ANALISYS OF EFFECTS GENERATED BY CYBERATTACKS ON AN SDN CONTROLLER IN A SYNTHETIC DATA ENVIRONMENT

NICOLAS STEVEN CEDEÑO ROMERO
YOUNG INVESTIGATOR PROGRAM

BOGOTÁ D.C. COLOMBIA
SANTO TOMÁS UNIVERSITY
2024

ANALISYS OF EFFECTS GENERATED BY CYBERATTACKS ON AN SDN CONTROLLER IN A SYNTHETIC DATA ENVIRONMENT

THESIS TO QUALIFY FOR THE TITLE OF TELECOMMUNICATIONS ENGINEER
SUPERVISED BY:
JULIANA ARÉVALO HERRERA

NICOLAS STEVEN CEDEÑO ROMERO
YOUNG INVESTIGATOR PROGRAM

BOGOTÁ D.C. COLOMBIA
SANTO TOMÁS UNIVERSITY
2024

CONTENT TABLE

FIGURE TABLE

Abstract

This project explores the dynamics of a Software-Defined Networking (SDN) environment in response to various cyberattacks, focusing on the impact on network performance and stability. Utilizing a controlled setup with Mininet and ONOS, we established a network topology designed to simulate conditions vulnerable to cyber threats. Through the implementation of ARP topology poisoning, three fake hosts were introduced, compromising the integrity of network communications. Additionally, a Packet-In flood attack was executed, resulting in the failure of the ONOS controller, which underscored the challenges of managing network traffic under attack scenarios. The experiments demonstrated how SDN's centralized control architecture can be exploited, revealing significant vulnerabilities in handling malicious traffic. This research highlights the critical need for enhanced security measures in SDN architectures to mitigate risks associated with evolving cyber threats, providing insights for future developments in network security strategies.

Introduction

The internet grew incredibly fast and these traditional network technologies, especially those in the datacenter, cannot cope with the real-time constant change of a modern agile infrastructure. Existing networks already devour walls with patches to contain new problems, leading to bloated systems with weaker control capabilities. Software-Defined Networking (SDN) has removed many of these issues by separating the control and data planes, providing a model that is more adaptable and agile. Programmability SDN's centralized architecture, which uses software-based controllers and APIs, enables network administrators to define policies, manage the network in real-time and optimize performance. Since it breaks down the network into application, control and data layers, SDN is highly customizable and can scale with ease. Advantages of SDN over traditional networking aside, its centralized nature also means security loopholes; a single-shot target for any network attacks that can bring down the entire network. Therefore, the presence of strategic safety mechanisms in SDN is fundamental to addressing the soft spots and increasing protection of such environments. [1] [2]

In traditional networks, it embeds control logic directly into the network device to give a more distributed management. But with an SDN, the downside there is that a global view of everything brought by the controller can also be a direct conflict against security. It may destabilize the whole network if it is attacked. SDN Adoption: SDN implementation challenges become increased as the use of SDN increases. More significant risks threaten SDN environments from attacks such as DHCP starvation, rogue server intrusions, and Denial of Service (DoS) & Distributed Denial of Service (DDoS). SDN uses foundational protocols like ARP, DHCP, ICMP in tandem with OpenFlow to create custom networks. While this enables compatibility with existing network taps and other networking tools, it also introduces familiar security challenges that bedevil both SDNs and their more conventionally built predecessors. Besides, many of these protocols can be attacked to flood SDN controllers with unnecessary information and hence putting lots of stress on hardware. Further, the SDN switches' flow table could potentially be targeted with DDoS attacks which would force the controller to fill it with unneeded rules. Once the switch flow table is full, no packets are allowed to go through and packets will be dropped, affecting network performance. In SDN, owing to the use of flow entries from the controller, it is harder to defend against DDoS compared to traditional networks. [3][4]

1. FRAMEWORK

This work is part of the research project AI004 of the Telecommunications Engineering faculty, initiated in 2024. The Research Unit of Universidad Santo Tomás, Bogotá sponsored it with the program Young Researcher.

1.1. General Objective

- To analyze and demonstrate the impact of various host-initiated cyberattacks on an SDN-based network using the OpenFlow protocol, focusing on specific vulnerabilities and performance degradation

1.2. Specific Objectives

- To explore cybersecurity aspects and relevant attacks in SDN by reviewing the architecture and applications
- To select and evaluate appropriate tools for configuring an SDN testbed.
- To set up an emulated SDN network environment to test and analyze the impact of cyberattacks on the controller.
- To simulate the effects of at least three attacks in SDN using Python scripts

1.3. Methodology

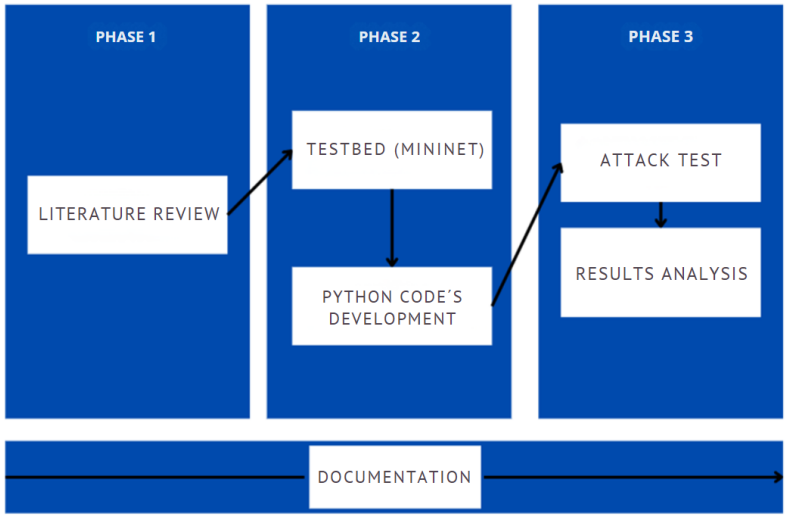

Figure 1 Method Scheme

In figure 1 a four-stage methodology is proposed to characterize the effects on the controller in an SDN network. This methodology is aligned with the FODEIN project, under which this proposal is presented

1.4. Literature Review

A review of studies published in recent years will be conducted across multiple databases to understand the OpenFlow protocol and its use in Software-Defined Networks. The goal is to identify specific attacks and effects directed at the SDN controller.

1.5. Environment Setup

A testing environment will be established using tools such as Mininet. Various scenarios will be configured to simulate normal operation and attack situations. The impact of these scenarios on the SDN network will be evaluated. For this, different Python scripts will be selected and generated, containing algorithms that simulate attacks on the SDN controller.

1.6. Data Collection

The data collection phase is crucial for structuring the collected data into a dataset. This process involves gathering network information from various sources, including traffic logs, event logs, device configurations, and relevant data for analysis. OpenFlow protocol data will also be considered to capture information on the SDN controller's behavior and data flows within the network. Data collection will be carried out carefully to ensure data integrity and quality. The data will be organized and structured coherently into a dataset that will serve as the basis for our subsequent analysis. Additionally, a labeling system will be implemented within the dataset to classify each data entry according to the type of attack or whether it belongs to a benign network state. This labeling will allow for precise categorization of the data and facilitate the identification of patterns and anomalous behaviors during the analysis. This phase is essential to ensure the availability of high-quality data for precise network analysis, helping to identify patterns, trends, and anomalies that may arise in our study.

1.7. Documentation

Phase 4 of the methodology involves the documentation that will be produced throughout the project. This stage is essential to ensure transparency, reproducibility, and understanding of the research process and the results obtained. Various types of documents will be generated, such as technical reports and any other necessary materials to document the procedures, methodologies, tools used, as well as the findings and conclusions reached during the project. Detailed documentation not only facilitates understanding and collaboration among the research team members but also enables sharing the acquired knowledge with the academic and professional community. Additionally, this documentation will serve as a reference for future projects and as the foundation for preparing the publication resulting from this research.

2. SDN ARCHITECTURE AND CYBERSECURITY ASPECTS

2.1. Software-defined networking (SDN)

Software-defined networking (SDN) introduces a new architectural paradigm that separates the control plane from the data plane, which consequently can be more programmatically flexible. This shift has transformed network management to ensure smoother operations through agility and efficiency. But with these benefits, considerable security threats come as well. These increased numbers of illegitimate packets are often because of slow per-packet processing time and an improper use model, such that efficient uses have justified the demand for new security mechanisms compatible with SDN. Even more complexity is given by the movement of 5G network users, especially Internet of Things (IoT) devices. [5] [6]

The difficulties have been addressed with innovative solutions such as adapted blockchain and handover authentication. The SDN model consisting of three layers enables centralized & programmatic network provisioning making networks programmable and intelligent. In fact, because SDN networks are centralized this also makes them ideal for DDoS attacks a significant security challenge. The secondary focus should be on mitigation, as without it these types of attacks will continue to bring a network down and restore network resources fully. Though, centralization makes SDN networks such lucrative targets for DDoS attacks a significant threat. It is a practical solution and should be raised to mitigate these attacks to prevent downtime. [7]

2.2. SDN Architecture

Network is divided into two main plane types such as control plane and data Plane. The control plane is an integral part of SDN which has a centralized nature and determines how the traffic can be forward to network. It oversees the network's logic, e.g. routing and security policies among other too high-level functions. In contrast, the data plane is distributed and performs real forwarding/processing of packets based on instructions from control-plane. The network separation allows flexibility, scalability and centralized management that make SDN an expedient networking environment process. Figure 2 shows the architecture of SDN.

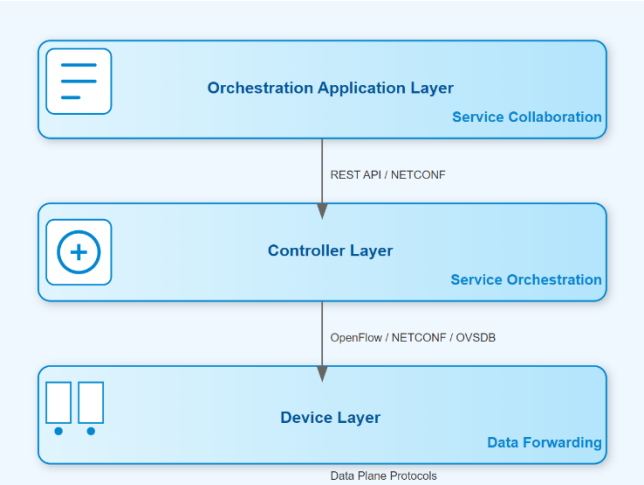

Figure 2 SDN architecture

Figure 2 is the Layered Architecture of SDN, the Orchestration Application Layer, which sits at the top, uses REST APIs or NETCONF to communicate with the Controller layer, and is responsible for collaboration of services and high-level policy direction. This is where the Controller Layer steps in, acting as the brain of your network that converts these policies into actionable instructions (via protocols such as OpenFlow or NETCONF) to the Device Layer. Finally, Network Devices (Device Layer): which focus only on the data forwarding based on controller signals (like switches) as this is where network operations take place in the most efficient and programmable way.

2.3. Planes

2.3.1. Data Plane

The data plane plays an important role in the forwarding/processing of actual packets through the network. The control plane is centralized and contains all the network policies, calls to make decisions as part of decision-making processes), whereas the data plane simply passes packets around between networking devices according to whatever rules have been provisioned for it. Flow tables detail how packets are processed by the data plane. Flow Table Each flow table contains entries which include match fields and instruction fields. The match fields define which packets the flow applies to and are followed by a list of instruction fields that describe what do with this packet (for example, forward it out some port) When a packet arrives at a switch, the data plane uses these flow table entries to look up how that type of header should be processed.

This will find a matching rule for incoming packets from the forwarding plane, which are created and programmed by control plane to perform packet operations, in case of no flows match found, switch will generate Packet_In packet and send it to control plane for further decision. This interaction helps in managing the packets thoroughly and as per the compliance of network policies. In this context, the division of responsibilities (data plane from control points) promotes greater efficiency in networks and hence better network

performance. The control plane handles network-wide policies and decisions. The data plane's focus is high-speed packet forwarding with low latency. The division enables network operators to manage networks in a scalable and agile manner: the control plane can now make changes at some centralized point of control, instead of configuring each device on its own.

Finally, data plane also plays an important role in that it takes care of the applications (security and QoS) policies which are defined on control path. Some of these tasks include traffic filtering, rate limiting and the enforcement of uniform security protocols across a network. [8]

2.3.1.1. Control Plane

The SDN control plane is the one who handles operational activities and commentary over the community as a result a unified view of entire network. In contrast to traditional networks, where forwarding decisions are made by each device along the packet path individually. The SDN control plane is centralized network management, which means it has a single point of decision-making ability that allows for more concerted and streamlined regulatory processes. It is responsible for defining and enforcing network policies (such as routing, security or Quality of Service [QoS] rules) while ensuring consistency throughout the network infrastructure.

It is the control plane which communicates with data plane through protocols like OpenFlow, telling how packets should be treated and forwarded. This interaction is important for the rules of policies defined by control plane to be enforced to all network devices in a consistent manner. The centralized structure of the control plane enables it to dynamically adjust and operate network configurations (i.e. changes in routing paths, insertions or removals of VLAN tags) responsive to variations in traffic loading patterns and security threats as well as other factors associated with operational needs far more quickly that a distributed system which might not be able respond effectively under high load conditions almost immediately upon detection/having been informed about node malfunctions).

Finally, the control plane of SDN can harvest data from all throughout network recording to provide visibility as well as analytics. With a complete view such as this, it is much easier to monitor how well your network and applications perform so that you can identify issues faster and make better decisions. The control plane also interoperates with network applications through APIs for tailored solutions that can make use of the programmability and sophistication features available in it to enrich end-to-end functionality. [9]

2.3.1.2. Application Plane

The SDN Application plane represents an important part of the SDN architecture, by enabling network policies and rules to be defined here. This plane is used to determine the behavior and services of a network for example traffic regulation, security policies etc.

Network administrators interact with this layer to specify what network operations should do, and that information is transmitted to the Control Plane.

The control plane processes the rules and policies of application plane in SDN architecture. The control plane, as the brain of teacher figure, interprets these instructions handling them into actions to be performed and sends it over a wire or rather something more direct from one's laptop then forward this along another type of transport mechanism to the Data plane for actioned upon. This decoupling allows the application plane to concentrate on policy-setting and respond dynamically without digressing into processing network traffic at a low-level.

This open-source platform that lays under the Application Layer provides a certain amount of flexibility and reduces vendor lock-in at this layer with own-vendor protocols. It allows administrators to create and deploy computer networks functioning as if they had been created on proprietary software running network appliances. This transparency means that network policies can be managed rapidly, adapting to shifting working conditions, security threats and functional needs in real-time. So, its robustness and control coming from the application plane enables a network to behave as controlled by its control plane, in combination with adaptability of policies in this layer. [10]

2.3.2. OpenFlow

As the fact protocol in Software-Defined Networking (SDN), OpenFlow was created to enable separation of the control and data planes, the two planes are highly integrated into network devices like routers and switches and each device generally makes its own decisions about how traffic is managed. OpenFlow disrupts this as an SDN controller is now able to directly manage and program the behavior of the underlying network devices. Using a protocol, OpenFlow, the controller provides it with directions on how to handle traffic that comes in at them. This spectral separation hands administrators more power and freedom about what the network can do.

Since the SDN controller (enabled by OpenFlow) is central in nature, it provides a global and more complete view of the entire topology and state of the network. And since the controller can now see workload, they can make smarter, more efficient decisions about traffic management. For instance, the controller can dynamically change traffic paths, move data around to avoid congestion or enforce policies that will gain optimal bandwidth usage. These capabilities can be particularly useful in large networks with traditional distributed decision-making, which can result in inefficiencies (e.g. hot potato problem), leading to bottlenecks or situations where available bandwidth is underutilized. OpenFlow to enhance on demand flow assignment with real-time flow of traffic, this provides a more responsive network for better performance.

A third major advantage of OpenFlow is its scalability and ease of introduction without large-scale hardware upgrades. The controller, like a hypervisor in computing, therefore,

decouples behavior of the network from underlying protocols and elements running on specific hardware to control the action in software enabling introduction of new protocols or security features or traffic optimization techniques by simply updating the respective software. This flexibility allows the network to scale in response to growing demands, like a requirement of high bandwidth within cloud data centers or even growing number of sensors in Internet of Things (IoT) environments. [11]

2.3.3. SDN Interfaces

In SDN architecture, Northbound Interface and Southbound interface are essential components. They are responsible for communication amongst different layers in the SDN stack, to keep the network up and running while managing it.

- The Northbound Interfaces (NBI) APIs are used for communication from this SDN control plane to the applications/services that are hosted on it. These interfaces provide an abstraction for network application to request services and information from the SDN controller. Developers can take advantage of NBIs to write applications that programmatically control network behavior, optimize performance and enforce security. For example, in network security use cases, NBIs allow for the automatic adaptation of policies based on current data to combat threats (like DDoS attacks). [5][6]

- Southbound Interfaces (SBI) gives pathways that facilitate communication between the Network devices in data plane and SDN control Plane. Together with the controller, SBIs translate the grand commands from controllers into detailed instructions to network devices. A feature allows OpenFlow based Switch-Level Management, importation of protocols such as the SBIs that are essential for SDN controller to control flow table on switches to ensure effective traffic forwarding and policy enforcement. This level of control is necessary for greatness in the fine tuning of network management tasks and resources.

This makes the network environment programmable and efficient by integrating NBIs /SBIs in SDN architecture. Using these interfaces, SDN can help manage the dynamic nature of IoT devices, for example on 5G networks as well secure infrastructure against evolving cyber threats. Utilizing advanced security techniques to manage all your network-related queries, NBIs and SBIs together aid in better serving modern networks with more comprehensive functionality.

2.4. Cybersecurity aspects

As this document is aligned with, cybersecurity is one of the biggest threats that community must take care for this new technology, there are a lot of ways of take care of but one of them is develop cyberattacks with the objective of making a defense against it, taking that approach, it will focus on 3 different cyberattacks, TCP SYN Flooding, Host Injection and Packet in Injection.

2.4.1. TCP Syn Flood

A TCP SYN Flooding attack is a type of denial-of-service (DoS) attack that targets the TCP handshake process. In a standard TCP connection setup, a client initiates the connection by sending a SYN (synchronize) packet to a server. The server responds with a SYN-ACK (synchronize-acknowledge) packet, and the client completes the handshake with an ACK (acknowledge) packet. During a SYN Flood attack, the attacker sends many SYN packets to the server, often with spoofed source addresses. On the figure 3 the server responds with SYN-ACK packets and allocates resources for each incoming request. However, the attacker does not complete the handshake, leaving the server with numerous half-open connections. This exhausts the server's resources, preventing it from processing legitimate connection requests and effectively denying service to valid users. [12]

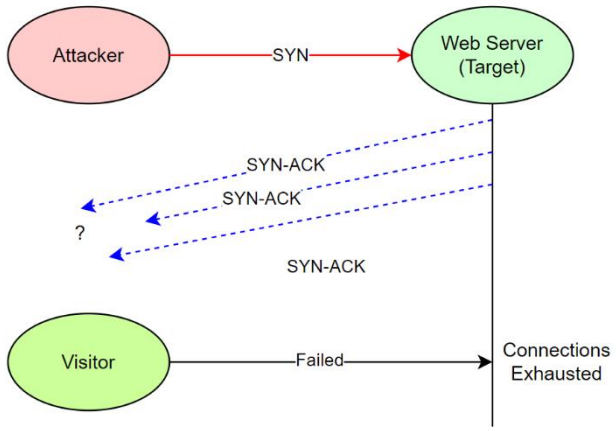

Figure 3 TCP SYN Attack

This attack can significantly impair a server's ability to handle legitimate traffic by overwhelming it with a high volume of incomplete connection requests. As the server continues to allocate resources for each half-open connection, its capacity to manage new, genuine requests diminishes. This can lead to severe disruptions, including degraded performance or complete unavailability of the server for legitimate users. The attack leverages the server's finite resources, exploiting its state management and connection handling mechanisms to create a bottleneck. Effective mitigation strategies often involve detecting unusual traffic patterns, implementing rate-limiting controls, and employing specialized hardware or software solutions to filter and manage incoming SYN requests, ensuring that the server remains available for legitimate users. [5]

The challenge of managing a TCP SYN Flooding attack underscores the broader responsibility of cybersecurity to maintain service availability. In the realm of digital communications, ensuring that systems are both secure and accessible is crucial. While defending against attacks such as SYN Flooding involves implementing robust detection and mitigation strategies, it also raises questions about the balance between security measures and system

availability. Effective cybersecurity must not only prevent unauthorized access and protect data integrity but also ensure that legitimate users can access services without disruption. This balance requires a proactive approach to security that anticipates potential threats and implements resilient infrastructure capable of handling high volumes of traffic. By addressing these challenges, cybersecurity professionals play a pivotal role in maintaining both the integrity and availability of digital services, ensuring that systems remain functional and reliable amidst evolving threats. [13]

2.4.2. Host Injection - Topology Poisoning

A Host Injection attack, particularly in the form of Topology Poisoning, targets the network's topology management system, which is crucial in SDN. In SDN, the network topology is dynamically managed and updated by the controller, which maintains a global view of the network. During a Topology Poisoning attack, an attacker injects false information into the network, misleading the controller about the actual network layout. This false data can include fake host entries, incorrect link states, or bogus network paths. [14] [15]

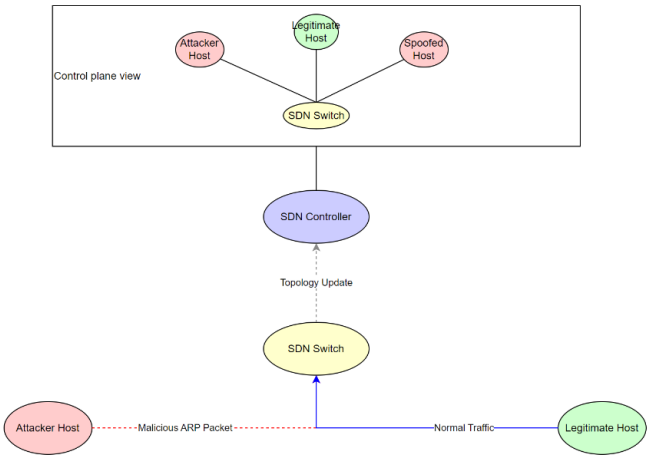

Figure 4 Host injection Attack

Figure 4 shows a topology Poisoning that exploits the dynamic and flexible characteristics of SDNs, which, while beneficial for scalability and efficiency, can also introduce vulnerabilities. For instance, the injection of bogus host entries or fabricated links can overwhelm the controller with false updates, degrading network performance and security. Therefore, robust validation and continuous monitoring of topology updates are crucial in mitigating the risks associated with Topology Poisoning attacks. [14]

2.4.3. Packet Injection

A kind of SDN attack involves sending a large volume of fake data packets to an SDN switch. Attackers can send so many requests to the controller, generating a lot of Packet_In messages for table misses which are not initially installed in flow table. An SDN switch will normally send a Packet_In message to the controller, if it encounters a packet for which

there is no flow rule that matches. A Flow Rule will evaluate the incoming packet and decide what should be done with them. It leads to performance degradation when the controller is flooded with messages, so it is not able to handle real traffic in time.

In figure 5 the packet injection attacks have features both on adapting SDN control plane, and in deteriorating the overall performance of the network. Such attacks can be identified by a great number of malicious data packet flows being sent to the switch that causes directly an increase in Packet_In messages. This can overwhelm the controller and alternatively render the network slow or unusable, since in addition to handling legitimate traffic it now also needs to process 100 control messages per second (of just about anything). This dynamic also has a double whammy, where it degrades not only the performance of the network but also its security due to control plane being very susceptible point for these vulnerabilities.

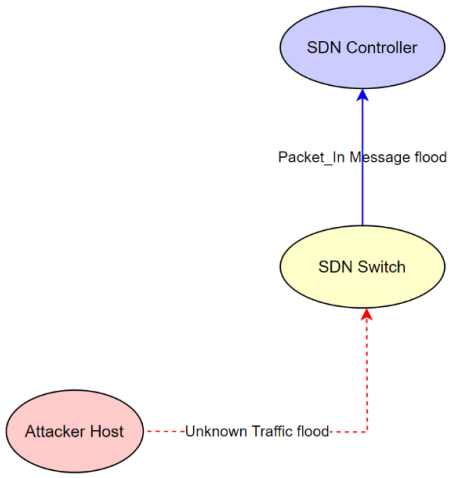

Figure 5 Packet Injection attack

To secure their systems from packet injection attacks, service providers implement many security apparatuses. Rate limiting and deep packet inspection implemented in hardware then assist to filter malicious packets before reaching the controller. An IDS can either be a hardware device, or more commonly as software only, which monitors the incoming traffic and/or system activity for suspicious behavior. Nevertheless, as the attackers are so adaptable that after some time, they will develop new evasive methods and exploit newly discovered SDN vulnerabilities due to constant changes in the evolving SDN environment. [15] [16]

3. AVAILABLE TOOLS TO CONFIGURE A TESTBED FOR SDN

SDN solves this scaling problem by decoupling the control and data planes dramatically changing the way networks are built, operated, and optimized. This paradigm shift provides higher flexibility, scalability and programmability for SDNs to be used as common architecture between multi-environmental resources like. IoT, Cloud Computing & Data Centers, etc. Due to the complexity of SDN, a lot of researchers and practitioners build/test validation through testbeds—just one of such controlled environments that enables flexible simulation, testing, and analysis for SDN networks.

In this study, we focus on finding these available tools to configure a testbed for SDN. As a result, that makes tools like Mininet, OpenvSwitch and SDN controllers such as ONOS or OpenDaylight crucial to let you emulate behavior of the actual network, simulate traffic and test all sorts of configurations and attacks. Here we aim to present an overview of what SDN tools can do, how they operate and their roles in testbed environment where researchers can simulate and experiment new SDN concepts, benchmarks performance studies, and analyze security exploits to design novel counter variety of tests and scenarios. Such worlds help not just build secure SDN testbeds but also pave ground for additional developments in traditional SDN.

3.1. Docker Engine

Docker is a container-based application development platform which enables users to create, deploy and run applications in a container form. In this project, Docker gets to facilitate the independent deployment of ONOS and Mininet, making it easier to manage the network and provision for scalability of the said infrastructure. The separation of these different components makes sure that performance is not compromised when dealing with various tests within these systems, and flexibility is optimized.

3.2. ONOS (Open Network Operating System)

The ONOS application is an appropriate operating system for SDN which aims to manage networks via a central point. It acts as the SDN traffic controller in the topology created using Mininet by directing traffic between switches with hosts. Due to its mini-configurational structure and scalability, it is appropriate for deploying security measures and combating attacks in SDN systems.

3.3. Mininet

Mininet is a network implemented solely for simulation purposes and allows quick and simple creation of virtual topologies. It is also used for network development so that each of the different simulated attacks can be tried out, and their effects monitored. Mininet creates an OpenVSwitch together with the hosts which gives a test bed for the study with

respect to the ever-changing traffic levels. The inbuilt python language API allows the attachment of Mininet to ONOS and the management of topology using the interface.

3.4. Python

Python is a general-purpose high-level programming language. In the project, it is used to build a web server on one of the hosts that can produce all the attacks listed before. It is also used to monitor the performance of the system outright and as part of an attack (to make CPU readings).

3.5. Tshark

Wireshark has a command line packet capture utility called Tshark. It could be used to gather data on traffic behavior before, during and after an attack by executing it as a tool observing traffic in some simulated network. Nothing to enjoy, data collection with Tshark is required to analyze the impact of the attack on the network and determine how well mitigation techniques are working.

Establishing a testbed for Software-Defined Networking (SDN) is a crucial step in advancing both research and practical applications in the field. A well-designed testbed allows for the simulation of diverse network scenarios, the evaluation of performance, and the analysis of network behavior under varying conditions. By providing a controlled and flexible environment, testbeds enable researchers to gain valuable insights into SDN's capabilities and challenges, contributing to the development of more efficient, secure, and scalable network solutions.

4. TESTBED IMPLEMENTATION

A Testbed was built to perform experimental evaluation of how an SDN-based network behaves under each proposed attack. The topology is a switch, with the outer switch controlled by ONOS and bridged through the flow-instruction application: 10 hosts are evenly connected to the switch with a port mirror for managing the operation of the network. Each instance played the role of a node over web and denial-of-service, topology poisoning attack on some of hosts.

The ONOS controller, which ran inside a Docker container and was configured with OpenFlow support, managed the forwarding and routing decisions of each switch. The controller was linked to virtualized switches (OpenVSwitch), and Mininet provided emulation for entire network topology.

The attacks were recorded by Python scripts utilizing Scapy, together with SYN flood and ARP poisoning. For the Packet injection attack a python script was run in the switch building and sending the PacketIn messages simulating the flood. Network traffic was monitored with Tshark, and the system resource usage data were collected using psutil. We used to save this data in CSV files for further analysis.

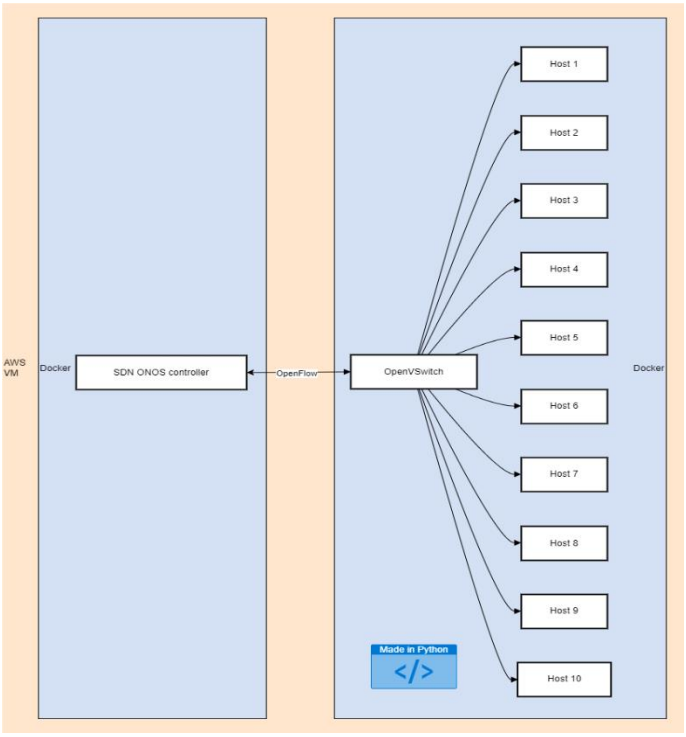

Figure 6 TestBed Diagram

Figure 6 represents the **SDN Testbed** architecture used for the project, with Docker-based deployment for both the SDN ONOS controller and Mininet. The ONOS controller, located in an AWS VM, manages the network via OpenFlow, dynamically updating the flow rules for

the OpenvSwitch (OvS). The OvS forwards both legitimate and malicious traffic, including TCP SYN flood attacks generated by attacker hosts, which aim to overwhelm the web server and trigger excessive Packet-In messages to the controller. Meanwhile, normal traffic is handled by other hosts connected to the switch. This setup demonstrates the interaction between the control and data planes in SDN and how the system responds to both normal and malicious traffic scenarios.

4.1. TCP SYN flood

Figure 7 explains the implementation and testing of SDN using ONOS controller with an OpenFlow connection to a Docker SDN TestBed running within an amazon EC2 instance. The controller links with an OvS, and OvS has another Docker container running in it. This switch is used to split the traffic between several hosts: some of which will generate normal (shown in green) while other hosts are used to simulate a SYN flood attacking sending TCP SYN packets maliciously directed at the web server taken from red aspect. This causes the attacker host to flood the web server (yellow host) with irregular traffic, filling it with hundreds of SYN messages sent simulating the attack.

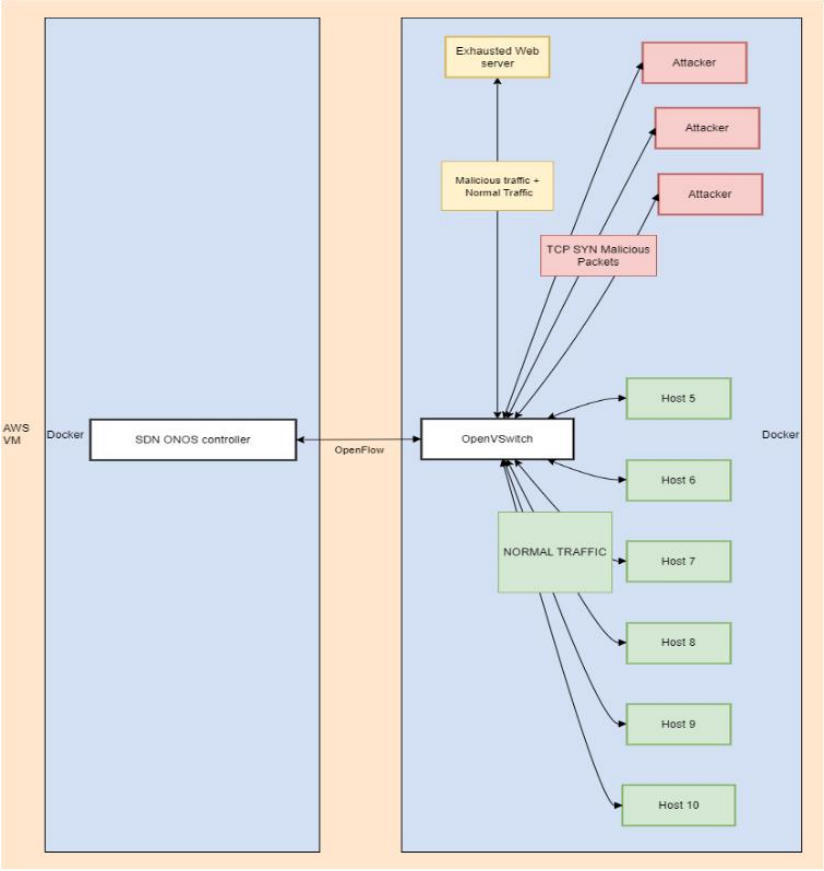

Figure 7 TCP SYN flood Scenario

4.2. Topology Poisoning

Figure 8 illustrates a scenario where an attacker executes a python script to generate fraudulent ARP packets with the intention of corrupting the network topology within the ONOS controller. These spoofed ARP packets are then forwarded by the OvS to the ONOS controller. The controller receives these packets and unintentionally rewrites the topology in believing that some fake hosts (which are generated through the Python script) exist here, victims are legitimate devices in this case. It breaks the real network topology: 10 real hosts are linked to OvS by Docker. In effect, it tricks the ONOS controller into submitting fake topology updates that affect the network integrity.

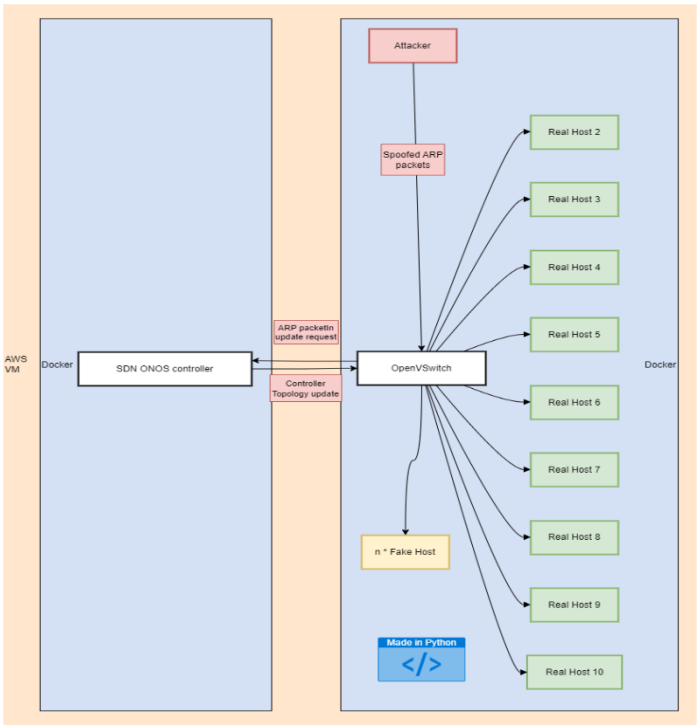

Figure 8 ARP topology Poisoning Scenario

4.3. Packet Injection (PacketIn Flood)

In Figure 9with a classic attack on an SDN infrastructure in which we analyse typical network security scenario. The ONOS controller is under targeted attack. The very low layer of abstraction is the controller, and the attack will flood this controller with OpenFlow Packet_In messages. Well, these messages are being generated using python script with the help of Scapy which is a packet manipulation tool. The objective of this attack is to overload the ONOS controller and the ONOS controller may run out, and finally it will crash. This attack exploits the centralized control in SDN architectures by making the controller function as a single point that controls all the network. Disrupting the controller may make it possible for an attacker to gain unauthorized access, disrupt network operations or cause a denial of

service. Such an attack reveals the need for strong security measures and monitoring tools in SDN-based networks.

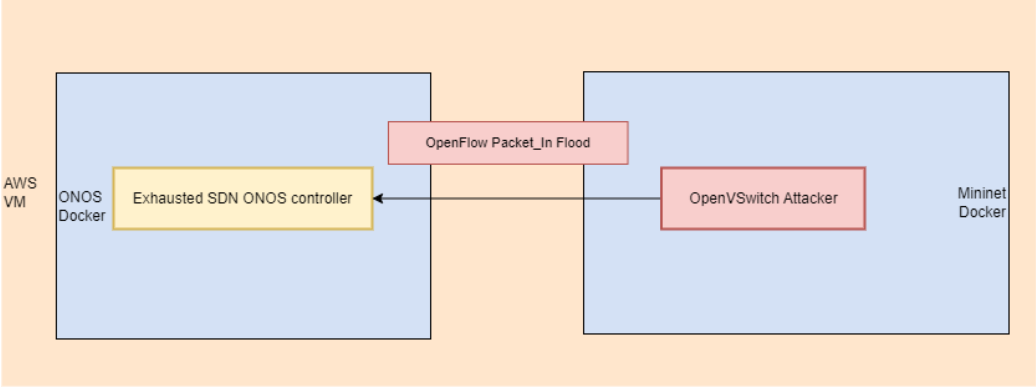

Figure 9 TestBed Scenario PacketIn flood

A testbed is an essential component for every research on software defined networking (SDN) that we can build a set of experimentation. It allows us to create an ideal and controlled environment to imitate real world scenarios, analyze the performance of very realistic network and have a review different configurations or strategies. Using a testbed allows researchers to study the interactions in an SDN, evaluate its robustness to challenges, and investigate possible enhancements. This process is not only going to increase the understanding of SDN systems, but also pave the way for new solutions and next-generation network design & management.

5. EFFECTS OF THE ATTACKS IN THE NETWORK

Assessing the effects of different types of cyberattacks on Network-Function Virtualization (NFV) environments is critical to improving security and resilience for SDN settings. In this section, we investigated the consequences of TCP SYN Flood (flooding one protocol), ARP Injection (injection one packet), and Packet-In Flood (pushing more packets at once) on SDN networks. Such attacks can severely impair network performance by causing delays, exhausting resources and even crashing controllers. This research seeks to analyze the effects of these types of attacks and gain an understanding of the risks that they introduce; knowledge which could be useful for designing strong security mechanism able to mitigate such threats and maintain SDN systems operationality in real world scenarios.

5.1. TCP SYN flood

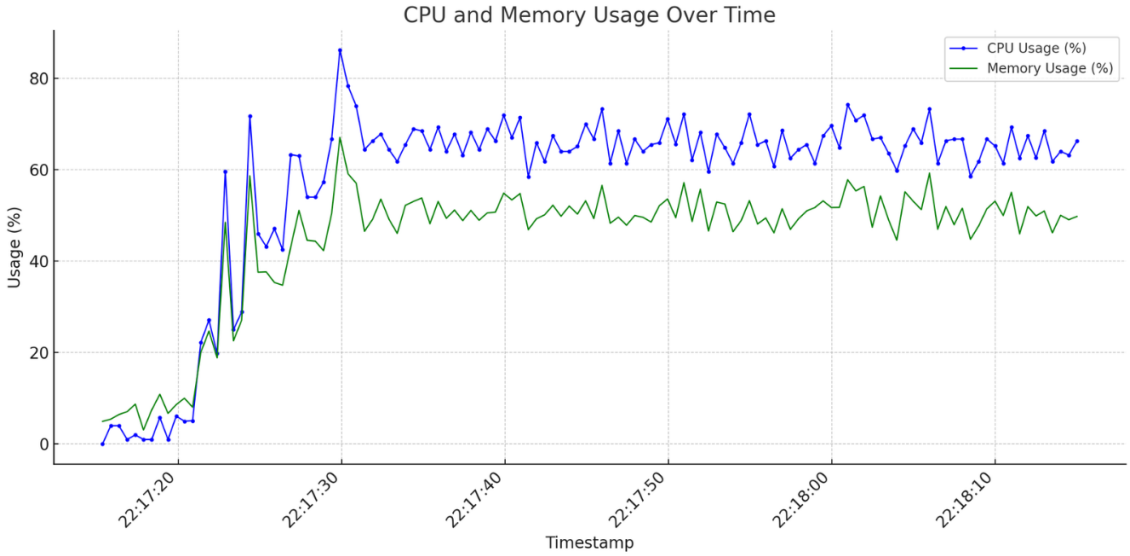

Figure 10 TCP Syn Flood Results

Figure 10 shows the CPU and memory usage over time in response to some activity, when the network attack is being performed. The blue line represents CPU usage as a percentage, while the green line shows memory usage as a percentage. Initially, both CPU and memory usage start around 20%. As time progresses, the CPU usage rapidly increases, spiking to above 80% before stabilizing around 60-70%. Meanwhile, memory usage also rises but to a lesser extent, stabilizing around 40-50%. This pattern indicates a sudden and sustained demand on system resources, which is typical in scenarios involving network attacks or high traffic, where the CPU is particularly impacted, while memory usage also shows a noticeable but more moderate increase.

5.2. PinGen

Following the ARP poisoning attack, the SDN project topology showed considerable changes with three other additional fake hosts. Originally, the network was made up of reputable hosts with uninterrupted connections to allow for effective communication. But after the

attack, the controller was cheated and did not know if they were seeing the real hosts or not (who just fed them with fake network information to lure them into their pit). These malicious actors were able to make incorrect mappings of IP addresses and MAC addresses by misrepresenting themselves, which in turn led to legitimate traffic being re-routed and trapped. In so doing, the manipulation did not only corrupt communication between legitimate hosts but also hugely exposed them to data breaches. The presence of incorrect hosts in the new topology indicates that SDN frameworks are susceptible to ARP Poisoning attacks and there is an urgent necessity for increasing security mechanisms against it. Figures 11 and 12 show the real topology versus the topology after the attack.

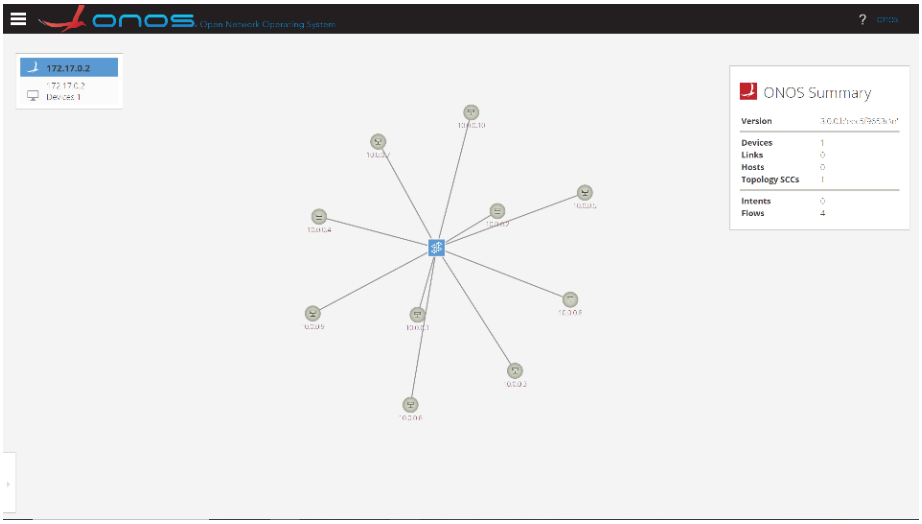

Figure 11 Original topology

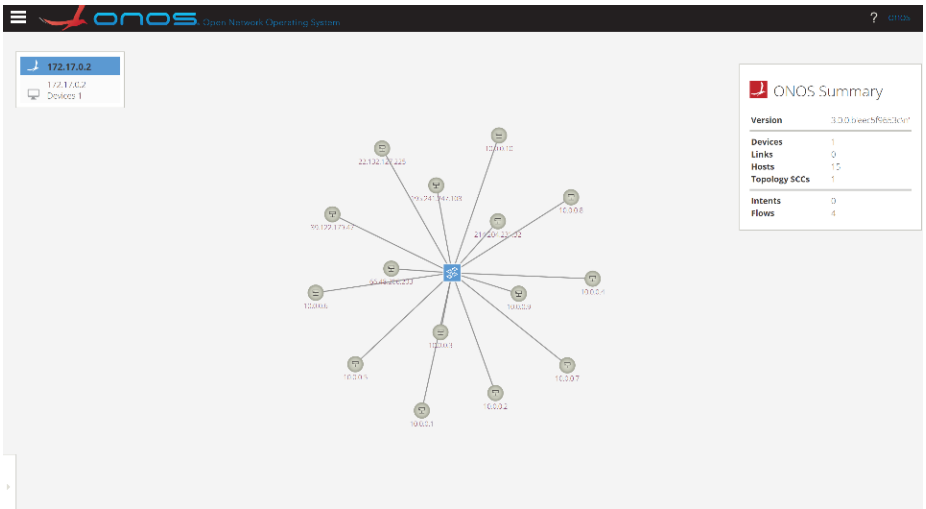

Figure 12 Host injection Done Successfully

5.3. Packet_In flood

The Packet-In flood attack also heavily degenerated the performance of ONOS controller to the extent that web socket connection to the server was shut down. This implies that the controller is unable to cope with so many Packet-In messages generated during the attack.

The ONOS interface, as a result, is incapable of displaying the current network topology or responding to commands in a time efficient manner. This incident underscores the vulnerabilities of SDN architecture, specifically, how the controllers in an SDN network manage traffic requiring the highest rate attacks. In addition to degrading the controller's functioning, this inability to keep in touch and have real time monitoring of what is going on in the network impacts not only the security against all sorts of threats but also the reliability of the network. This emphasizes that strong mitigation strategies and adaptation resources to make SDN controller resilient should be created as soon as possible so that it can survive such attack turning back into its valid state securely and would be efficient enough to keep a stable network structure. Figure 13 shows the effects on the graphic user interface of the controller after the successful attack.

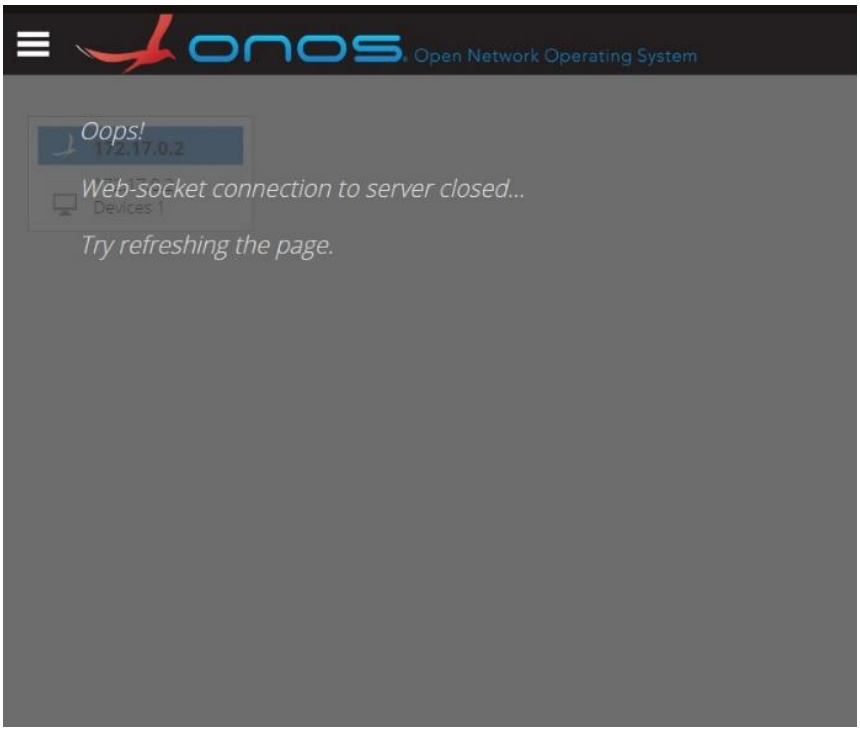

Figure 13 ONOS Controller Exhausted

In figure 13, the ONOS controller interface displays an error message: “Oops! Web-socket connection to server closed... Try refreshing the page." This indicates the disruption of the controller's operations caused by a Packet-In Flood attack. The attack overwhelms the ONOS controller with an excessive number of Packet-In messages, exhausting its computational resources. As a result, the controller becomes unresponsive, and its web socket connection fails, preventing real-time updates from being displayed on the GUI. This figure provides visual evidence of the attack's effectiveness in compromising the SDN controller's functionality.

6. CONCLUSION

This project has provided ambitious research, and a comprehensive study of challenges faced in an SDN environment by examining the behavior associated with several attack scenarios, particularly on performance and stability. A network topology was set up to simulate different kinds of cyber-attacks such as ARP injection, Packet-In flood and a couple others using Mininet and ONOS.

Topology poisoning attack used to manipulate state of network where now three impostor hosts are shown, and this shows the capabilities, ARP attacks only rely on fake address at victim. Furthermore, packet-in flood attack rendered ONOS controller dysfunctional by causing congestion and tampering of traffic as it is seen to be impactable system. The results of this thesis ensure the necessity of robust ingress filtering mechanisms in SDN environments to protect network integrity and service availability.

This research provided a substantial insight on how SDN architectures react to cyberattacks, and the need of strong countermeasures to reduce these threats for building safer and higher resilient networks in the years to come.

7. BIBLIOGRAPHY

[1] L. Mhamdi and M. M. Isa, “Securing SDN: Hybrid autoencoder-random forest for intrusion detection and attack mitigation,” *J. Netw. Comput. Appl.*, vol. 225, p. 103868, May 2024, doi: 10.1016/j.jnca.2024.103868.

[2] M. W. Nadeem, H. G. Goh, Y. Aun, and V. Ponnusamy, “Detecting and Mitigating Botnet Attacks in Software-Defined Networks Using Deep Learning Techniques,” *IEEE Access*, vol. 11, pp. 49153–49171, 2023, doi: 10.1109/ACCESS.2023.3277397.

[3] M. AbdulRaheem *et al.*, “Machine learning assisted snort and zeek in detecting DDoS attacks in software-defined networking,” *Int. J. Inf. Technol.*, vol. 16, no. 3, pp. 1627–1643, Mar. 2023, doi: 10.1007/s41870-023-01469-3.

[4] H. U. Ishtiaq, A. A. Bhutta, and A. N. Mian, “DHCP DoS and starvation attacks on SDN controllers and their mitigation,” *J. Comput. Virol. Hacking Tech.*, vol. 20, no. 1, pp. 15–25, Mar. 2023, doi: 10.1007/s11416-023-00483-0.

[5] C. Gonzalez and S. M. Charfadine, “SDN Controllers and ML-Based Anomaly Detection in Embedded Systems: A Comparative Analysis,” in *Proceedings - 10th International Conference on Wireless Networks and Mobile Communications, WINCOM 2023*, 2023. doi: 10.1109/WINCOM59760.2023.10322912.

[6] I. H. Abdulqadder, D. Zou, and I. T. Aziz, “The DAG blockchain: A secure edge assisted honeypot for attack detection and multi-controller based load balancing in SDN 5G,” *Futur. Gener. Comput. Syst.*, vol. 141, pp. 339–354, Apr. 2023, doi: 10.1016/j.future.2022.11.008.

[7] S. Karnani and H. K. Shakya, “Mitigation strategies for distributed denial of service (DDoS) in SDN: A survey and taxonomy,” *Information Security Journal*, vol. 32, no. 6. Taylor & Francis, pp. 444–468, Nov. 02, 2023. doi: 10.1080/19393555.2022.2111004.

[8] M. Yue, Q. Yan, Z. Lu, and Z. Wu, “CCS: A Cross-Plane Collaboration Strategy to Defend Against LDoS Attacks in SDN,” *IEEE Trans. Netw. Serv. Manag.*, 2024, doi: 10.1109/TNSM.2024.3363490.

[9] D. Tang, Z. Zheng, C. Yin, B. Xiong, Z. Qin, and Q. Yang, “FTODefender: An efficient flow table overflow attacks defending system in SDN,” *Expert Syst. Appl.*, vol. 237, p. 121460, Mar. 2024, doi: 10.1016/j.eswa.2023.121460.

[10] D. Tang, Z. Zheng, K. Li, C. Yin, W. Liang, and J. Zhang, “FTOP: An Efficient Flow Table Overflow Preventing System for Switches in SDN,” *IEEE Trans. Netw. Sci. Eng.*, 2023, doi: 10.1109/TNSE.2023.3297650.

[11] N. Aslam, S. Srivastava, and M. M. Gore, “A Comprehensive Analysis of Machine Learning- and Deep Learning-Based Solutions for DDoS Attack Detection in SDN,” *Arab. J. Sci. Eng.*, vol. 49, no. 3, pp. 3533–3573, Mar. 2024, doi: 10.1007/s13369-023-08075-2.

[12] Toyeer-E-Ferdoush, H. Rahman, and M. Hasan, “A convenient way to mitigate DDoS TCP SYN flood attack,” *J. Discret. Math. Sci. Cryptogr.*, vol. 25, no. 7, pp. 2069–2077, Oct. 2022, doi: 10.1080/09720529.2022.2133246.

[13] H. Qian and L. Cai, “Improved K-means-based solution for detecting DDoS attacks in SDN,” *Phys. Commun.*, vol. 64, p. 102318, Jun. 2024, doi:

10.1016/j.phycom.2024.102318.
[14] Y. Gao and M. Xu, “Defense Against Software-Defined Network Topology Poisoning Attacks,” *Tsinghua Sci. Technol.*, vol. 28, no. 1, pp. 39–46, 2023, doi: 10.26599/TST.2021.9010077.
[15] S. Sunil Gowda and R. B. Dayananda, “Detection And Prevention of ARP Attack in Software Defined Networks,” in *2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT 2023*, 2023. doi: 10.1109/ICCCNT56998.2023.10307123.
[16] A. T. Phu *et al.*, “Defending SDN against packet injection attacks using deep learning,” *Comput. Networks*, vol. 234, p. 109935, Oct. 2023, doi: 10.1016/j.comnet.2023.109935.