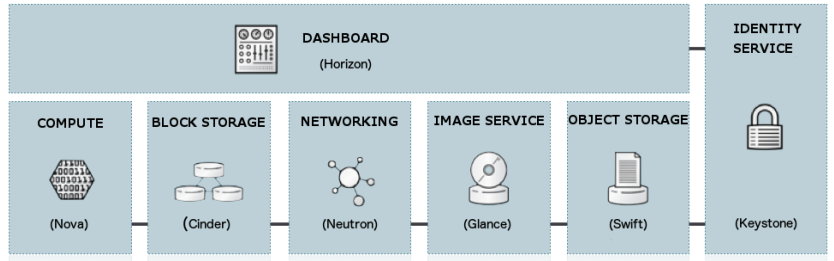

Figure 3.9.: OpenStack service overview

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 [42] 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.

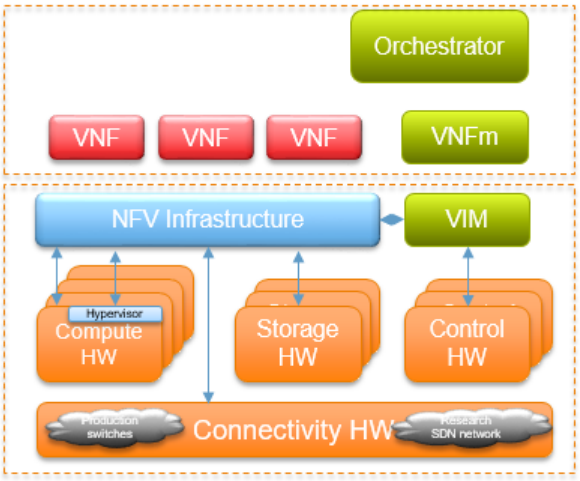

Figure 3.10.: Layout of the in-house cloud

3.2.3. 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 v-switch (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 [43] 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 [42] to perform packet look-ups 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.

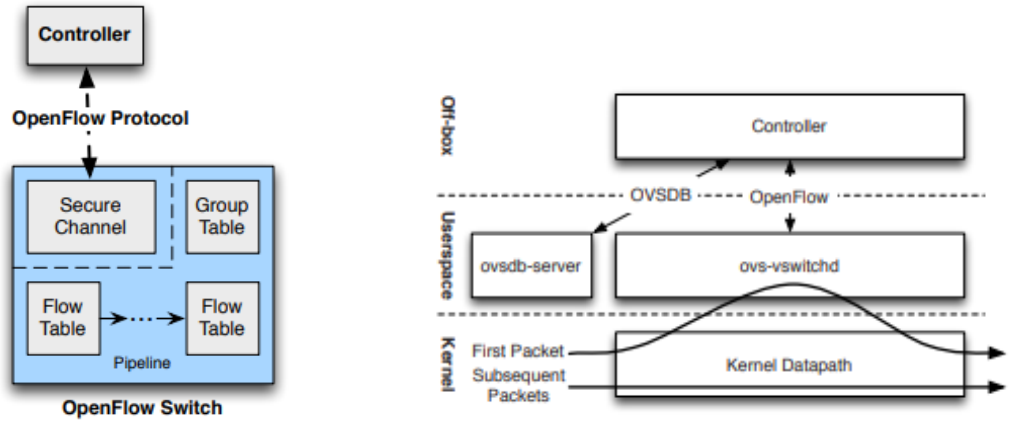

Figure 3.11.: (a) and (b): Open vSwitch components and interface

As presented in [44] 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 utilities 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 [43].

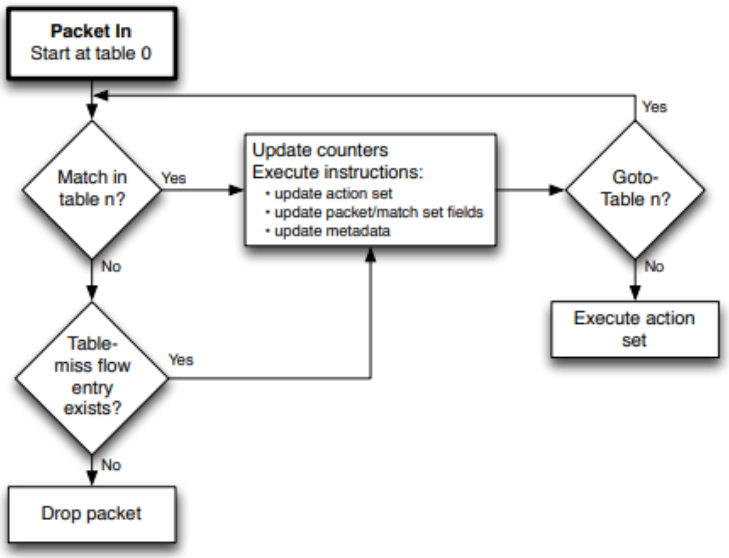

Figure 3.12.: Packet flow through an OpenFlow switch

3.2.4. 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.

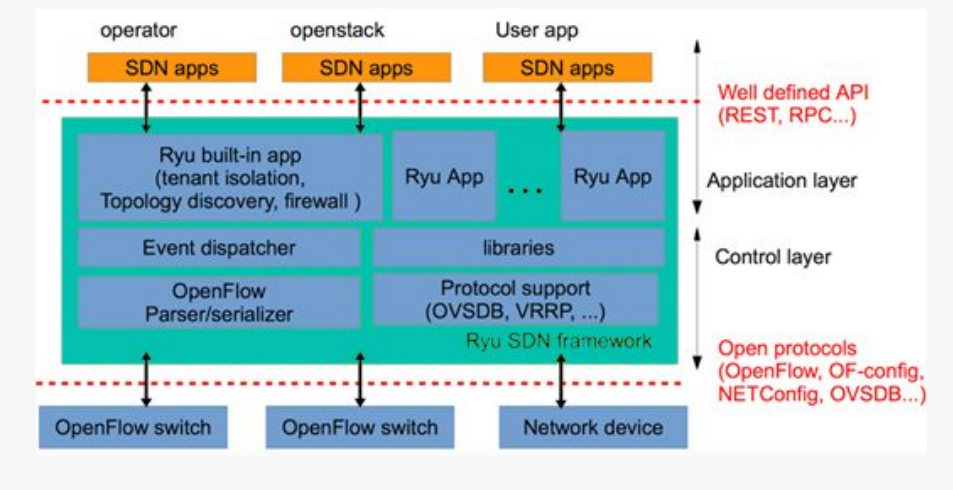

Figure 3.13.: Components of Ryu SDN controller

The components of Ryu are as depicted in fig. 3.13. The main executable is the bin/ryu-manager [45]. 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 [46]. 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.

3.2.5. 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 incoming feed of security data while including disparate tools to control incidents and to offer quicker response times. 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 and 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, endpoints and assets [47]. Security response analysts can view, act on incidents and various attributes, manage related information through the life-cycle of an incident, develop next generation 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 administrator builds playbooks graphically. The workflow in Phantom is shown in fig. 3.14.

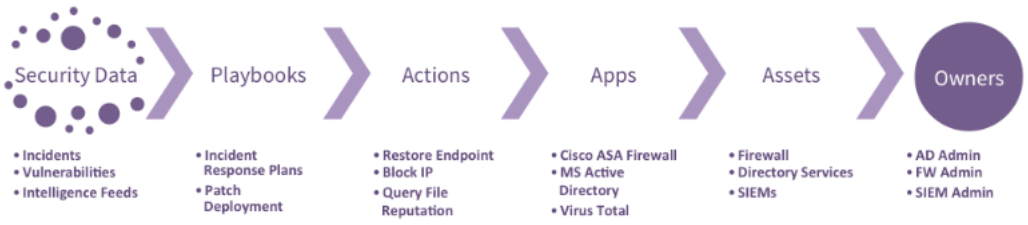

Figure 3.14.: Workflow in security orchestrator: Phantom [46]

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 fig 3.14, workflow is listed in table 3.1. 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, sample generator or through a REST API. This input data triggers our mitigation plan, designed in the form of python based playbooks which automates running several actions. Containers are the top level data structure that playbooks operate on in Phantom and they are used to group objects together that are related to a particular event that

Table 3.1.: Security orchestrator - 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.

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 and also a new app *Ryu* was built for automated network communication between Phantom platform and the third party SDN-controller RYU. The 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, Phantom Action Daemon invokes an executable that imports an appropriate module via IPC connection as shown in fig 3.16. Further, the workflow needs Assets- 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 information needed to communicate with apps such as IP address, username and password. For configuration of input asset, an existing container or a new one can be added and items coming in via that asset will get the container label. After configuration, existing or customized playbooks are

built and incorporated.

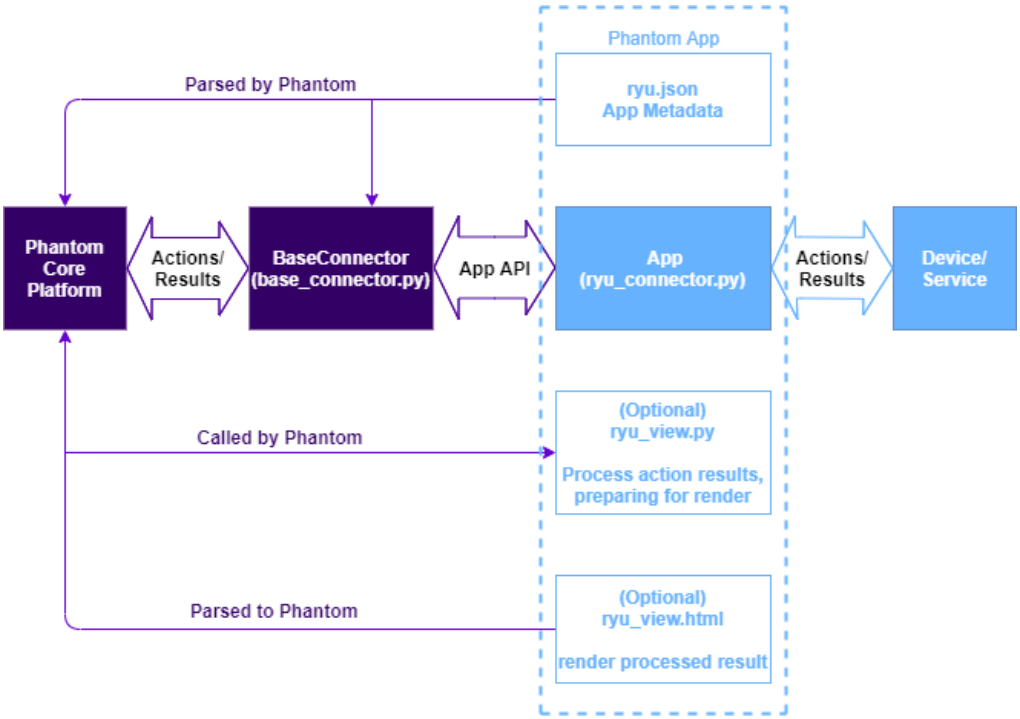

Figure 3.15.: Phantom architecture and components [Image: Phantom Splunk]

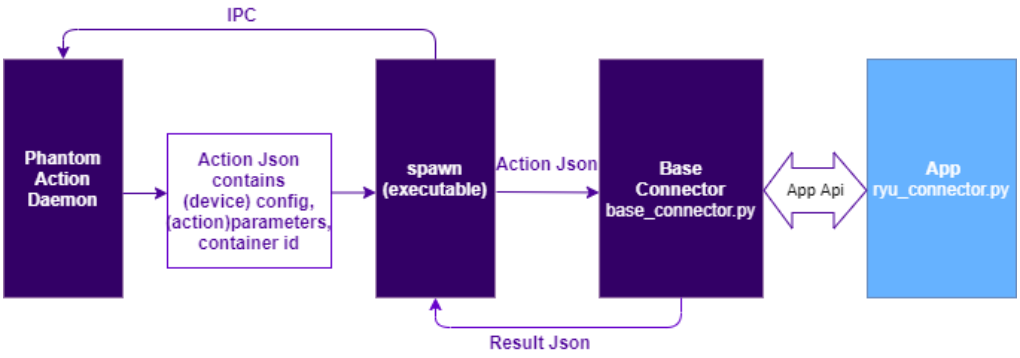

Figure 3.16.: Phantom connector module

Phantom provides built-in functions as filter or actions to schematize an incident response plan as in fig 3.17; its actions are briefed in table 3.2. To run a playbook its settings have to be configured specifying type of events, container labels, logging, etc. and be saved. Phantom also provides a GUI based python editor, 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 for each action and also the returned message metadata between subsequent playbook blocks.

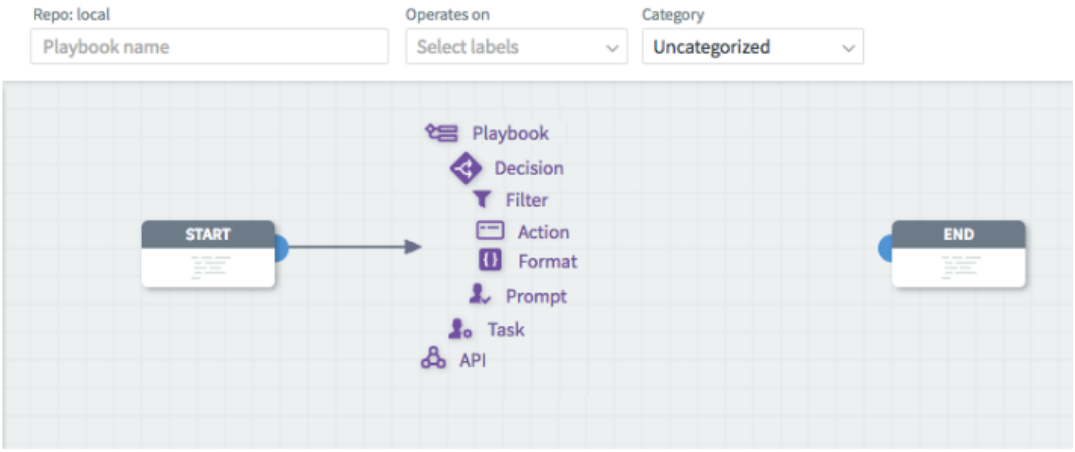

Figure 3.17.: Phantom Visual Playbook Editor

Table 3.2.: Phantom 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.

3.3. 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*.

3.3.1. 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 [48] is used on port number *80*. For demonstration, Linux network namespaces and netns in particular are used [28] [49]. 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.

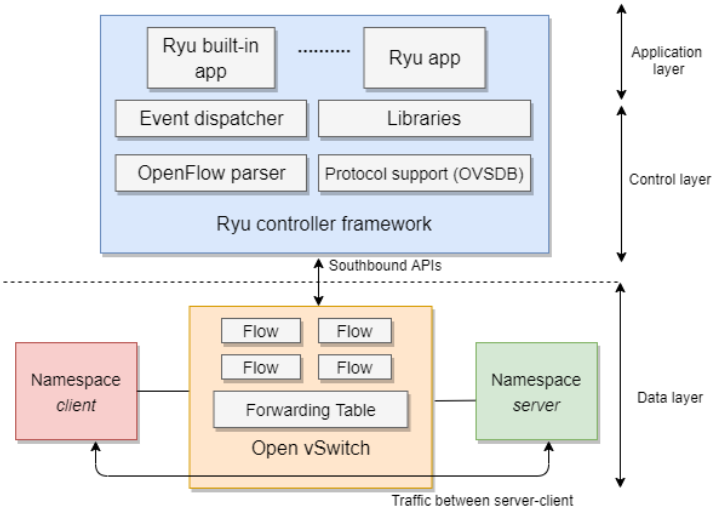

Figure 3.18.: Experimental setup in normal scenario

3.3.2. Attack traffic scenario

For design of the security function a specific kind of cyber-attack type has been assumed. Data ex-filtration attack defined in Chapter 2 is an attack involving the malicious activity of copying, retrieving or transferring of data by an unauthorized user. At the onset person intending information theft, not possessing valid credentials resorts to continuous effort of stealing restricted data regarded as Advanced Persistent Threats (APTs). In a DoS attack, system is flooded with enormous valid requests making it incapable of fulfilling them while the overload exhausts resources and bandwidth such that future genuine requests cannot be completed as described in chapter 2. In all these cases, possibilities exist for the system to respond and release sensitive information as per the request unaware of an attack. In the scope of the thesis, it has been assumed that such an attack has already taken place. As APTs or DoS involves continuous efforts to break into a system, high traffic volumes can be encountered which is 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 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 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.

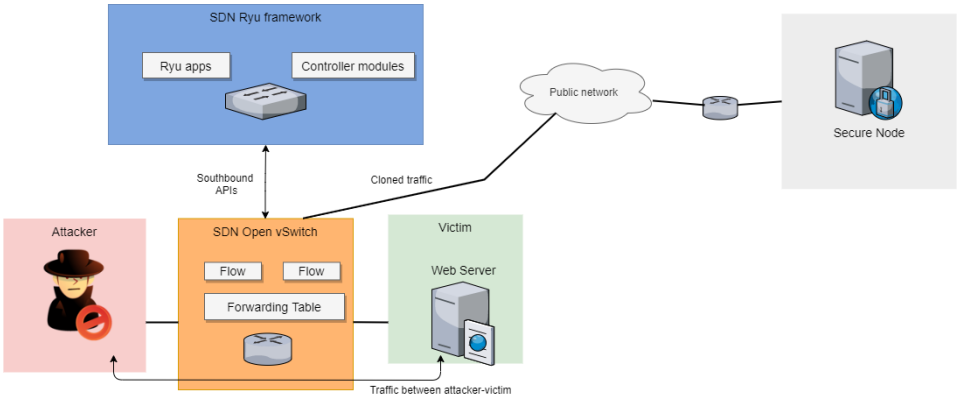

Figure 3.19.: Experimental setup in expected attack scenario

1. Use Case: Traffic cloning and encryption

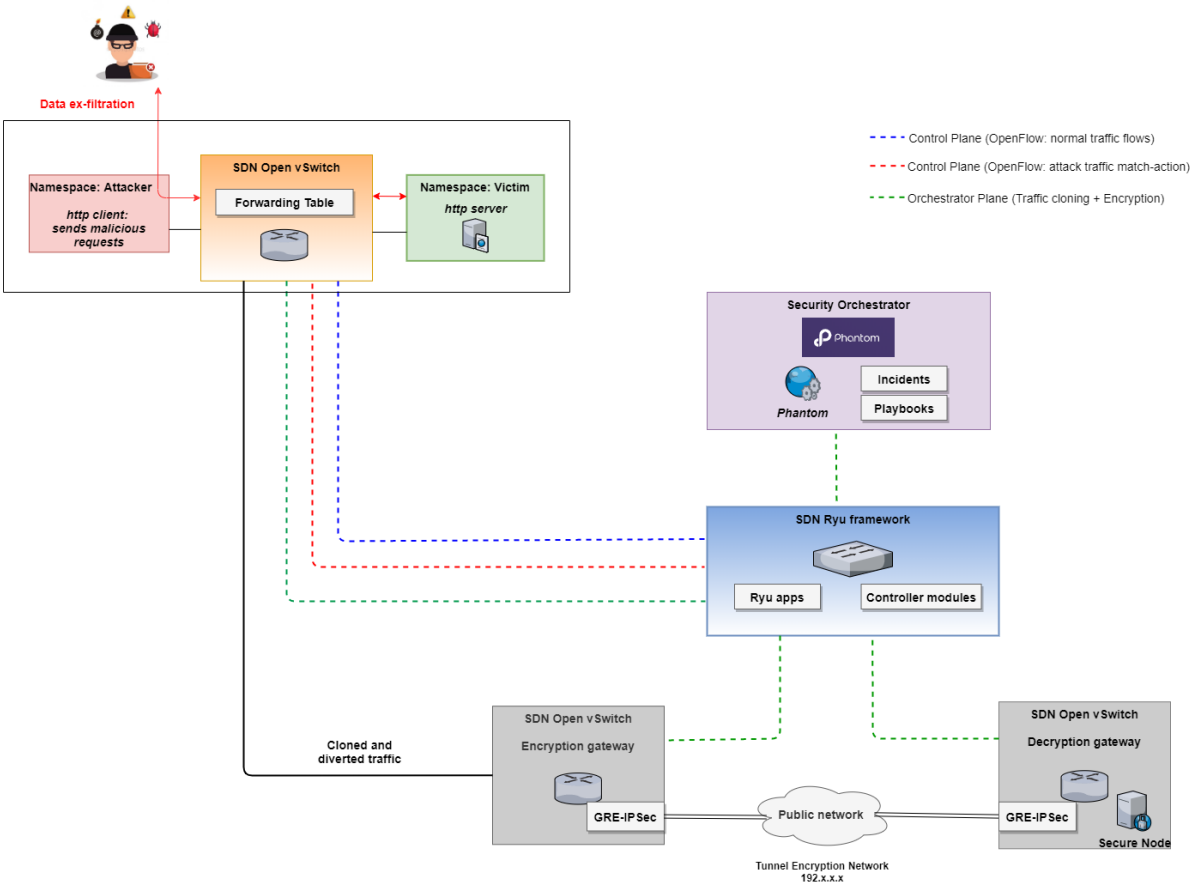

Figure 3.20.: Use Case 1: Network topology

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.

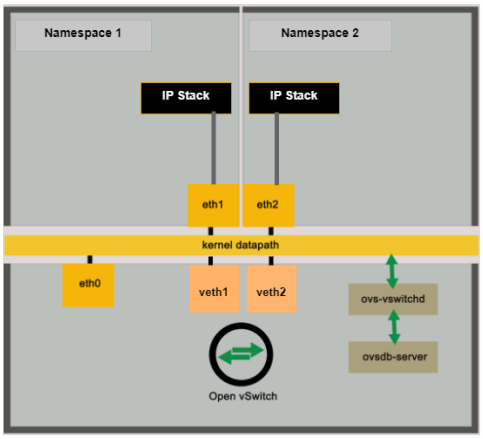

Figure 3.21.: Linux network namespaces

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 [28] 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.

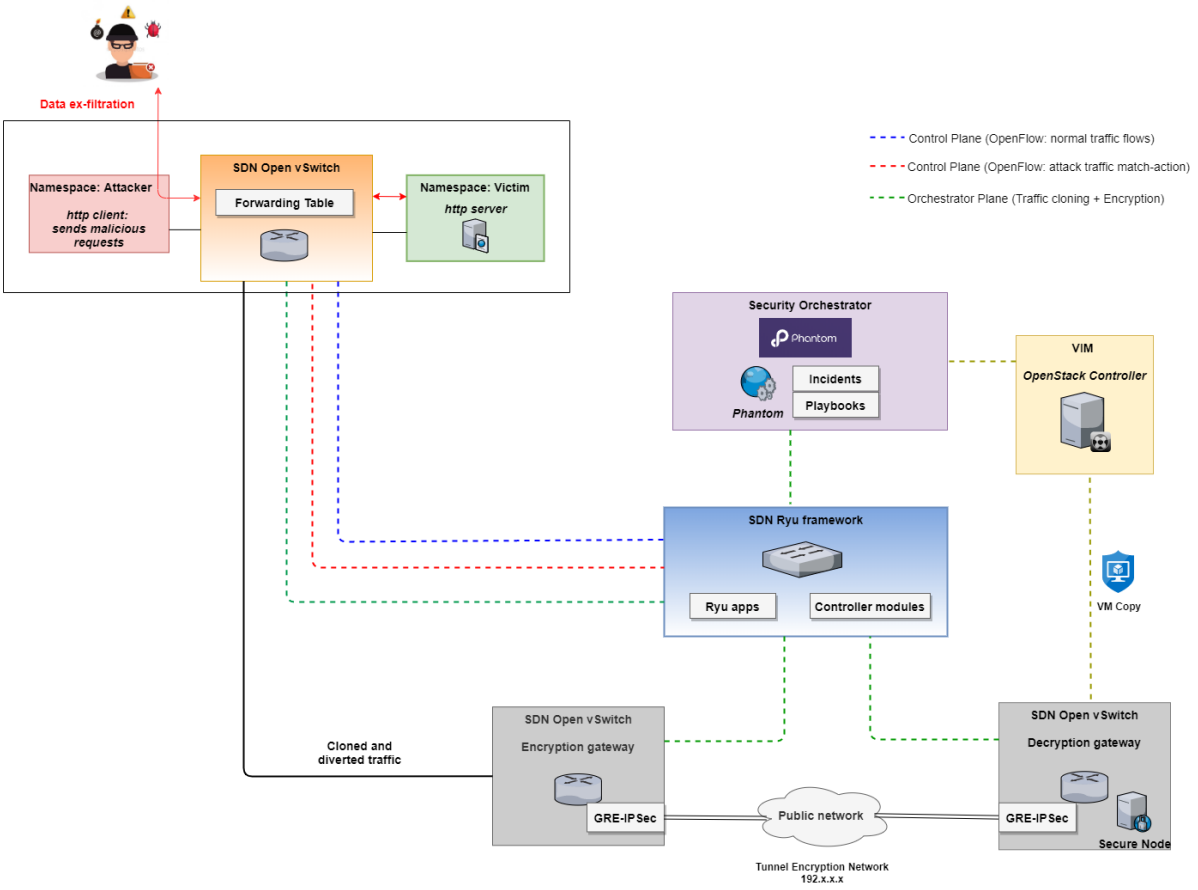

Figure 3.22.: Use Case 2: Network topology

4. Attack Detection: Traffic Prediction using Machine Learning

4.1. Classification algorithms

The technique of classification categorizes data into groups based on its features and separates data points from each other in a well-defined manner. The process of classification becomes complex when multiple features are considered and a relationship has to be established between them. Machine learning (ML) is ideal for efficient analyses in such scenarios as ML algorithms represent every instance of a data set using same features and further in supervised learning these instances are offered with known labels as opposed to unsupervised learning with unlabeled instances. In machine learning, the classification technique employs a supervised learning approach in which the algorithm sorts the data input into distinct classes using labels and further uses this learning to classify a new observation. The process of defining such a classifier or learning a set of rules from input data instances to predict new instances is called inductive machine learning. The algorithm may classify the data set into only two possible outcomes using a binary classifier or more than two distinct classes with a multi-class classifier. The machine learning classifiers use different principle types such as Linear Classifiers, Support Vector Machine, Nearest Neighbor, Decision Trees, Random Forest, Boosted Trees and Neural Networks. The purpose of using such an algorithm in the course of the thesis is to detect high volume traffic cases from machine-learning based traffic prediction. A similar purpose is addressed in [50] and [51] where highly accurate ML models have been employed for traffic prediction. In our case of an expected normal scenario, heavy traffic volume is both unexpected and unusual, therefore we assume it to be a result of malicious activity. Hence, our aim is to learn about 'good' and 'malicious' network traffic within our system using training points and to predict future traffic behavior. Further, a specific learning algorithm is to be chosen and for performing binary classification we employ the Support Vector Machine (SVM) classifier. SVM revolves around the notion of selecting an optimal margin which on its either sides separates the data into two data classes. The initial step is collection of a suitable input data sample and selection of feature subset. SVM is memory efficient as a subset of training points from the input data set is used in the classifier decision function. The evaluation of the accuracy of the selected classifier is important and for testing this the technique of splitting the data set into two-third as training data and one-third to estimate the performance is employed. The following sections explain about SVM, its mathematical approach for classification-prediction and its respective Python implementation. In subsequent sections of this Chapter, process initialization and process selection involving feature subset selection, traffic generation, train and test are presented in detail.

4.2. Support Vector Machine

Support vector machine is a classifier algorithm that operates on training data to represent the data samples as points in space and separates into categories using a dividing margin that optimizes the width between the data samples [52]. In the case of new observations, that is not part of the training data, the data points are mapped onto the same space and predicted to belong to either of the categories based on which side of the margin they fall.

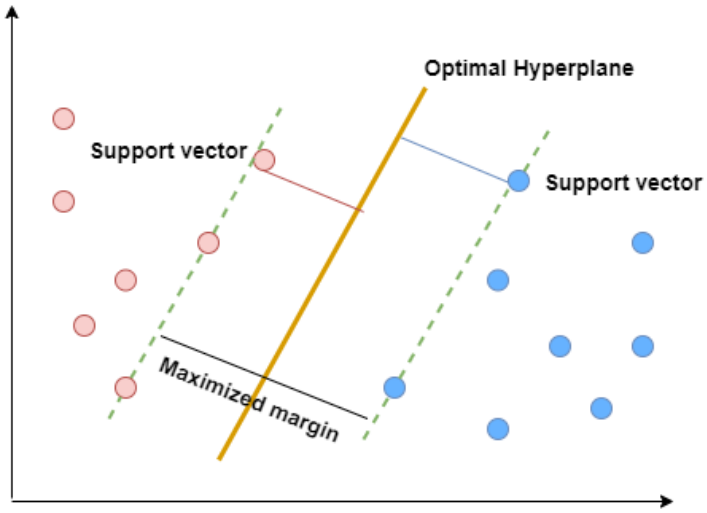

Figure 4.1.: Support Vector Machine classification

In fig 4.1, SVM classification is depicted where the instances from the data set are represented as points in space. The optimal data points from the data set are called support vectors that are placed on the decision boundaries (green) on either side of the margin. The support vectors assist to decide the position and the orientation of the central margin called the hyper-plane. The optimal hyper-plane segregates classes on either of its sides and placing the plane farthest in distance from its support vectors maximizes the margin between them. The optimization problem is therefore to decide the position and orientation of the hyper-plane to maximize the margin between support vectors. There are three main parameters that are employed in constructing a SVM classifier- Kernel, Gamma and the C parameter. When a non-linear data set is used, classification becomes complex in a 2-D space and hence the data points are mapped onto a higher dimensional space to make it easier to deduce the optimal hyper-plane. However, deducing the mapping is not required in a SVM classifier as a Kernel is used instead, which is a math function that returns the inner product of two points establishing a similarity between them and thus reducing the complexity. There are different types of kernels such as linear, non-linear, polynomial, Gaussian, sigmoid, etc. SVM also implements automatic complexity control to reduce over-fitting. The gamma parameter determines the distance up to which a single data sample exerts its influence thereby adjusting the curvature of the decision boundary. The SVM classifer employs a quadratic optimization problem that looks for maximizing the margin between classes and minimizing the amount of

incorrect classifications. The regularisation parameter 'C' is used as a penalty parameter for errors that sets a trade-off between decision boundary and misclassifications. In the course of this thesis, a linear data set was used with a linear kernel and a regularization parameter factor of 0.1. The gamma parameter is not necessary for linear conditions.

4.2.1. SVM classification and prediction

The SVM classifier categorizes data set into different classes and the key function [53] is to find a mathematical classification function $\hat{\phi} : X \rightarrow Y$, which on the basis of the set of measures $\{x_i, y_i\}_{i=1}^{m}$ with an input pattern $x_i \in X \subseteq \Re$ and the corresponding target $y_i \in$ Y={ 1,+1 } shall approximate the unknown ϕ. In the case of a binary classification, the SVM finds the hyperplane with the maximal distance (margin) from the two classes as explained in [54] and presented below as,

$$w^T.x = 0 \tag{1}$$

which can be represented using y = ax + b in a 2-D plane. The classification boundary is to be calculated such that,

$$(maximum\,distance)/2 = optimal\,hyperplane$$

Considering the hyperplane at '0' and equation(1) in the line form, the support decision boundaries for positive and negative points can be deduced as,

$$D_1 = w.x + b = 1 \Rightarrow hyperplane^{+} \tag{2}$$

$$D_2 = w.x + b = -1 \Rightarrow hyperplane^{-} \tag{3}$$

From equations (2) and (3), the general classification for support vectors is as represented in (4) and optimizing weights to have the maximum margin is deduced as follows by (5), such that the margin increases with decrease in the magnitude of weights.

$$(w^T.x + b)y_i = 1 \qquad where\ (y_i = y_+, y_-) \tag{4}$$

$$D_1 - D_2 = 2/\|w\| \tag{5}$$

The SVM classifier adjusts parameters on all support vectors points to perform classification operation with the following logic:

IF a point, $(x).(w^T.x + b)y_i = 1$:
    {point is a support vector,
    correct classification, save parameters}
ELSE IF a point, $(x).(w^T.x + b)y_i > 1$:
    {correct classification, save parameters}
ELSE:
    {incorrect classification, adjust parameters}

The optimization problem is addressed which further allows for the prediction of unknown points into positive and negative classes of a binary classification. As discussed in [54] and presented in fig 4.2, a projection P is made onto unknown points u1 and u2 from the weight vector w.

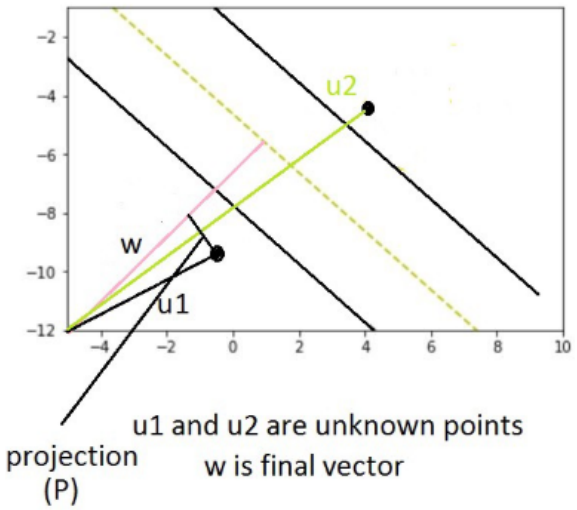

Figure 4.2.: Mathematical prediction for unknown points

The equations (1), (2) and (3) can be written in the form,

$$w^T.u_1 + b = 0 \Rightarrow Decisionboundary \tag{6}$$

$$w^T.u_1 + b > 0 \Rightarrow Positiveclass \tag{7}$$

$$w^T.u_1 + b < 0 \Rightarrow Negativeclass \tag{8}$$

The prediction of new points can thus be equated as,

$$P.\|\theta\| >= 1 \qquad if \;\; y_i = 1 \tag{9}$$

$$P.\|\theta\| <= -1 \qquad if \;\; y_i = 0 \tag{10}$$

4.3. SVM: Mathematical implementation in Python

The SVM classifier is implemented using the Python machine learning library *sklearn* and the convex optimization package *cvxopt*. The implementation together aims at addressing the optimization problem and solving it through quadratic programming [55] [56]. The primal problem is established below. Deriving From equation (4), the linear function is defined as

$$\hat{y}_n = (w^T.x_n + b) \tag{11}$$

where $(\hat{y}_n \in \{1, -1\}$; $(n \in \{1...N\}$,
x_n denotes a data point, w is a vector of weights, b is the bias and resulting in the model's prediction y_n. In order to compute derivatives, with the reciprocal method, the margin distance between support vectors from equation (5) can be written as an optimization problem [57],

$$d = min(\frac{1}{2}\|w\|)^2 \tag{12}$$

This results in defining the optimization problem as a quadratic programming (QP) problem which can be implemented and solved using Python libraries. With the combination of equation (12) and a restriction of $y_n\hat{y}_n \geq 1$, the equation (11) can be re-written to be a QP problem as,

$$L(w, b, \alpha) = \frac{1}{2}w^T.w - \sum_{n=1}^{N} \alpha_n[y_n(w^T.x_n + b) - 1] \tag{13}$$

where every restriction formed by (x_n,y_n) is associated with a Lagrange multiplier and every data point x_n whose α value is greater than zero form the support vectors. Equation (13) now defines a QP problem and further the aim is to decrease the dependency to a single parameter as opposed to three parameters in the equation. Therefore, by calculating the derivative of L with respect to w, b in equation (13) and by equating the derivatives to zero the dual problem is addressed.

$$\frac{\partial L}{\partial w} = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \Rightarrow w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{14}$$

$$\frac{\partial L}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0 \tag{15}$$

Further, the dependency is decreased to a single parameter and the QP problem now depends only on the Lagrange multiplier, α and by re-arranging the above equations into equation (13), we have:

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N} \alpha_m\alpha_n y_m y_n x_m x_n \tag{16}$$

Kernel functions are used to transform non-linearly separable spaces to linearly-separable ones. In our case a linear kernel is used and in the above equation, $x_m x_n$ together can be expressed as a kernel function. Equation (17) defines a QP problem for maximization with a linear kernel function and a restriction that enforces all α to be greater than or equal to zero.

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n K(x_m, x_n) \qquad where \;\; \alpha_n \geq 0; \sum_{n=1}^{N} \alpha_n . y_n = 0 \tag{17}$$

Generally, datasets have features that may overlap, as is in our case and cannot always be clearly classified. This may result in misclassification and therefore preference is given to a soft-margin SVM to address misclassified points over a hard-margin SVM. To further misclassification and make the optimization problem suitable for a soft-margin SVM, a tolerance variable needs to be introduced. The derivation of tolerance variable is presented in the Appendix A.1 and with this factor the dual problem is now represented as:

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n K(x_m, x_n) \tag{27a}$$

under the conditions,

$$0 \leq \alpha_n \leq C; \tag{27b}$$

$$\sum_{n=1}^{N} \alpha_n . y_n = 0 \tag{27c}$$

The convex optimization problems [58] are presented as below,

$$\begin{aligned} &minimize \;\; \frac{1}{2} x^T P x + q^T x \\ &subject \; to \; Gx \leq h \\ &Ax = b \end{aligned} \tag{28}$$

Furthermore, to solve the above dual problem for maximization from existing Python libraries it needs to be instead expressed as a minimization problem in a way such that *CVXOPT* optimization library can process it. For this reason, the optimal parameters needs to be expressed as matrices and this conversion is presented in Appendix A.2. After deducing P, q, G, h, A, b parameters from the above convex optimisation problem, the QP problem is solved via the solvers.qp() function from the *CVXOPT* library that calculates and prints cost values at each epoch until convergence. The Python implementation of solvers.qp() function is presented in Appendix A.3.

4.4. Process Initialization for classifier

4.4.1. Sample selection

Data ex-filtration, Advanced Persistent Threats (APT) and Denial-of-Service attacks have been defined in Chapter 2 for the benefit of the reader. APTs involve continuous efforts to break into a system resulting in unexpected high traffic volumes within the network as a result of malicious activity. In a DoS attack, the system is flooded with enormous valid requests making it incapable of fulfilling them while the overload in turn exhausts resources and bandwidth such that future genuine requests cannot be completed either. In the course of the thesis, as described under the Use cases section in Chapter 3, we assume that the attacker resorts to data ex-filtration/APT or a DoS attack. In the detection phase, high volumes of custom network traffic is generated and fed as input data to the machine learning SVM classifier for traffic classification. In order to perform a prediction analysis based on traffic volumes, the packets received at the victim or the node under attack is vital. Hence the sample feature selection include:

- Number of packets
- Number of bytes

The number of received packets and the number of received bytes is fetched from switch statistics using REST Communication. An observation time period is set between subsequent REST fetch calls. The difference in the switch statistics for each elapsed time period is calculated and recorded in the data set.

4.4.2. Traffic generation

In the course of the thesis, for the detection phase customized random network traffic is generated using hping3 generator to form data sets with varied features. Hping3 is a network tool capable of sending custom TCP/IP packets and display target replies while it can also handle packet fragmentation, arbitrary packet sizes and file transfers. The Hping3 generator is run from the IP address of attacker namespace: nsa using the following 'ip netns' command:

sudo ip netns exec nsa hping3 +ip+ - - +str(*IP address of NSA*)+ -S -V -p 80 -i +str(packet-rate)+ -c +str(number-packets)+ -d +str(packet-size)+

where the command attributes stand for:

- -S: - - syn; To set the SYN TCP flag
- -V: - - verbose; To enable the verbose output
- -p: - - dest port[ ]; To set the destination port to port 80
- -i: - - packet rate; To set the packet rate interval in secs or secs

- -c: - - count; To set the count to stop after sending (and receiving) response packets
- -d: - - data size; To set the packet body size

Data set samples

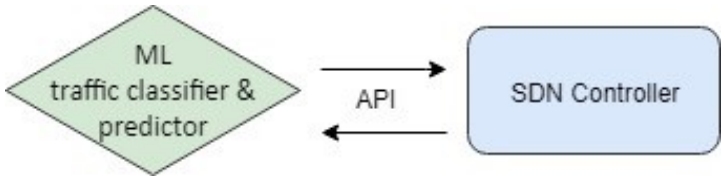

Figure 4.3.: REST API to fetch flow stats

The machine learning module collects data from the SDN-Controller- Ryu using REST API features such as GET Port stats. The received packets and received bytes at the OVS switch port connected to the victim network namespace are fetched and difference between each fetch call is calculated. The Python module 'Statistics Collector' handles the steps where flows are collected, processed and recorded into data set text files. Using the above mentioned features, firstly a data set of normal traffic is generated with around 5000 data points for the proposed features with a range as mentioned below.

- -i: packet rate range (500, 10000)
- -c: number of packets count range (20, 100)
- -d: packet size range (64, 1048)
- (r-1): sleep time between fetches with random number range (3, 10)
- IP address of Attacker network namespace directed to port 80

Correspondingly, malicious traffic data samples are generated to perform traffic classification and prediction. Out of the several tests performed, four distinct cases of malicious traffic data sets with 5000 data points each are selected to analyse how overlapping of features with the 'good' condition traffic data sets could impact traffic classification and prediction so as to optimize the classification margin. The three conditions of minimum overlapping, maximum overlapping and no overlapping for features such as packet interval, packet count rate are used as parameters for malicious traffic generation with a range as mentioned below.

- Range Overlapping for Packet interval -i:
  - Minimum Overlapping: packet rate range (500, 1000)
  - Maximum Overlapping: packet rate range (500, 3000)
- Range Overlapping for Packet count -c:
  - Maximum Overlapping: packet rate range (20, 300)
  - No Overlapping: packet rate range (100, 300)

Packet-count zero overlap case: Traffic generation characteristics

- -d: packet size range (64, 1048)
- Normal Traffic:
  1. -i: packet rate range (500, 10000)
  2. -c: number of packets count range (20, 100)
  3. (r-1): sleep time between fetches with random number range (3, 10)
- Malicious Traffic:
  1. -i: packet rate range (500, 1000) with Minimum Overlapping
  2. -c: number of packets count range (100, 300) with No Overlapping
  3. (r-1): No additional sleep time between fetches with a random number range ensuring higher traffic volume
- Total number of samples:
  Normal traffic (≈ 5000) + Malicious traffic (≈5000) = ≈ 10000 samples

NORMAL TRAFFIC			MALICIOUS TRAFFIC	
received packets	received bytes		received packets	received bytes
66	11682		133	68628
50	27874		913	750080
86	18920		793	335425
101	47742		864	664498
67	14874		1137	833245
53	26871		945	537434
58	13514		874	725375
1	42		942	522271
62	35402		751	383196
100	37200		815	480525
89	32602		1047	522075
95	87210		995	660855
69	10833		1027	607996
1	42		1028	595598
55	50545		1147	455165
21	7329		1083	799405
27	25488		1181	770007
63	10395		1113	592926
1	42		1150	681578
77	83006		867	529757
1	42		1071	1010812
70	53235		1122	460002
45	47925		1134	498751
80	50512		1047	704663
27	21087		1043	849939
21	11949		973	536182
50	19750		947	600925

Figure 4.4.: Packet-count zero overlap - hping3 generated Normal vs. Malicious traffic

Controlled overlap case: Traffic generation characteristics

- -d: packet size range (64, 1048)
- Normal Traffic:
  1. -i: packet rate range (500, 10000)
  2. -c: number of packets count range (20, 100)
  3. (r-1): sleep time between fetches with random number range (3, 10)
- Malicious Traffic:
  1. -i: packet rate range (500, 1000) with Minimum Overlapping
  2. -c: number of packets count range (20, 300) with Maximum Overlapping
  3. (r-1): No additional sleep time between fetches with a random number range ensuring higher traffic volume
- Total number of samples:
  Normal traffic (≈ 5000) + Malicious traffic (≈5000) = ≈ 10000 samples

NORMAL TRAFFIC			MALICIOUS TRAFFIC	
received packets	received bytes		received packets	received bytes
66	11682		892	556063
50	27874		1177	626442
86	18920		1062	434118
101	47742		859	484124
67	14874		682	594016
53	26871		1045	440862
58	13514		710	627160
1	42		1029	747958
62	35402		700	365326
100	37200		947	447213
89	32602		691	593747
95	87210		926	433801
69	10833		968	467081
1	42		662	482564
55	50545		1143	713575
21	7329		787	455659
27	25488		869	503842
63	10395		966	439008
1	42		736	435154
77	83006		785	383484
1	42		1111	677638
70	53235		820	504131
45	47925		814	608160
80	50512		614	270887
27	21087		840	503167
21	11949		520	450790
50	19750		837	432739

Figure 4.5.: Controlled overlap case - hping3 generated Normal vs. Malicious traffic

Packet-interval zero overlap case: Traffic generation characteristics

- -d: packet size range (64, 1048)
- Normal Traffic:
  1. -i: packet rate range (500, 10000)
  2. -c: number of packets count range (20, 100)
  3. (r-1): sleep time between fetches with random number range (3, 10)
- Malicious Traffic:
  1. -i: packet rate range (500, 3000) with Maximum Overlapping
  2. -c: number of packets count range (100, 300) with No Overlapping
  3. (r-1): No additional sleep time between fetches with a random number range ensuring higher traffic volume
- Total number of samples:
  Normal traffic (≈ 5000) + Malicious traffic (≈5000) = ≈ 10000 samples

NORMAL TRAFFIC			MALICIOUS TRAFFIC	
received packets	received bytes		received packets	received bytes
66	11682		876	609402
50	27874		917	500020
86	18920		1036	679103
101	47742		1070	621521
67	14874		1023	670858
53	26871		765	551272
58	13514		1048	758641
1	42		1110	962083
62	35402		976	667086
100	37200		870	549313
89	32602		1031	731573
95	87210		923	707576
69	10833		1103	671719
1	42		999	447466
55	50545		871	620191
21	7329		909	672584
27	25488		967	644895
63	10395		1019	499417
1	42		924	546797
77	83006		767	239314
1	42		968	525269
70	53235		1260	724586
45	47925		1105	436458
80	50512		1260	878745
27	21087		1028	510388
21	11949		1025	418116
50	19750		712	440542

Figure 4.6.: Packet-interval zero overlap - hping3 generated Normal vs. Malicious traffic

Maximum overlap case: Traffic generation characteristics

- -d: packet size range (64, 1048)
- Normal Traffic:
  1. -i: packet rate range (500, 10000)
  2. -c: number of packets count range (20, 100)
  3. (r-1): sleep time between fetches with random number range (3, 10)
- Malicious Traffic:
  1. -i: packet rate range (500, 3000) with Maximum Overlapping
  2. -c: number of packets count range (20, 300) with Maximum Overlapping
  3. (r-1): No additional sleep time between fetches with a random number range ensuring higher traffic volume
- Total number of samples:
  Normal traffic (≈ 5000) + Malicious traffic (≈5000) = ≈ 10000 samples

NORMAL TRAFFIC			MALICIOUS TRAFFIC	
received packets	received bytes		received packets	received bytes
66	11682		320	90134
50	27874		995	475711
86	18920		355	272275
101	47742		538	308695
67	14874		856	589682
53	26871		677	502784
58	13514		1055	551512
1	42		930	677053
62	35402		748	451465
100	37200		706	476757
89	32602		974	709722
95	87210		716	384433
69	10833		713	424924
1	42		696	452984
55	50545		723	544913
21	7329		773	390958
27	25488		832	574297
63	10395		646	452979
1	42		793	476981
77	83006		594	427553
1	42		776	316863
70	53235		732	478023
45	47925		750	295307
80	50512		532	389226
27	21087		907	584709
21	11949		553	411157
50	19750		975	638667

Figure 4.7.: Maximum overlap case - hping3 generated Normal vs. Malicious traffic

4.4.3. Process selection: Training and Testing

In the case of most supervised machine learning models such as SVM, a part of the data set is fed as input. This data has prediction labels for classification and is used to Train the algorithm. If the input data set samples contain high amount of noise, the algorithms try to fit as much as possible thus creating an over-fitting problem. This results in poor accuracy and precision of the algorithm's prediction. Hence, a part of the data is used for accuracy evaluation and this is regarded as the Test data. The process selection of SVM is as presented in fig 4.8, where the training data is used to train the algorithm so as to produces weights and further using these calculated weights, classification and prediction are performed. The process of calculating weights is discussed in section 4.2.1 whereas the mathematical implementation of classification and prediction is discussed in section 4.2.

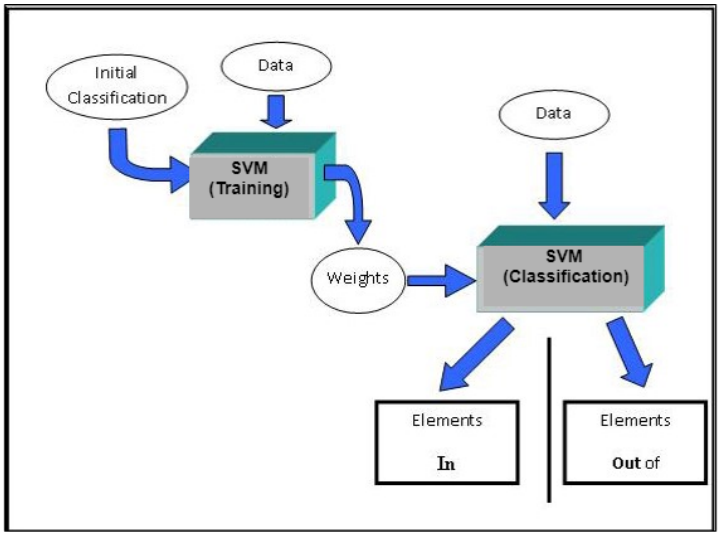

Figure 4.8.: Training and Classification process selection

It is common practice to split the input data sample as approximately as 75% for training and 25% as test data and this is handled by the *sklearn* Python library in our case. The *CVXOPT* library calculates and prints cost values at each epoch until convergence. The library refers to input samples as X-label and similarly the classification of traffic into 'malicious' and 'good' is handled by y-label comprising of 0's and 1's.

To evaluate the performance of the algorithm, the evaluation matrix calculates and provides performance metrics such as accuracy, prediction, detection rate and false alarm rate for further analysis which shall be presented in the results section under Chapter 6. Once the algorithm is trained and tested, when the Random Traffic generator is executed, the SVM algorithm makes a 'good' and 'malicious' prediction for each generated sample.

5. Mitigation: Implementing security function

5.1. Security Orchestrator: Phantom

5.1.1. 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 [59].
- 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.

5.1.2. Building a Phantom app: Ryu

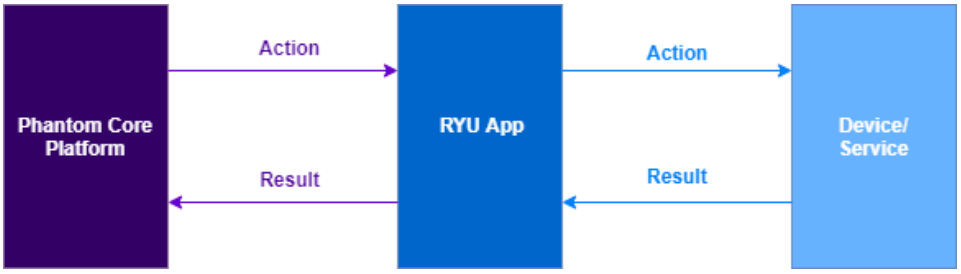

Figure 5.1.: Phantom App architecture

Phantom Apps provide a mechanism to extend platform capability by connecting third party technologies to execute actions [60]. As shown in fig 5.1, the app is instructed to perform an 'action' which then translates into set of specific commands comprehensible by the device or service and vice-versa. For this purpose, 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 instance 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.

5.2. Use Case 1: Traffic cloning and encryption

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 5.2, 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.

```
ubuntu@sdn-sw:~$ sudo ovs-ofctl dump-flows sarnet1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=18.859s, table=0, n_packets=0, n_bytes=0, idle_age=18, priority=10,tcp,tp_dst=100 actions=output:5,output:2
 cookie=0x0, duration=18.859s, table=0, n_packets=0, n_bytes=0, idle_age=18, priority=0 actions=CONTROLLER:65535
ubuntu@sdn-sw:~$
```

Figure 5.2.: 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 5.3. In case of any packet other than TCP, traffic flows normally within the switch-namespace environment.

```
ubuntu@sdn-sw:~$ sudo ovs-ofctl dump-flows sarnet1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1328.418s, table=0, n_packets=0, n_bytes=0, idle_age=1328, priority=10,tcp,tp_dst=100 actions=output:5,output:2
 cookie=0x0, duration=137.890s, table=0, n_packets=13, n_bytes=3024, idle_age=129, priority=1,in_port=2,dl_src=52:5e:3f:25:a7:61,dl_dst=c
6:d4:64:ca:f8:e1 actions=output:1
 cookie=0x0, duration=136.894s, table=0, n_packets=12, n_bytes=851, idle_age=129, priority=1,in_port=1,dl_src=c6:d4:64:ca:f8:e1,dl_dst=52
:5e:3f:25:a7:61 actions=output:2
 cookie=0x0, duration=1328.418s, table=0, n_packets=5, n_bytes=307, idle_age=136, priority=0 actions=CONTROLLER:65535
```

Figure 5.3.: 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.

5.2.1. Playbook for automated mitigation

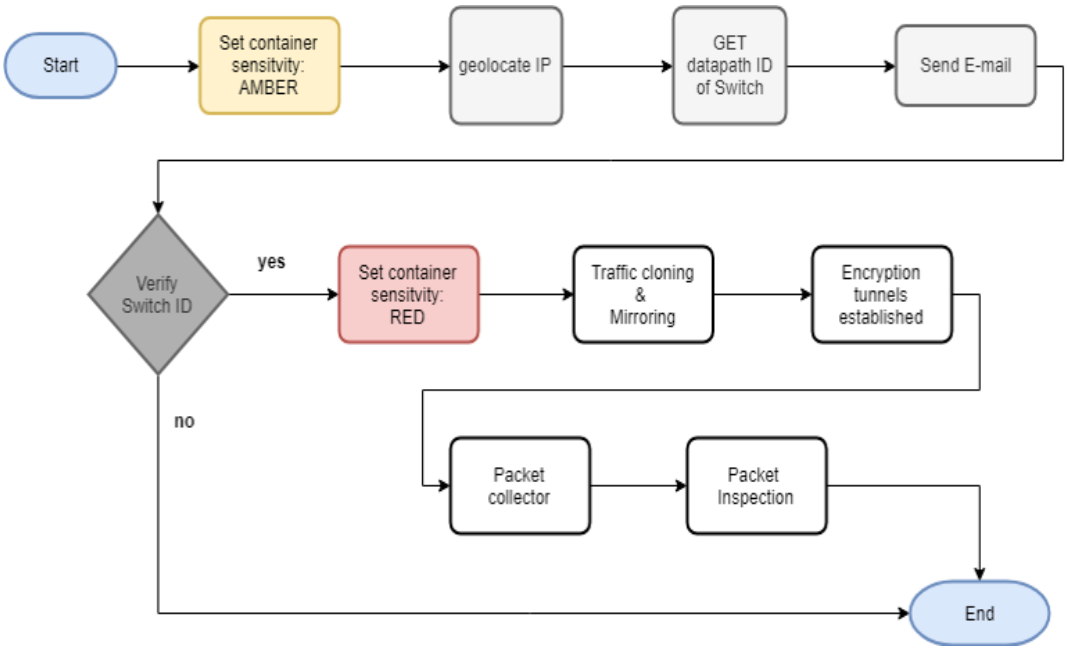

Figure 5.4.: Use case 1: Playbook flowchart

The scheme for the first playbook is drawn in fig. 5.4 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 5.5.

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 5.5, 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 5.6, the playbook layout as drawn in the Phantom Visual Playbook Editor is presented.

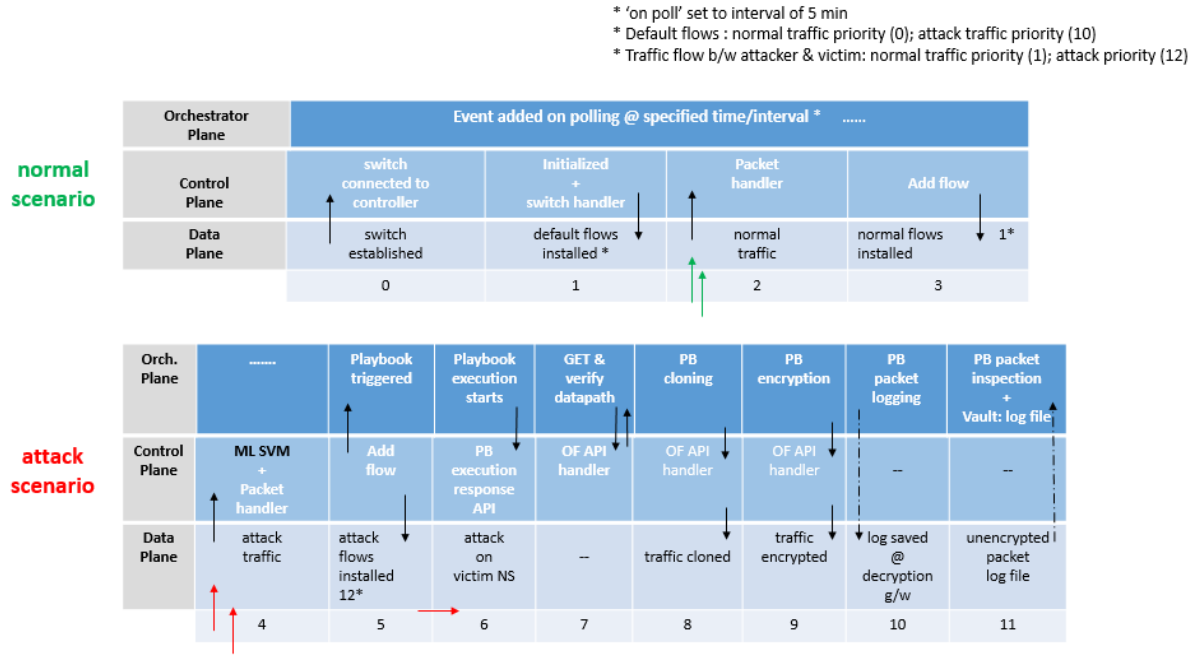

Figure 5.5.: Use case 1: Flow of action between Data, Control and Orchestrator plane

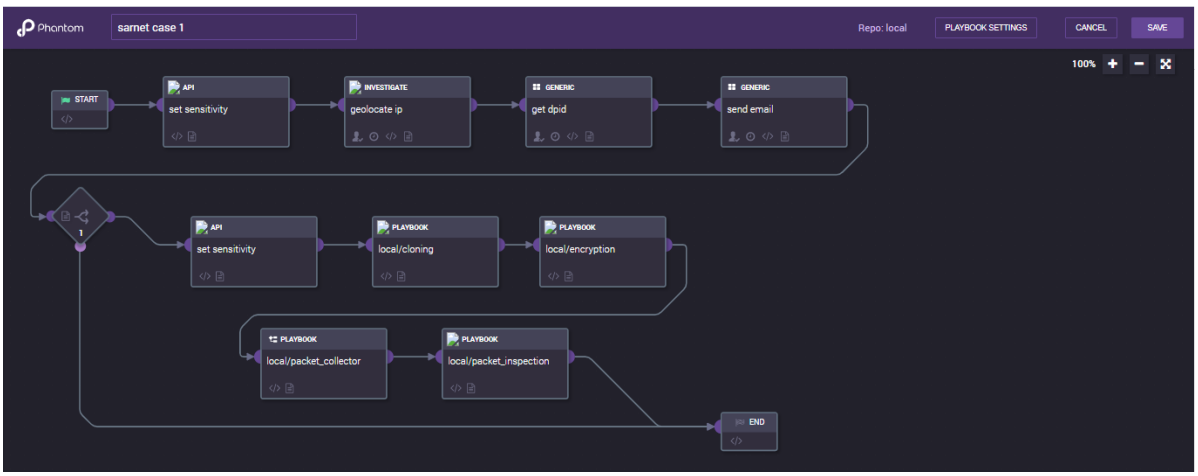

Figure 5.6.: Use case 1: Phantom playbook - sarnet

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

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.

5.3.1. Playbook for automated mitigation

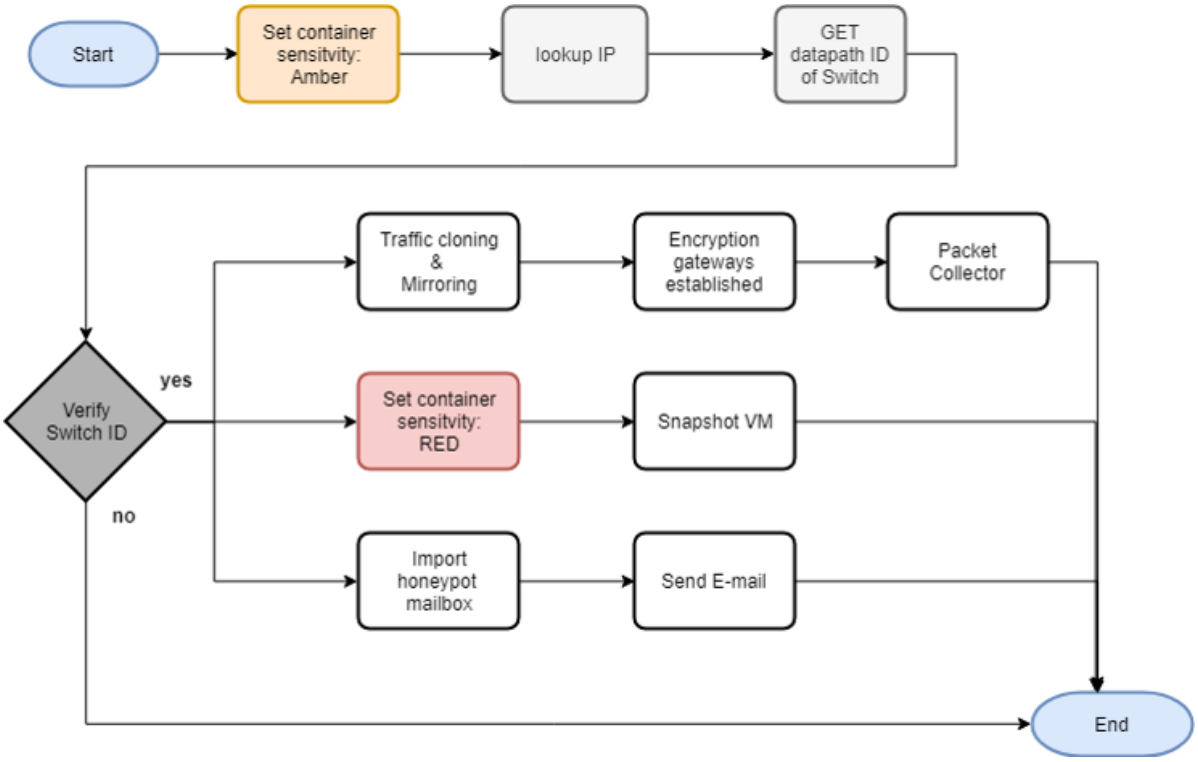

Figure 5.7.: Use case 2: Playbook flowchart

The scheme for the playbook is drawn in fig. 5.7 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 which 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. The 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 5.8., 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.

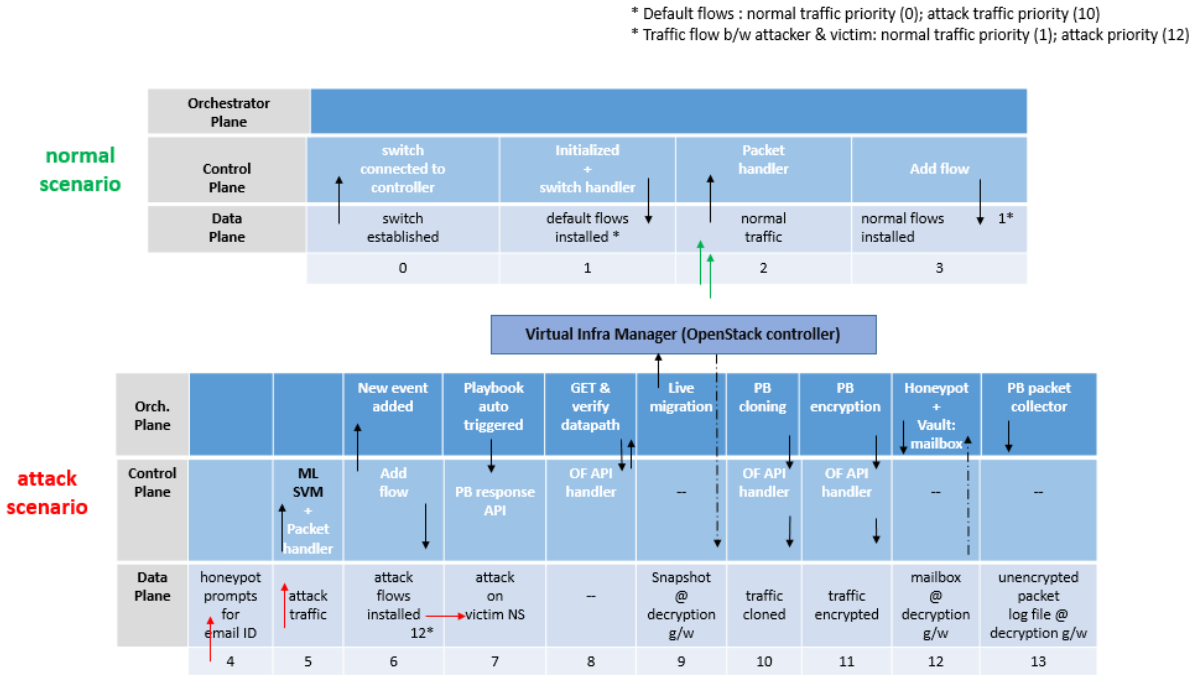

Figure 5.8.: Use case 2: Flow of action between Data, Control and Orchestrator plane

5.4. 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 5.9, 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.

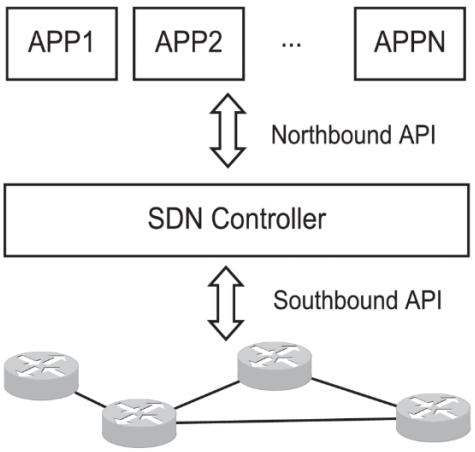

Figure 5.9.: REST APIs in OpenFlow-based SDN network

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 [48].

- 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.

5.4.1. 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 5.10.: 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 5.11.: 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 5.12.: 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 5.13.: 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 5.14.: Phantom API using GET to fetch playbook run details

6. Results and Analysis

6.1. Detection

Traffic Classification and Prediction: SVM Distribution

The binary classification of traffic by a central hyper-plane along with its supporting decision boundaries (dotted lines) is presented in 6.1 for generated traffic with the below characteristics.

- Normal traffic:
  1. -i: packet rate range (500, 10000)
  2. -c: number of packets count range (20, 100)
  3. (r-1): sleep time between fetches with random number range (3, 10)
- Malicious traffic:
  1. -i: packet rate range (500, 1000) with Minimum Overlapping
  2. -c: number of packets count range (100, 300) with No Overlapping
  3. (r-1): no additional sleep time between fetches to yield higher traffic volume

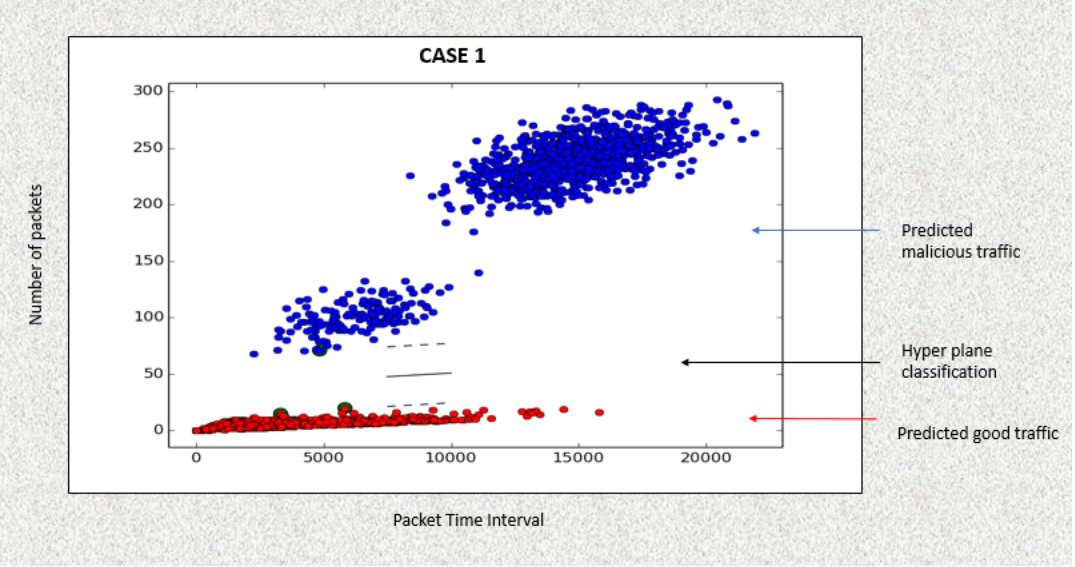

Figure 6.1.: SVM Distribution for Packet-count zero overlap case

Traffic generation and sample selection

	Packet Interval		Number of Packets	
	Normal	Malicious	Normal	Malicious
Packet-count zero overlap / Case 1	500-10000	500-1000	20-100	100-300
Controlled overlap / Case 2	500-10000	500-1000	20-100	20-300
Packet-interval zero overlap / Case 3	500-10000	500-3000	20-100	100-300
Maximum overlap / Case 4	500-10000	500-3000	20-100	20-300

Figure 6.2.: Comparison of sample feature characteristics

The selected sample features are packet interval and packet count rate. The fig 6.2 depicts comparison of traffic sample generation characteristics for 'normal' and 'malicious' with the respect to sample features- packet interval and packet count rate.

The three conditions of overlapping for sample features used while traffic generation are minimum, maximum and no overlapping. A combination of these three conditions were used to draw the four cases: Packet-count zero overlap, Controlled overlap, Packet-interval zero overlap and Maximum overlap presented in chapter 4 and are respectively depicted as Bad Case 1-4 in fig 6.3 and 6.4 whose range for generating malicious traffic is stated accordingly.

- Range overlapping for Packet interval -i:
  - Minimum Overlapping: packet rate range (500, 1000)
  - Maximum Overlapping: packet rate range (500, 3000)

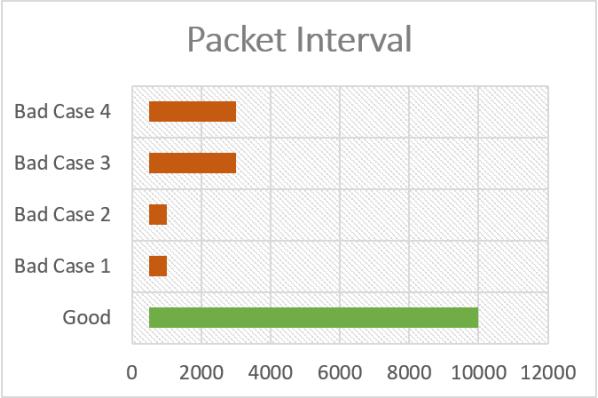

Figure 6.3.: Overlapping for the feature Packet Interval range

- Range overlapping for Packet count -c:
  - Maximum Overlapping: packet rate range (20, 300)
  - No Overlapping: packet rate range (100, 300)

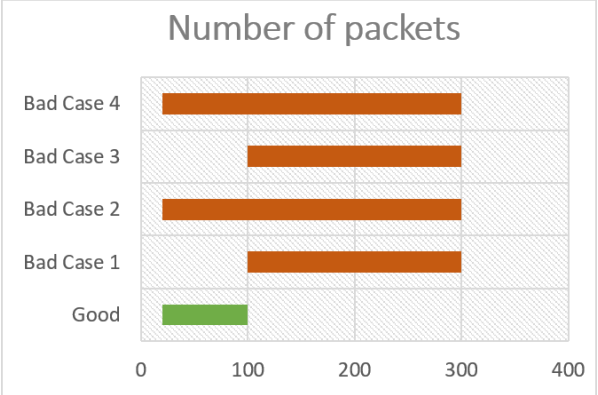

Figure 6.4.: Overlapping for the feature Packet count range

6.2. Evaluation Metrics

In order to understand and analyse the performance of a machine learning algorithm the model is evaluated with respect to different aspects such as accuracy, precision, detection rate among other metrics [61]. Some of the different types of evaluation metrics used in our case are as listed below.

Classification Accuracy

Classification Accuracy is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}$$

The metric works well for datasets with equal number of samples belonging to each class. Since the dataset used in our case is balanced this disadvantage is avoided. However, while it is important to test the accuracy of a model for a binary classification, this metric alone is not sufficient to evaluate our model.

Confusion Matrix

As a more extensive metric, a statistical table called the confusion matrix is considered for analysing the performance of a classification model that yields a matrix as output and describes the complete performance of the model and also forms the basis for the other types of metrics. The two classes for such a matrix is to consider a positive and a negative case. With respect to the binary classification of this thesis, we consider:

- Positive Case: Detecting a malicious traffic sample
- Negative Case: Detecting a good traffic sample

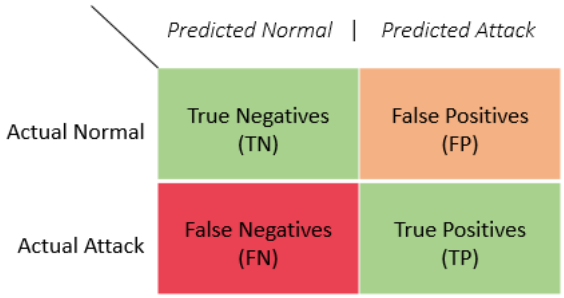

Figure 6.5.: Confusion Matrix

With the definition of the two classes, the matrix elements are as presented in fig 6.5 and calculated as described below.

- True Positives: Number of cases in which the model predicts a sample as malicious traffic (1) and the actual traffic instance is also a malicious sample (1) and the prediction is correct.
- True Negatives: Number of cases in which the model predicts a sample as good traffic (0) and the actual traffic instance is also a good sample (0) and the prediction is correct.
- False Positives: Number of cases in which the model predicts a sample as malicious traffic (1) and the actual traffic instance is a good sample (0) and hence the prediction is incorrect.
- False Negatives: Number of cases in which the model predicts a sample as good traffic (0) and the actual traffic instance is a malicious sample (1) and hence the prediction is incorrect.

In highly critical applications such as diagnosis of Novel Corona Virus in humans or in a Intrusion Detection System, the risk of a FN is higher as compared to a FP. The FN fails to recognize an existing infection or a threat and can lead to catastrophic results. To achieve a model with high performance, the aim is to maintain higher number of total cases under TP and TN whereas lower number of cases under FN and FP. Further, the confusion matrix forms the basis for the other types of metrics which are calculated as below.

- Accuracy: The ratio of correct/true predictions to the total number of (good and malicious traffic) predictions by the classifier.

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP} = \frac{TN + TP}{N + P}$$

- Precision: The ratio of correct malicious traffic predictions to the total positive results predicted by the classifier.

$$Precision = \frac{TP}{FP + TP}$$

- Recall: The ratio of number of correct positive to the number of all relevant samples (all samples that should have been identified as positive).

$$Recall = \frac{TP}{FN + TP}$$

- True Positive Rate/Sensitivity: Sensitivity is in principle the same as Recall as defined above.

$$Sensitivity = \frac{TP}{FN + TP}$$

  Since the aim is to successfully detect an attack/onset of malicious activity, Recall and Sensitivity shall be also treated as Detection Rate.

- True Negative Rate/Specificity: The proportion of negative data points that are incorrectly considered as positive, with respect to all negative data points.

$$Specificity = \frac{TN}{TN + FP}$$

- False Positive Rate: The proportion of positive data points that are incorrectly considered as positive, with respect to all negative data points.

$$FPR = \frac{FP}{FP + TN}$$

  Also, we have FPR = 1-Specificity.

- False Negative Rate: The proportion of negative data points that are incorrectly considered as negative, with respect to all positive data points.

$$FNR = \frac{FN}{FN + TP}$$

  Also, we have FNR = 1-Sensitivity.

F1 Score

F1 Score is the harmonic Mean between precision and recall whose range for the score is [0, 1] giving an indication of how precise the classifier is as well as how robust it is. The greater the F1 Score, the better is the performance of a model.

- F1 Score tries to find the balance between precision and recall.

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2TP}{2TP + FP + FN}$$

6.3. Performance Evaluation: Test A, B, C

Test A

Test A	Packet Interval		Number of Packets		Training	Test
	Good	Malicious	Good	Malicious	Good + Malicious	Good + Malicious
Case 1	500-10000	500-1000	20-100	100-300	3000	814
Case 2	500-10000	500-1000	20-100	20-300	3000	814
Case 3	500-10000	500-3000	20-100	100-300	3000	814
Case 4	500-10000	500-3000	20-100	20-300	3000	814

Figure 6.6.: Sample features and number of Training+Test samples - Test A

1. Confusion Matrix

$$CM = \begin{bmatrix} TrueNegatives(TN) & FalsePositives(FP) \\ FalseNegatives(FN) & TruePositives(TP) \end{bmatrix}$$

		Predicted	
	Actual	Normal	Attack
Case 1	Normal	401	6
	Attack	0	407
Case 2	Normal	350	57
	Attack	0	407
Case 3	Normal	397	10
	Attack	0	407
Case 4	Normal	400	7
	Attack	0	407

Figure 6.7.: Confusion Matrix - Test A

2. Accuracy of the model:

$$AverageAccuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}/4$$

Average Accuracy of Test A = 97.54

3. Evaluation Metrics:
The measured and calculated metrics values from the confusion matrix for Test B are as below.

Test A	Precision	Accuracy	Sensitivity	Specificity	FPR	FNR
Case 1	98.54	99.26	100.0	98.52	1.47	0
Case 2	87.71	92.99	100.0	85.99	14.00	0
Case 3	97.60	98.77	100.0	97.54	2.45	0
Case 4	98.30	99.14	100.0	98.28	1.719	0

Figure 6.8.: Recorded Values (in %) for evaluation metrics - Test A

4. F1 Score:
The greater the F1 Score, the better is the performance of a model.

$$F1score = \frac{2TP}{2TP + FP + FN}$$

Table 6.1.: F1 Score calculated for Test A

Case	Value
Case 1	99.26
Case 2	93.45
Case 3	98.78
Case 4	99.14

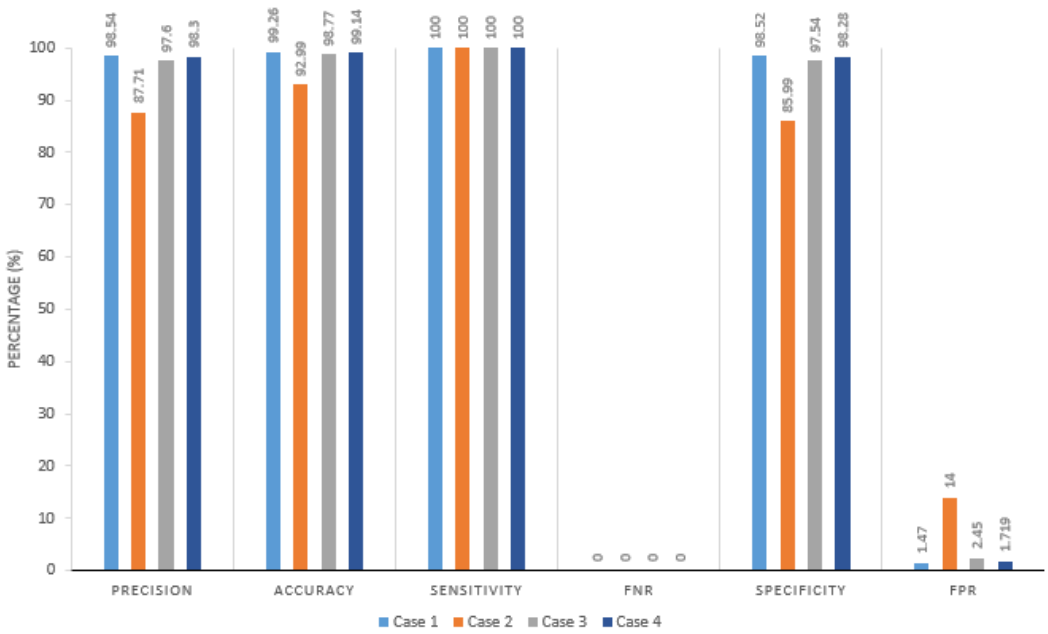

Figure 6.9.: Performance - Test A

- Detecting the onset of malicious activity is the aim of this experiment and a Detection Rate (Sensitivity) of 100% is observed throughout.
- We know, FNR = 1-Sensitivity
  Hence, sensitivity is observed to be inversely proportional to the False Negative Rate.
- Similarly, FPR = 1-Specificity
  Hence, specificity is observed to be inversely proportional to the False Positive Rate.
- Best Case and Feature Overlapping: With respect to all the metrics, Case 1 is observed to have the best performance. Overlapping is noted to be,
  Packet interval - minimum; Number of packets - no overlap
- Worst Case and Feature Overlapping: With respect to all the metrics, Case 2 is observed to have the worst performance. Overlapping is noted to be,
  Packet interval - minimum; Number of packets - maximum
- Test A has a relatively smaller Train (3000) and smaller Test (814) dataset sample .

Test B

Test B	Packet Interval		Number of Packets		Training	Test
	Good	Malicious	Good	Malicious	Good + Malicious	Good + Malicious
Case 1	500-10000	500-1000	20-100	100-300	5600	2000
Case 2	500-10000	500-1000	20-100	20-300	5600	2000
Case 3	500-10000	500-3000	20-100	100-300	5600	2000
Case 4	500-10000	500-3000	20-100	20-300	5600	2000

Figure 6.10.: Sample features and number of Training+Test samples - Test B

1. Confusion Matrix

$$CM = \begin{bmatrix} TrueNegatives(TN) & FalsePositives(FP) \\ FalseNegatives(FN) & TruePositives(TP) \end{bmatrix}$$

		Predicted	
	Actual	Normal	Attack
Case 1	Normal	883	117
	Attack	1	999
Case 2	Normal	863	137
	Attack	198	802
Case 3	Normal	971	29
	Attack	1	999
Case 4	Normal	114	886
	Attack	0	1000

Figure 6.11.: Confusion Matrix - Test B

2. Accuracy of the model:

$$AverageAccuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}/4$$

Average Accuracy of Test B = 82.88

3. Evaluation Metrics:
The measured and calculated metrics values from the confusion matrix for Test B are as below.

Test B	Precision	Accuracy	Sensitivity	Specificity	FPR	FNR
Case 1	89.51	94.1	99.9	88.3	11.7	0.1
Case 2	85.41	83.25	80.2	86.3	13.7	19.8
Case 3	97.17	98.5	99.9	97.1	2.9	0.1
Case 4	53.02	55.7	100.0	11.4	88.6	0

Figure 6.12.: Recorded Values (in %) for evaluation metrics - Test B

4. F1 Score:
The greater the F1 Score, the better is the performance of a model.

$$F1score = \frac{2TP}{2TP + FP + FN}$$

Table 6.2.: F1 Score calculated for Test B

Case	Value
Case 1	94.42
Case 2	82.72
Case 3	98.52
Case 4	20.46

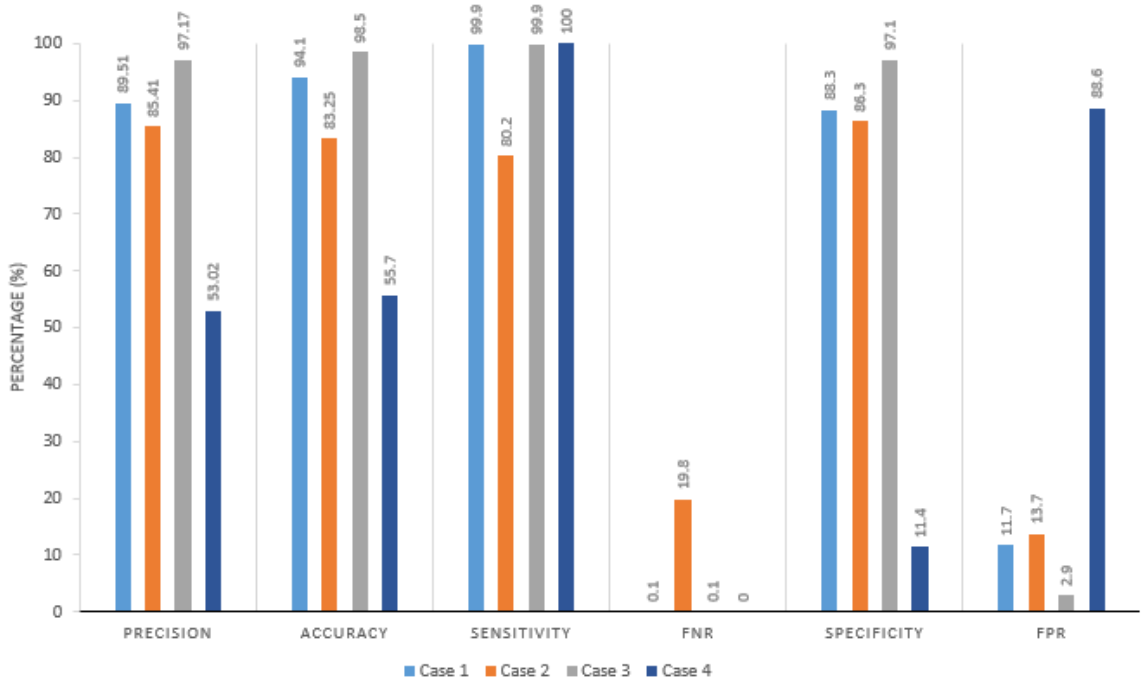

Figure 6.13.: Performance - Test B

- Detecting the onset of malicious activity is the aim of the experiment and a Detection Rate (Sensitivity) of ≈100% is observed. However, Case 2 observes only 80% rate.
- FNR = 1-Sensitivity
  Hence, sensitivity is observed to be inversely proportional to the False Negative Rate.
- FPR = 1-Specificity
  Hence, specificity is observed to be inversely proportional to the False Positive Rate.
- Best Case and Feature Overlapping: With respect to all the metrics, Case 3 is observed to have the best performance. Overlapping is noted to be,
  Packet interval - maximum; Number of packets - no overlap
- Worst Case and Feature Overlapping: Case 4 is observed to have the worst performance with respect to most of the metrics, however Case 2 has the least detection rate. Overlapping is noted to be,
  Packet interval - maximum; Number of packets - maximum
- Test B has a relatively bigger Train (5600) and bigger Test (2000) dataset sample.

Test C

Test C	Packet Interval		Number of Packets		Training	Test
	Good	Malicious	Good	Malicious	Good + Malicious	Good + Malicious
Case 1	500-10000	500-1000	20-100	100-300	5600	1600
Case 2	500-10000	500-1000	20-100	20-300	5600	1600
Case 3	500-10000	500-3000	20-100	100-300	5600	1600
Case 4	500-10000	500-3000	20-100	20-300	5600	1600

Figure 6.14.: Sample features and number of Training+Test samples - Test C

1. Confusion Matrix

$$CM = \begin{bmatrix} TrueNegatives(TN) & FalsePositives(FP) \\ FalseNegatives(FN) & TruePositives(TP) \end{bmatrix}$$

		Predicted	
	Actual	Normal	Attack
Case 1	Normal	710	90
	Attack	1	799
Case 2	Normal	696	104
	Attack	148	652
Case 3	Normal	783	17
	Attack	1	799
Case 4	Normal	89	711
	Attack	0	800

Figure 6.15.: Confusion Matrix - Test C

2. Accuracy of the model:

$$AverageAccuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}/4$$

Average Accuracy of Test C = 83.24

3. Evaluation Metrics:
The measured and calculated metrics values from the confusion matrix for Test C are as below.

Test C	Precision	Accuracy	Sensitivity	Specificity	FPR	FNR
Case 1	89.87	94.31	99.87	88.75	11.25	0.12
Case 2	86.24	84.25	81.5	87.0	13.0	18.5
Case 3	97.91	98.87	99.875	97.87	2.125	0.12
Case 4	52.94	55.56	100.0	11.1	88.87	0

Figure 6.16.: Recorded Values (in %) for evaluation metrics - Test C

4. F1 Score: The greater the F1 Score, the better is the performance of a model.

$$F1Score = \frac{2TP}{2TP + FP + FN}$$

Table 6.3.: F1 Score calculated for Test C

Case	Value
Case 1	94.61
Case 2	83.80
Case 3	98.88
Case 4	69.23

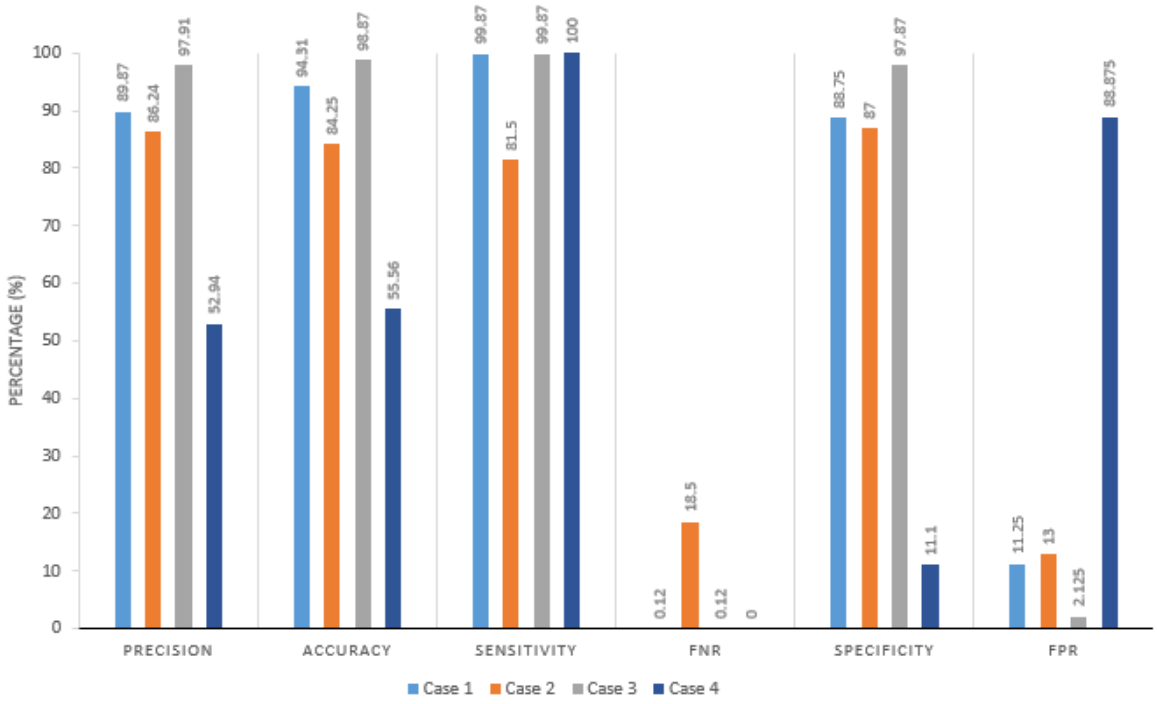

Figure 6.17.: Performance - Test C

- Detecting the onset of malicious activity is the aim of the experiment and a Detection Rate (Sensitivity) of ≈100% is observed. However, Case 2 observes only 81.5% rate.
- FNR = 1-Sensitivity
  Hence, sensitivity is observed to be inversely proportional to the False Negative Rate.
- FPR = 1-Specificity
  Hence, specificity is observed to be inversely proportional to the False Positive Rate.
- Best Case and Feature Overlapping: With respect to all the metrics, Case 3 is observed to have the best performance. Overlapping is noted to be,
  Packet interval - maximum; Number of packets - no overlap
- Worst Case and Feature Overlapping: Case 4 is observed to have the worst performance with respect to most of the metrics, however Case2 has the least detection rate. Overlapping is noted to be,
  Packet interval - maximum; Number of packets - maximum
- Test C has a relatively bigger Train (5600) and smaller Test (1600) dataset sample.

In A.4, similar test results for Test D and Test E is presented.

Performance Analysis

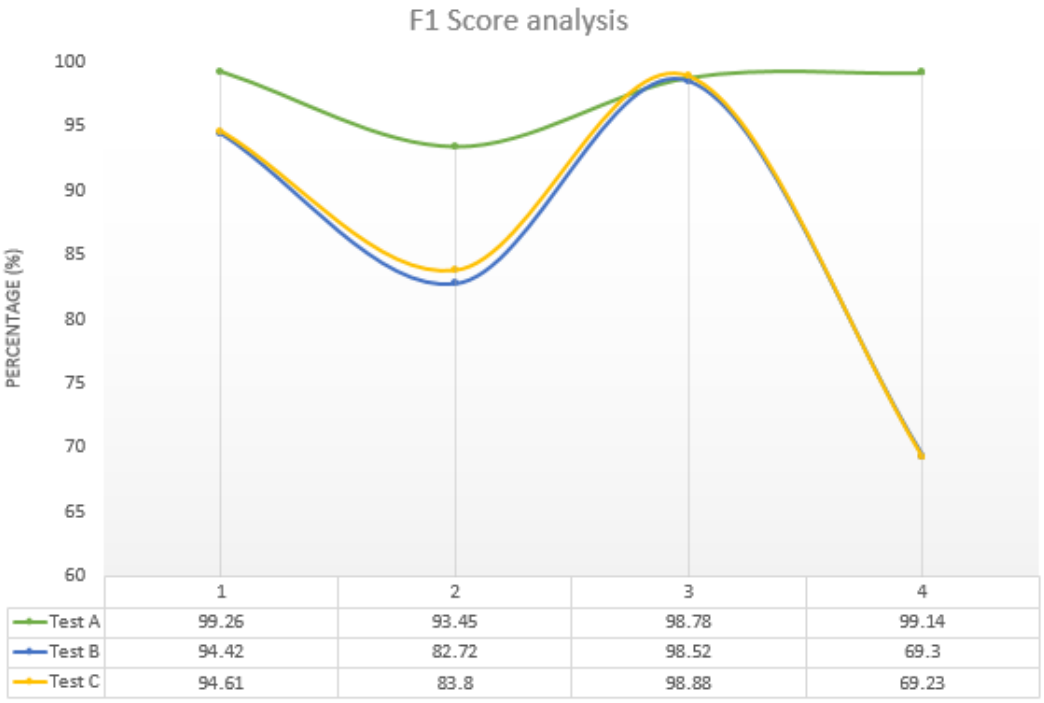

Figure 6.18.: F1 Score comparison - Test A,B,C

Test A is performed on a smaller dataset of 3814 samples while the other two are larger datasets. Test B has a dataset size of 7600 samples and Test C has a dataset size of 7200 samples. Although Tests A, B and C have similar performances, the size of datasets play a role in deciding the F1 Score and Accuracy.

- F1 Score: Test A is observed to have the best performance with a dip observed only in Case 2. Among the bigger dataset samples both Test B and Test C have similar performance observing a dip in Case 2 and a deeper dip in Case 4 resulting from similar F1 scores as presented in fig 6.1.
- Accuracy: From the tests, scores noted in Test A exhibits best average accuracy at 97.54%. With bigger samples, average accuracy can be observed at around 82%. Also, accuracy increases as overlapping decreases. The overall accuracy of the classifier is recorded at 87.88%.

Detection Rate:

The aim of the thesis is detecting the onset of malicious activity, thus making detection rate a vital performance parameter. Throughout the Tests A, B and C most cases tend to a ≈100% detection rate (sensitivity) but a dip is observed in Case 2 of both Test B and C (≈80%). The overall detection rate of the SVM classifier is recorded at 96.77% making it an efficient and reliable approach to detect volume based malicious traffic.

The performance gap between tests can be attributed to the characteristic variations of sample features during traffic generation. Cases with minimum or no overlapping between sample features of good and malicious traffic records the best performance as opposed to cases recording lower performance with maximum overlapping of sample features. The sample features individually bearing different effects on the outcome can be observed.

- Packet Interval feature: The overlapping with respect to the feature characteristics varies from minimum to maximum. As there is overlapping to some extent in all cases it appears to have lesser impact on the overall performance.
- Packet Count feature: Feature characteristics varies from no overlapping to maximum and appears to have greater impact on the performance. Case1 and Case3 has no feature overlapping offering better performance whereas Case2 and Case4 have maximum overlapping recording all of the worse cases in Tests A, B and C.

6.4. Mitigation

The aim is to mitigate the attack, after detecting the onset of malicious activity, by transferring the infected traffic and/or infected node to a safe node over an encrypted secure medium. For this purpose and also to respond at speeds beyond human capabilities, playbooks were designed as incident response. The total duration for execution of playbook 1 and playbook 2 is as presented in fig 6.19 and 6.20 respectively.

- Playbook 1: Of the total duration of the incident response during playbook execution the time taken by traffic encryption is 13.53 secs and traffic cloning is 1.8 secs. The playbook operates by executing blocks in a consecutive fashion.
- Playbook 2: As part of the incident response during playbook execution the highest time taken in is dedicated for VM snapshot and migration (85.07). The traffic cloning and encryption together takes up approximately 5 secs. The incident response of the playbook in Phantom is designed in a branched fashion, thereby traffic cloning+encryption takes lesser time than playbook 1.

The playbooks are designed to allot the maximum time for packet collection (60 secs) which is not part of the original incident response to the attack but for further analysis of the infected packets. The time allotted for packet collection is 66.1 secs and 60 secs in each case.

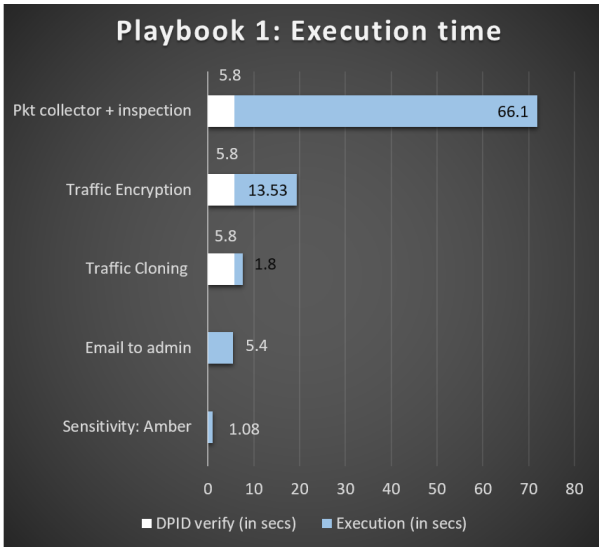

Figure 6.19.: Execution time taken by Playbook 1

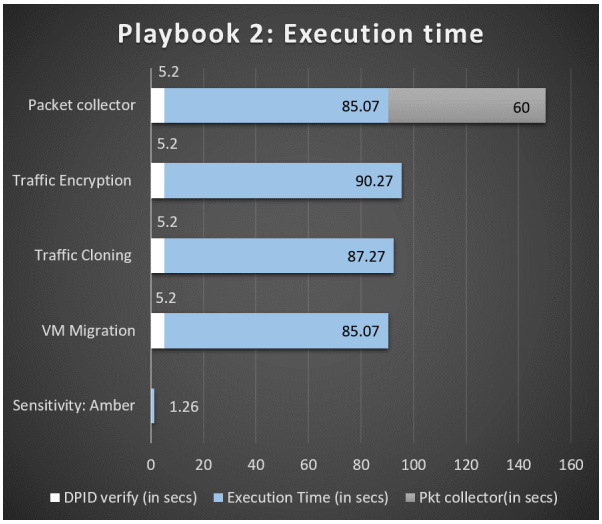

Figure 6.20.: Execution time taken by Playbook 2

Infected traffic: The important part of mitigation is to clone, encrypt packets and collect them at the decryption gateway safe node to perform further analysis. Using the packet analyzer tool Wireshark, these packets were collected. Fig 6.21 shows the encrypted packets routed through a dedicated tunneling network with 192.178.0.18 being the near-end and 192.178.0.23 the far-end of the GRE-IPSec tunnel. Given the possibility of encountering public networks, the infected traffic is encrypted to avoid further compromise in the course of the path. In fig 6.22, the decrypted packets reveal the original infected TCP traffic collected at the safe node. The source and destination address is the same as the attacker (10.20.30.30) and victim namespace (10.20.30.33).

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.178.0.18	192.178.0.23	ISAKMP	226	Identity Protection (Main Mode)
2	0.000498586	192.178.0.23	192.178.0.18	ISAKMP	166	Identity Protection (Main Mode)
3	0.004790243	192.178.0.18	192.178.0.23	ISAKMP	270	Identity Protection (Main Mode)
4	0.006431992	192.178.0.23	192.178.0.18	ISAKMP	270	Identity Protection (Main Mode)
5	0.008535734	192.178.0.18	192.178.0.23	ISAKMP	118	Identity Protection (Main Mode)
6	0.009081755	192.178.0.23	192.178.0.18	ISAKMP	118	Identity Protection (Main Mode)
7	1.002650054	192.178.0.18	192.178.0.23	ISAKMP	374	Quick Mode
8	1.004617971	192.178.0.23	192.178.0.18	ISAKMP	342	Quick Mode
9	1.005264895	192.178.0.18	192.178.0.23	ISAKMP	102	Quick Mode
10	2.031743385	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
11	2.031816247	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
12	2.032322406	192.178.0.18	192.178.0.23	ESP	230	ESP (SPI=0x0b7893be)
13	2.033791285	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
14	2.033835542	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
15	2.033848535	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
16	2.033861406	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
17	2.033873818	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
18	2.033885597	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
19	2.033897609	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
20	2.038333792	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
21	4.069558098	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
22	4.069628611	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
23	4.070102269	192.178.0.18	192.178.0.23	ESP	230	ESP (SPI=0x0b7893be)
24	4.071325529	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)
25	4.071370482	192.178.0.18	192.178.0.23	ESP	150	ESP (SPI=0x0b7893be)

Figure 6.21.: Encapsulated Security Payload (ESP) collected at the far-end of GRE-IPSec tunnel

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.20.30.30	10.20.30.33	TCP	74	36658 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=334929 TSecr=0 WS=128
2	0.000051354	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=334929 TSecr=334929
3	0.000065900	10.20.30.30	10.20.30.33	HTTP	141	GET / HTTP/1.1
4	0.001580266	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=18 Win=29312 Len=0 TSval=334930 TSecr=334930
5	0.001903154	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=56 Win=29312 Len=0 TSval=334930 TSecr=334930
6	0.002339991	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=93 Win=29312 Len=0 TSval=334930 TSecr=334930
7	0.002827632	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=133 Win=29312 Len=0 TSval=334930 TSecr=334930
8	0.002972824	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=154 Win=29312 Len=0 TSval=334930 TSecr=334930
9	0.003157468	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=156 Win=29312 Len=0 TSval=334930 TSecr=334930
10	0.003329266	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [ACK] Seq=76 Ack=1002 Win=30976 Len=0 TSval=334930 TSecr=334930
11	0.007802360	10.20.30.30	10.20.30.33	TCP	66	36658 → 80 [FIN, ACK] Seq=76 Ack=1003 Win=30976 Len=0 TSval=334931 TSecr=334930
12	2.021042936	10.20.30.30	10.20.30.33	TCP	74	36660 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=335434 TSecr=0 WS=128
13	2.021091189	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=335435 TSecr=335434
14	2.021515565	10.20.30.30	10.20.30.33	HTTP	141	GET / HTTP/1.1
15	2.021541531	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=18 Win=29312 Len=0 TSval=335435 TSecr=335435
16	2.021545469	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=56 Win=29312 Len=0 TSval=335435 TSecr=335435
17	2.021548851	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=93 Win=29312 Len=0 TSval=335435 TSecr=335435
18	2.021552186	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=133 Win=29312 Len=0 TSval=335435 TSecr=335435
19	2.021555533	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=154 Win=29312 Len=0 TSval=335435 TSecr=335435
20	2.021558926	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=156 Win=29312 Len=0 TSval=335435 TSecr=335435
21	2.021562228	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [ACK] Seq=76 Ack=1002 Win=30976 Len=0 TSval=335435 TSecr=335435
22	2.022814708	10.20.30.30	10.20.30.33	TCP	66	36660 → 80 [FIN, ACK] Seq=76 Ack=1003 Win=30976 Len=0 TSval=335435 TSecr=335435
23	4.052492985	10.20.30.30	10.20.30.33	TCP	74	36662 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=335942 TSecr=0 WS=128
24	4.052531714	10.20.30.30	10.20.30.33	TCP	66	36662 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=335942 TSecr=335942
25	4.053234264	10.20.30.30	10.20.30.33	HTTP	141	GET / HTTP/1.1
26	4.054468015	10.20.30.30	10.20.30.33	TCP	66	36662 → 80 [ACK] Seq=76 Ack=18 Win=29312 Len=0 TSval=335943 TSecr=335943
27	4.054497787	10.20.30.30	10.20.30.33	TCP	66	36662 → 80 [ACK] Seq=76 Ack=56 Win=29312 Len=0 TSval=335943 TSecr=335943

Figure 6.22.: Infected traffic collected at the decryption gateway

7. Conclusion and Outlook

Through the course development of this autonomous security response orchestration system, it was seen that by pursuing a systematic method of detection and mitigation, it was possible to design and implement an action plan for a data ex-filtration kind of malicious activity. As mentioned in chapter 4, the machine learning SVM classifier algorithm can be trained to handle traffic prediction which thereby performs real time classification of incoming traffic into 'good' and 'malicious' depending on the volume of traffic the network encounters. In chapter 3, the mitigation use cases for expected attack scenarios- traffic cloning and encryption; VM live migration with traffic cloning and encryption were defined which also further outlined the most significant network topology features that were drawn to address these use cases. Following these design features, as presented in chapter 5 the autonomous security response orchestration was built as automated playbooks to perform mitigation. This followed approach proves to be capable of detection that further triggers the mitigation plan as recorded in chapter 6. Following the results obtained, the following conclusion can be made. In the detection phase, the traffic cases tested with sample features where minimum or no overlapping is noted records the best performance while the traffic cases with maximum overlapping of sample features records lower performance. Also, the overall detection rate and accuracy of the system is recorded $\approx$ 97% and $\approx$ 87%, respectively. In the mitigation phase, no human interaction was required to trigger the mitigation plan thereby automating the security function and reducing response time to an attack. Also, the transfer of the infected traffic/node to a safe node without providing knowledge of the mitigation to the attacker was achieved thereby accommodating for any further analysis. Following the development of this system, there were a few improvements and scope for future development that are discussed in the following section 7.1

7.0.1. Outlook

The implemented security response orchestration was developed for a data ex-filtration attack [**5**]. The detection and mitigation plan can further be developed and improvised to test for different attack types such as phishing, denial-of-service attack and so on.
As part of the detection phase, volume based traffic prediction was considered taking into account the received packets and received bytes only at the incoming victim node that was retrieved using switch statistics with the help of the SDN-controller. In future, while working on systems involving multiple nodes, considering that before hitting the victim node packets travel over additional switch ports the switch statistics of precursor nodes, transferred packets and transferred bytes can be retrieved to detect any possible packet loss or additional overhead.

The random customized traffic generated to train and test the SVM classifier operated with a linear data set using a linear kernel and a regularization parameter of 0.1 as presented in chapter 4. In future, a more comprehensive dataset with additional features to test and evaluate detection can be proposed. With the increasing number of sample features of the dataset the complexity naturally increases but also data of a non-linear nature can be tested for non-linear kernels, gamma and regularization parameters.
The Phantom platform incorporates variety of cyber security applications and its playbooks delve with numerous built-in functions. While some of the functions were utilized while building the mitigation plan, several unexploited features can be considered for future work, specifically for analysis of the affected traffic or affected VM on the safe node at the decryption gateway. The advance analysis was beyond the scope of this thesis however malicious traffic investigating techniques such as deep packet inspection, sandbox or quarantine can be provisioned as future work.
During the construction of playbooks, the ipinfo.io app was used to retrieve geo-location details of the possible attack inflictor using the URL *http://ipinfo.io* as mentioned in chapter 5. While the application was configured only for demonstration purposes, the utility can be made more robust in cases of other attack such as an IP Spoof attack. Similarly, the honeypot as part of the playbook for Use case 2 was executed only for demonstration purposes of this security function and can be made further robust for complex scenarios where the attack tries to surpass the login screen prompt.

A. General Addenda

A.1. Soft-Margin SVM: Tolerance variable

In equation (11) from chapter 4, we have:

$$\hat{y}_n = (w^T.x_n + b) \tag{11}$$

In order to mathematically handle correct predictions by the SVM model, the prediction- $\hat{y}_n$ and the actual prediction- y_n are introduced as a restriction [57].

$$y_n\hat{y}_n \geq 1 \Rightarrow y_n(w^T.x_n + b) \geq 1 \tag{18}$$

For support vector points lying on the margin a value of ε=0 and for misclassified points $\varepsilon \geq 1$ is expected. By introducing the tolerance factor as a parameter along with the restriction from equation (18), we have:

$$y_n(w^T.x_n + b) \geq 1 - \varepsilon_n \tag{19}$$

$$min(\frac{1}{2}w^T.w + C\sum_{n=1}^{N}\varepsilon_n) \tag{20}$$

The aim is to minimize ε and hence as the new variable C increases the overall misclassification represented by the ε values will be smaller ensuring a stricter margin. Introducing the Lagrange multiplier μ to the primal problem in equation(13), we arrive at:

$$L = \frac{1}{2}w^T.w + C\sum_{n=1}^{N}\varepsilon_n - \sum_{n=1}^{N}\alpha_n[y_n(w^T.x_n + b) - 1 + \varepsilon_n] - \sum_{n=1}^{N}\mu_n\varepsilon_n \tag{21}$$

Differentiating equation (21), we have:

$$\frac{\partial L}{\partial w} = w - \sum_{n=1}^{N}\alpha_n y_n x_n = 0 \tag{22}$$

$$\frac{\partial L}{\partial b} = -\sum_{n=1}^{N}\alpha_n y_n = 0 \tag{23}$$

$$\frac{\partial L}{\partial \varepsilon_n} = C - \alpha_n - \mu_n = 0 \tag{24}$$

Thus, the above equations now become:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{25a}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \tag{25b}$$

$$\alpha_n = C - \mu_n \Rightarrow 0 \geq \alpha_n \geq C \tag{26}$$

The dual problem from Chapter 4, equation (17) with the addition of the tolerance factor now becomes:

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n K(x_m, x_n) \tag{27a}$$

under the conditions,

$$0 \leq \alpha_n \leq C; \tag{27b}$$

$$\sum_{n=1}^{N} \alpha_n . y_n = 0 \tag{27c}$$

A.2. Soft-Margin SVM: Implementation in CVXOPT

From Chapter 4, equation (16) we have,

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n x_m x_n \tag{16}$$

where a function of α can be denoted as,

$$f(\bar{\alpha}) = \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n x_m x_n \tag{28}$$

The convex optimization problems[58] are represented as below and the dual problem has to be modified such that it can be parsed by CVXOPT.

$$\begin{aligned} &minimize \ \frac{1}{2} x^T P x + q^T x \\ &subject\ to\ Gx \leq h \\ &Ax = b \end{aligned} \tag{29}$$

- To deduce P:
  Considering similar terms from the above equations, P needs to be expressed as a matrix [57]. The values of elements for a P_{3x3} matrix can be deduced as,

$$\begin{aligned} P_{11}\alpha_1^2 &\Rightarrow P_{11} = (y_1^2 K(x_1, x_1) \\ P_{22}\alpha_2^2 &\Rightarrow P_{22} = (y_2^2 K(x_2, x_2) \\ P_{33}\alpha_3^2 &\Rightarrow P_{33} = (y_3^2 K(x_3, x_3) \\ (P_{12} + P_{21})\alpha_1\alpha_2 &\Rightarrow P_{12} = P_{21} = (y_1 y_2) * K(x_1, x_2) \\ (P_{13} + P_{31})\alpha_1\alpha_3 &\Rightarrow P_{13} = P_{31} = (y_1 y_3) * K(x_1, x_3) \\ (P_{23} + P_{32})\alpha_2\alpha_3 &\Rightarrow P_{23} = P_{32} = (y_2 y_3) * K(x_3, x_2) \end{aligned}$$

  Resulting in,

$$P_{NxN} = (y_T y_T) * K(x_m, x_n) \tag{30}$$

  where K is the gram matrix of calculated dot products of kernel values over the entire dataset.

- To deduce q^T:
  The dual problem needs to be converted to a minimization objective. Hence in equation (16), the inverse sign is to be considered and thus can further be expressed as:

$$-\sum_{i=1}^{n} \alpha_i = q^T \alpha \qquad where \;\; q_{nX1} = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \tag{31}$$

  Together with terms P and q^T, we have first condition of convex optimization problem as:

$$L(min) = \frac{1}{2}\alpha^T P \alpha - q^T \alpha$$

- To deduce A and b:
  From equation(27c) we have,

$$\sum_{m=1}^{N} \alpha_n . y_n = 0 \tag{27c}$$

  Considering $A = [y_1, y_2 .... y_n]$ and b = 0,

$$\sum_{i=1}^{n} \alpha_i \begin{bmatrix} y_1 \\ y_1 \\ \vdots \\ yn \end{bmatrix}^T = 0 \tag{32}$$

  Thus, second condition of convex optimization problem, Ax = b is deduced.

- To deduce G and h:
  From equation (27b) we have,

$$0 \leq \alpha_n \leq C; \tag{27b}$$

  such that,

$$-\alpha_n \leq 0$$

  Resulting in,

$$\begin{bmatrix} -1 & 0 & 0 \cdots & 0 \\ 0 & -1 & 0 \cdots & 0 \\ 0 & 0 & -1 \cdots & 0 \\ \vdots & & & \\ 0 & 0 & 0 \cdots & -1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_n \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{33}$$

  Thus, third condition of convex optimization problem, $Gx \leq h$ is deduced.

A.3. Soft-Margin SVM: Python QP solvers in CVXOPT

The optimal parameters of the convex optimization problem needs to be expressed as matrices and this conversion is presented in Appendix B. After deducing P, q, G, h, A, b parameters the QP problem is solved via the solvers.qp() function from the *CVXOPT* library. The Python implementation is as presented below.

```
# Gram matrix
K = np.zeros((n_samples, n_samples))
for i in range(n_samples):
  for j in range(n_samples):
    K[i,j] = self.kernel(X[i], X[j])

P = cvxopt.matrix(np.outer(y,y) * K)
q = cvxopt.matrix(np.ones(n_samples) * -1)
A = cvxopt.matrix(y, (1,n_samples))
b = cvxopt.matrix(0.0)

if self.C is None:
  G = cvxopt.matrix(np.diag(np.ones(n_samples) * -1))
  h = cvxopt.matrix(np.zeros(n_samples))
else:
  tmp1 = np.diag(np.ones(n_samples) * -1)
  tmp2 = np.identity(n_samples)
  G = cvxopt.matrix(np.vstack((tmp1, tmp2)))
  tmp1 = np.zeros(n_samples)
  tmp2 = np.ones(n_samples) * self.C
  h = cvxopt.matrix(np.hstack((tmp1, tmp2)))

# solve QP problem
solution = cvxopt.solvers.qp(P, q, G, h, A, b)

# Lagrange multipliers
a = np.ravel(solution['x'])
```

Figure A.1.: Optimization problem implementation in Python

A.4. Performance Evaluation: Test D, E

Test D

Test D	Packet Interval		Number of Packets		Training	Test
	Good	Malicious	Good	Malicious	Good + Malicious	Good + Malicious
Case 1	500-10000	500-1000	20-100	100-300	5600	1778
Case 2	500-10000	500-1000	20-100	20-300	5600	1778
Case 3	500-10000	500-3000	20-100	100-300	5600	1778
Case 4	500-10000	500-3000	20-100	20-300	5600	1778

Figure A.2.: Sample features and number of Training+Test samples - Test D

1. Confusion Matrix

$$CM = \begin{bmatrix} TrueNegatives(TN) & FalsePositives(FP) \\ FalseNegatives(FN) & TruePositives(TP) \end{bmatrix}$$

		Predicted	
	Actual	Normal	Attack
Case 1	Normal	782	107
	Attack	1	888
Case 2	Normal	764	125
	Attack	198	691
Case 3	Normal	863	26
	Attack	1	888
Case 4	Normal	101	788
	Attack	0	889

Figure A.3.: Confusion Matrix - Test D

2. Evaluation Metrics:
   The measured and calculated metrics values from the confusion matrix for Test D are as below.

Test D	Precision	Accuracy	Sensitivity	Specificity	FPR	FNR
Case 1	89.24	93.92	99.88	87.96	12.03	0.0011
Case 2	84.83	81.83	77.72	85.93	14.06	0.222
Case 3	97.15	98.48	99.88	97.07	2.92	0.0011
Case 4	53.01	55.68	100.0	11.36	88.63	0

Figure A.4.: Recorded Values (in %) for evaluation metrics - Test D

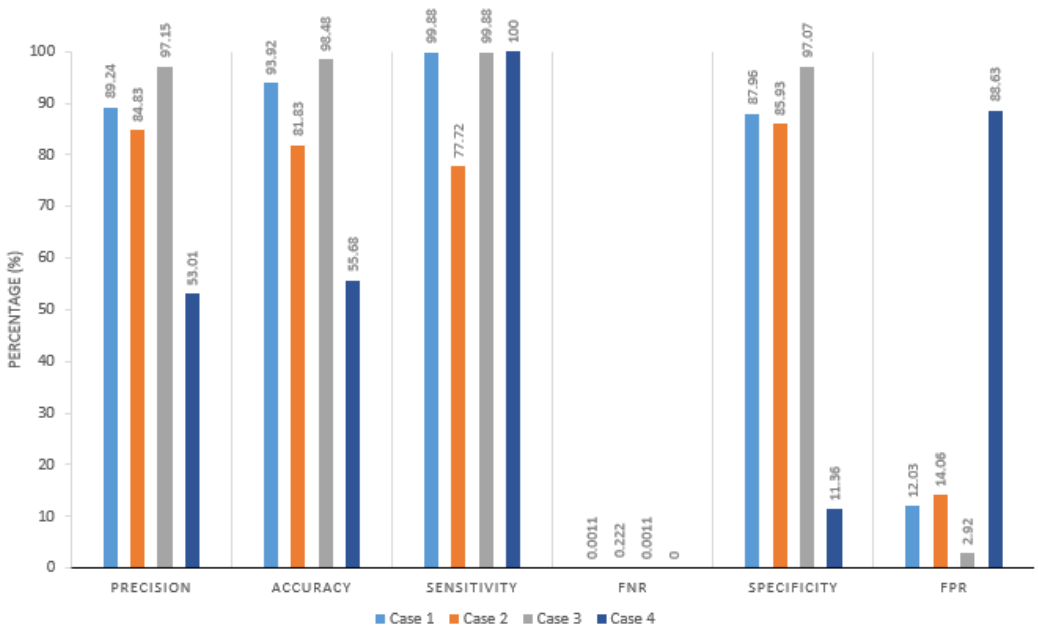

Figure A.5.: Performance - Test D

Test E

Test E	Packet Interval		Number of Packets		Training	Test
	Good	Malicious	Good	Malicious	Good + Malicious	Good + Malicious
Case 1	500-10000	500-1000	20-100	100-300	5400	1938
Case 2	500-10000	500-1000	20-100	20-300	5400	1938
Case 3	500-10000	500-3000	20-100	100-300	5400	1938
Case 4	500-10000	500-3000	20-100	20-300	5400	1938

Figure A.6.: Sample features and number of Training+Test samples - Test E

1. Confusion Matrix

$$CM = \begin{bmatrix} TrueNegatives(TN) & FalsePositives(FP) \\ FalseNegatives(FN) & TruePositives(TP) \end{bmatrix}$$

		Predicted	
	Actual	Normal	Attack
Case 1	Normal	857	112
	Attack	1	968
Case 2	Normal	833	136
	Attack	198	771
Case 3	Normal	940	29
	Attack	1	968
Case 4	Normal	109	860
	Attack	0	969

Figure A.7.: Confusion Matrix - Test E

2. Evaluation Metrics:
   The measured and calculated metrics values from the confusion matrix for Test E are as below.

Test E	Precision	Accuracy	Sensitivity	Specificity	FPR	FNR
Case 1	89.62	94.16	99.89	88.44	11.55	0.001
Case 2	85.00	82.76	79.56	85.96	14.03	20.43
Case 3	97.09	98.45	99.89	97.00	2.99	0.001
Case 4	52.97	55.62	100.0	11.24	88.75	0

Figure A.8.: Recorded Values (in %) for evaluation metrics - Test E

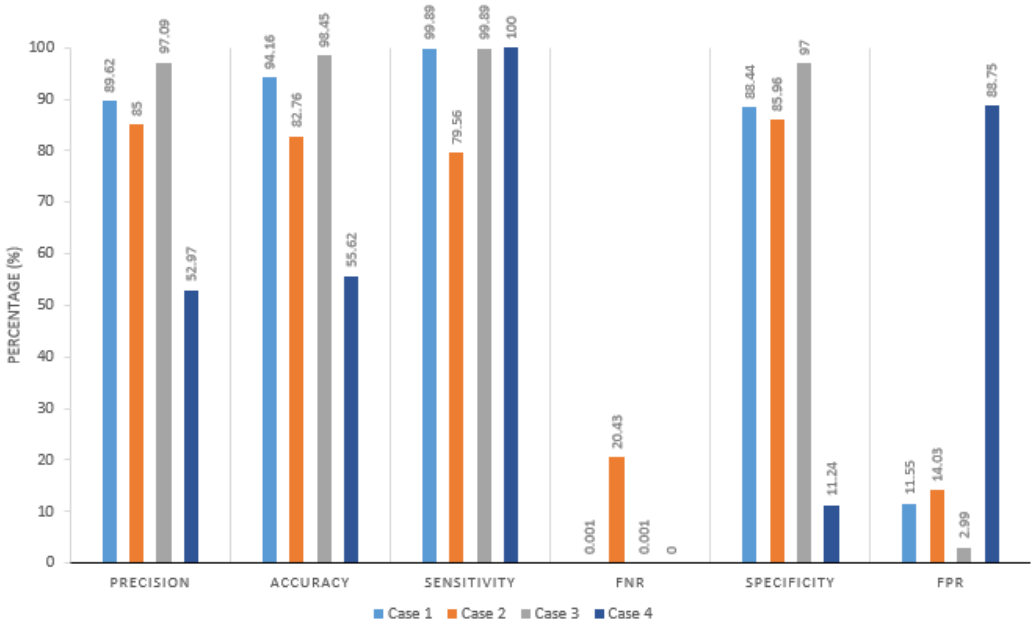

Figure A.9.: Performance - Test E

A.5. Scripts used in detection and mitigation

Subsequently, switch statistics are fetched from the SDN Controller using APIs and the difference between fetches of received packets and received bytes are documented as the dataset. Further, the SVM classifier also performs real time classification as 'good' or 'bad' traffic behavior. When a bad traffic packet is detected, the ML module passes a control signal to the SDN controller which then triggers the playbooks from the security orchestrator.

```
    delta_rx_packets = 0
    delta_rx_bytes = 0
    if rx_packets <= int(data['rx_packets']):
        delta_rx_packets = int(data['rx_packets'])-rx_packets
        rx_packets = int(data['rx_packets'])
    else:
        rx_packets = int(data['rx_packets'])
    if rx_bytes <= int(data['rx_bytes']):
        delta_rx_bytes = int(data['rx_bytes'])-rx_bytes
        rx_bytes = int(data['rx_bytes'])
    else:
        rx_bytes = int(data['rx_bytes'])

    print 'Delta RX_Packets:  ',delta_rx_packets
    print 'Delta RX_Bytes: ',delta_rx_bytes
    print '\n'

    # SVM Checking
    trigger = 0
    if ML.predict([[delta_rx_bytes/100.0, delta_rx_packets/10.0]])[0] == 1.0:
        # Point is in Safezone
        print('Kind is good traffic behavior')
    else:
        print('Kind is bad traffic behavior')
        if trigger == 0:
            subprocess.call(["bash","trigger_playbook.sh"])
            trigger = 1
except:
    print 'StatisticCollector | There is an error... Exited on switch ',self.dpid
    #file_save.close()
    break
time.sleep(t_observation)
```

Figure A.10.: SVM real time classification

```
in_br1=sarnet4
in_br2=sarnet5

enc_port=ens5
dec_port=ens5

port_req1=1
port_req2=2
port_req3=2
port_req4=1

enc_ip1=192.178.0.3
dec_ip1=192.178.0.8

sw4=10.20.5.46
sw5=10.20.5.2

#Encryption g/w
sudo ovs-vsctl --db=tcp:$sw4:6640 add-port $in_br1 $enc_port -- set Interface $enc_port
ofport_request=$port_req2 type=ipsec_gre options:remote_ip=$dec_ip1
options:local_ip=$enc_ip1 options:psk=secret

curl -X POST -d '{"dpid": 0000000000000002, "priority": 99, "flags": 1, "match":{"in_port": 1},
"actions":[ { "type":"OUTPUT", "port": 2 } ] }' http://10.20.5.3:8080/stats/flowentry/add

#Decryption g/w
sudo ovs-vsctl --db=tcp:$sw5:6640 add-port $in_br2 $dec_port -- set Interface $dec_port
ofport_request=$port_req4 type=ipsec_gre options:remote_ip=$enc_ip1
options:local_ip=$dec_ip1 options:psk=secret

curl -X POST -d '{"dpid": 0000000000000003, "priority": 99, "flags":1, "match":{"in_port": 1},
"actions":[ { "type":"OUTPUT", "port": 2 } ] }' http://10.20.5.3:8080/stats/flowentry/add
```

Figure A.11.: Setting up encryption tunneling over a public unencrypted network

Bibliography

[1] Vincent Farhat, Bridget McCarthy and Richard Raysman. *Cyber Attacks: Prevention and Proactive Responses*. Practical Law Publishing Limited and Practical law Company, 2011, p. 11.

[2] Junaidu Bello Marshall and Mua'zu Abdullahi Saulawa. *CYBER-ATTACKS: THE LEGAL RESPONSE*. International Journal of International Law: ISSN: 2394-2622 (Volume 1 Issue 2), p. 3.

[3] United Nations. *Conference on Trade and Development*. URL: https://unctad.org/en/Pages/DTL/STI_and_ICTs/ICT4D-Legislation/eCom-Cybercrime-Laws.aspx.

[4] Aman Singh and Madhusudan Singh. "An Empirical Study on Automotive Cyber Attacks". In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), IEEE, 2018, p. 11.

[5] Cisco. *Common types of cyber attacks*. URL: https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html.

[6] Centre for Strategic International Studies. *Significant cyber incidents*. URL: https://www.csis.org/programs/cybersecurity-and-governance/technology-policy-program/other-projects-cybersecurity.

[7] Tavish Vaidya. *Survey and Analysis of Major Cyberattacks*. URL: https://security.cs.georgetown.edu/~tavish/cyberattacks_report.pdf.

[8] Emily Tamkin, Foreign Policy. *10 years After the Landmark Attack on Estonia, Is the World better prepared for Cyber Threats?* URL: https://foreignpolicy.com/2017/04/27/10-years-after-the-landmark-attack-on-estonia-is-the-world-better-prepared-for-cyber-threats/.

[9] Centre for Strategic International Studies. *Significant Cyber Incidents since 2006*. URL: https://csis-prod.s3.amazonaws.com/s3fs-public/190211_Significant_Cyber_Events_List.pdf.

[10] Jonathan Fildes, BBC. *Timeline: How Stuxnet attacked a nuclear plant*. URL: https://www.bbc.com/timelines/zc6fbk7.

[11] Jon Lockett, The Sun. *CYBER WARFARE? Google 'hit by WORST EVER cyberattack' with traffic 'hijacked' and routed through Russia and China in 'war games experiment'*. URL: https://www.thesun.co.uk/news/7726913/google-cyber-attack-traffic-highjacked-russia-china/.

[12] William Smart. *Lessons learned review of the WannaCry Ranswomware Cyber Attack*. Tech. rep. 2018, p. 11.

[13] BBC News. *Ransomware cyber-attack: Who has been hardest hit?* URL: https://www.bbc.com/news/world-39919249.

[14] International Telecommunication Union. *Definition of cybersecurity*. URL: https://www.itu.int/en/ITU-T/studygroups/com17/Pages/cybersecurity.aspx.

[15] Karen Scarfone, Dan Benigni and Tim Grance, NIST. *Cyber Security Standards*. URL: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=152153.

[16] European Commission. *State of the Union 2017 – CyberSecurity: Commission scales up EU's response to cyber-attacks*. URL: http://europa.eu/rapid/press-release_IP-17-3193_en.htm.

[17] Dr. Steve Purser. *Best practices in Computer Network Defense: Incident Detection and Response*. Standards for Cyber Security, European Union Agency for Network and Information Security (ENISA). pages 101-104, 2014, p. 11.

[18] European Commission. *European Cloud Strategy 2012*. URL: https://ec.europa.eu/digital-single-market/en/european-cloud-computing-strategy.

[19] TNO. *Cyber Security Robustness*. URL: https://www.tno.nl/en/focus-areas/information-communication-technology/expertise-groups/cyber-security-robustness/.

[20] SARNET. *Security Autonomous Response with programmable NETworks*. URL: https://delaat.net/sarnet/index.html.

[21] Leon Gommans, John Vollbrecht, Betty Gommans-de Bruijn, Cees de Laat. *The Service Provider Group Framework: A framework for arranging trust and power to facilitate authorization of network services*. Future Generation Computer Systems, 2014, p. 11.

[22] Leon Gommans. "Multi-Domain Authorization for e-Infrastructures". PhD thesis. Universiteit van Amsterdam, 2014.

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

[24] Ralph Koning, Ben Graaff, Cees de Laat, Robert Meijer and Paola 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.

[25] Ralph Koning, Ben Graaff, Gleb Polevoy, Robert Meijer, Cees de Laat and Paola 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.

[26] Enio Kaljic, Almir Maric, Pamela Begovic and Mesud 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.

[27] Wolfgang Braun and Michael 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.

[28] Li Li, Wu Chou, Wei Zhou and Min 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.

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

[30] Dmytro Ageyev, Oleg Bondarenko, Walla Alfroukh and Tamara Radivilova. "Provision security in SDN/NFV". In: Feb. 2018, pp. 506–509. DOI: 10.1109/TCSET.2018.8336252.

[31] Marco de Benedictis and Antonio Lioy. "On the establishment of trust in the cloud-based ETSI NFV framework". In: Nov. 2017. DOI: 10.1109/NFV-SDN.2017.8169864.

[32] ETSI ISG. *NFV in ETSI*. URL: https://www.etsi.org/technologies/nfv.

[33] Bernd 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.

[34] Gartner Research. *Gartner Market Guide for Security Orchestration, Automation, and Response Solutions*. [Image; Source ID:389446]. URL: https://www.rapid7.com/info/gartner-market-guide-soar/.

[35] Kingsley A. Ogudo. "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.

[36] Bill Stackpole. *An introduction to IPSec*. Auerbach Publications, Jan. 2007, pp. 2093–2101.

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

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

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

[40] Saeed Makhsous, Anton Gulenko, Odej Kao and Feng Liu. "High available deployment of cloud-based virtualized network functions". In: July 2016, pp. 468–475. DOI: 10.1109/HPCSim.2016.7568372.

[41] OpenStack. *OpenStack service overview*. URL: https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html.

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

[43] Open Networking Foundation. *OpenFlow Switch Specification*.

[44] Ben Pfaff, Justin Pettit, Teemu Koponen et al. "The Design and Implementation of Open vSwitch". In: Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15), Oakland, USA, 2015.

[45] Ryu. *Components of Ryu*. URL: https://ryu.readthedocs.io/en/latest/components.html.

[46] Ryu. *Ryu application API*. URL: https://ryu.readthedocs.io/en/latest/ryu_app_api.html.

[47] Phantom. *Overview: Introduction*. URL: https://my.phantom.us/3.5/docs/automation/overview.

[48] Python. *HTTP servers*. URL: https://docs.python.org/3/library/http.server.html.

[49] Linux manual, man7.org. *IP-NETNS(8)*. URL: http://man7.org/linux/man-pages/man8/ip-netns.8.html.

[50] Mihail Balanici and Stephan Pachnicke. "Machine Learning-Based Traffic Prediction for Optical Switching Resource Allocation in Hybrid Intra-Data Center Networks". In: *2019 Optical Fiber Communications Conference and Exhibition (OFC)*. 2019, pp. 1–3.

[51] Mihail Balanici and Stephan Pachnicke. "Server Traffic Prediction Using Machine Learning for Optical Circuit Switching Scheduling". In: *Photonic Networks; 20th ITG-Symposium*. 2019, pp. 1–3.

[52] Mariette Awad and Rahul Khanna. "Support Vector Machines for Classification". In: Jan. 2015, pp. 39–66. ISBN: 978-1-4302-5989-3. DOI: 10.1007/978-1-4302-5990-9_3.

[53] Davide Anguita, A Boni and S. Ridella. "SVM learning with fixed-point math". In: Aug. 2003, 2072–2076 vol.3. ISBN: 0-7803-7898-9. DOI: 10.1109/IJCNN.2003.1223727.

[54] Madhu Sanjeevi. *Support Vector machine with Math*. URL: https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be.

[55] Theodoros Evgeniou and Massimiliano Pontil. "Support Vector Machines: Theory and Applications". In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[56] Andrew Ng. *Support Vector Machines*. URL: http://cs229.stanford.edu/notes/cs229-notes3.pdf.

[57] Oscar Contreras Carrasco. *Support Vector Machines for Classification*. URL: https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3.

[58] Stephen Boyd and Lieven Vandenberghe. "Convex Optimization". In: 2009, pp. 6–8. ISBN: 978-0-521-83378-3.

[59] Phantom Splunk. *Overview - Playbooks*. URL: https://my.phantom.us/3.5/docs/automation/playbooks.

[60] Phantom Splunk. *Phantom Apps Overview*. URL: https://my.phantom.us/3.5/docs/appdev/connector.

[61] Adithya Mishra. *Metrics to Evaluate your Machine Learning Algorithm*. URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.