Securing SDN: Implementing ETLS and Dynamic Flow Management in OpenFlow

MSc Research Project
Msc in Cybersecurity

Safal Harshan Vadassery
Student ID: X22243909

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Safal Harshan Vadassery……. …		
Student ID:	X22243909…………………………………………………………………………………………………….…….		
Programme:	Msc in Cybersecurity………………………	**Year:**	2023-24…………………………..
Module:	Msc Research Project………………………………………………………………………….………		
Supervisor:	Mark Monaghan………………………………………………………………………………….………		
Submission Due Date:	12 August……………………………………………………………………………….………		
Project Title:	Securing Software-Defined Networks: Implementing Enhanced Transport Layer Security and Dynamic Flow Management in Openflow ……………………………………………………………………………………………………….………		
Word Count:	7152……………………… **Page Count**……19……………………………………….……..		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Safal Harshan Vadassery………………………………………………………

Date: 12 August……………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Contents

Securing Software-Defined Networks: Implementing Enhanced Transport Layer Security and Dynamic Flow Management in Openflow

Safal Harshan Vadassery
X22243909

ABSTRACT

The following paper discusses implementation and evaluation of encrypted OpenFlow communication in an SDN environment. Since SDN is continuously evolving, it is very necessary to secure communication between the control plane and data plane. I am using GNS3, a very popular network simulation tool, to create a very close-to-real test environment. Open vSwitch works under this setting as a virtual switch infrastructure, while Ryu controller serves as the SDN controller. My main goal is to configure and secure the OpenFlow messages exchanged between these components using TLS encryption. It is a standard for securing data transmission over networks. The whole process—from setting up GNS3, incorporating OVS instances, step-by-step configuring OVS bridges and ports, to ensuring the implemented topology works within GNS3—is well explained in the report. We then install and configure the Ryu controller itself, including developing a simple Ryu application that will be in charge of every OpenFlow switch. One of the most critical aspects of this setup is how to generate and manage SSL certificates. This is how to create a Certificate Authority, generate certificates for the Ryu controller and the OVS instances, and then sign these certificates to create a trusted link. As a final exercise in the series, capture and analyze network traffic to check encrypted communication for proof of successful TLS encryption implementation in SDN communication. These results underline the feasibility and high security gain for securing SDN communication and give valuable insights into the practical implementation of secure SDN environments.

Keywords: GNS3, OpenFlow, Software Defined Network (SDN), OVSwitches, TLS/SSL, Ryu Controller

1 INTRODUCTION

a) Background and Motivation

SDN is a new networking paradigm that separates the control plane from the data plane to dynamically adjust the behavior of the network. Among others, OpenFlow is one of the foundational protocols that enables SDN through the communication between an SDN controller and the network devices. OpenFlow, designed by the Open Networking Foundation, offers access to and manipulation of the forwarding plane of network switches and routers. An SDN controller can use the OpenFlow protocol to instruct network devices on how to handle packets. This would then entail control of the paths that packets take, header changes at each node, and policies relating to management of traffic. OpenFlow enables ease of management of the network by centralizing control, hence making the network adaptive and programmable. The result of this is a much more agile network with better utilization of resources, advanced networking capabilities delivered through load balancing, and traffic engineering security features [1].

As the most deployed protocol in SDN, OpenFlow enables communication between the control plane and the data plane. In the case that OpenFlow messages are not encrypted, an attacker can intercept them and change them at will—severe security problems right there. Now that data security has become a more critical concern for every organization, the confidentiality and integrity of OpenFlow messages are very important. The encryption of these messages avoids risks arising from eavesdropping and man-in-the-middle attacks, increasing the overall security in SDN deployments [1] [2].

Ryu is an open-source SDN controller written in Python under the Apache 2.0 license. It serves as a flexible platform for managing network devices according to the OpenFlow protocol. Ryu provides a strong framework for SDN applications to be developed with a well-defined API and rich sets of libraries, supporting high levels of customization so that network operators could develop and deploy complex network management solutions. It provides control logic for various networking tasks, such as topology discovery, device management, and traffic engineering. On the other hand, Ryu ensures the effective handling of network events because of its event-driven nature and modularity for easy extension and integration with other systems. The platform supports multiple versions of OpenFlow and can work together with other protocols and tools, so it is suitable for both research and production environments. Through Ryu, a very high degree of control of the network and automation for enhanced performance in the network and reliability can be achieved by users [3].

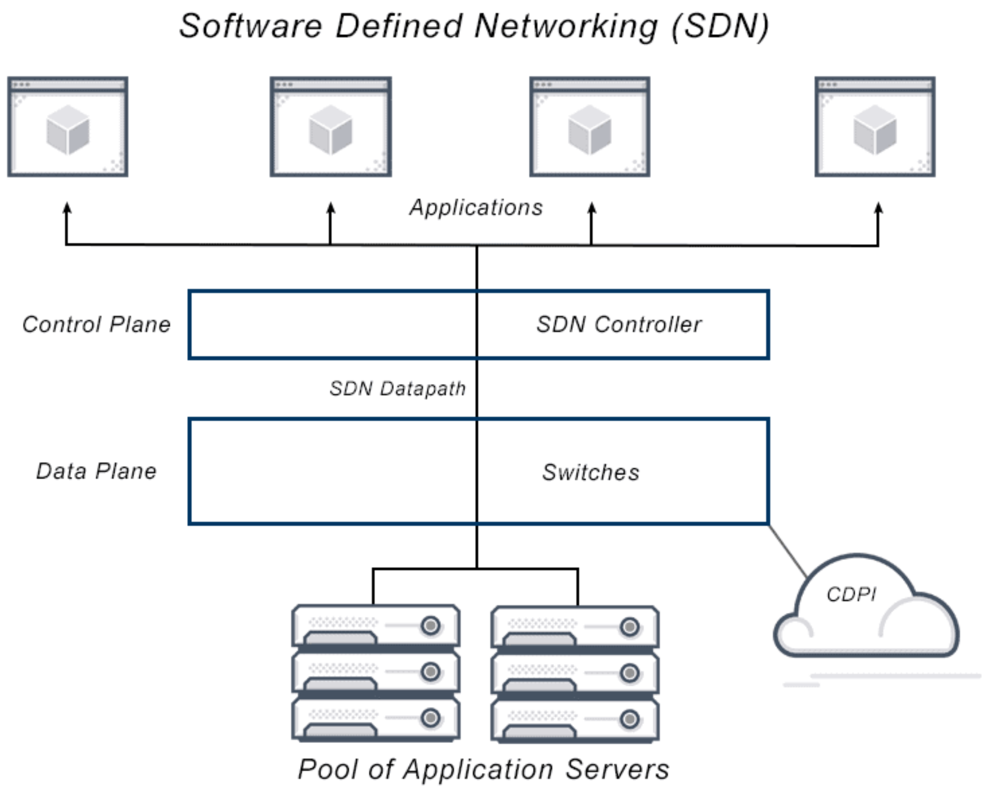

Figure 1. SDN Architecture [13]

b) Objectives

The purpose of this report is to demonstrate in practice encrypted OpenFlow communication using Transport Layer Security in the SDN environment. We will create a configuration for the secured communication channel using GNS3 as the network simulator, Open vSwitch as the virtual switch infrastructure, and the Ryu controller as the SDN controller. Primary aims:

i) Setting up a realistic SDN environment using GNS3.

ii) Installing and configuring Open vSwitch instances.

iii) Deploy and configure the Ryu controller

iv) generate and manage SSL certificates for establishing trust

v) configure OVS and Ryu to use TLS to perform encrypted communications

vi) verify that traffic is encrypted by capturing and analyzing it.

c) Significance

Since it is the SDN controller and devices in an SDN environment that are in charge of directing and managing network communication, securing this link in terms of integrity and confidentiality is of essence. This study gives an end-to-end guideline on implementing TLS encryption in an SDN setting; this adds to the body of knowledge by giving practitioners a practical solution to a very important security concern. Accordingly, the findings and methodologies presented in this report may be used for reference by network administrators and researchers who wish to improve the security of their SDN deployments.

d) Structure of the report

The structure of the report is as follows:

- Abstract: Very brief introduction of how encrypted OpenFlow communication was implemented and evaluated.
- Related Work: A state-of-the-art review of various literatures and earlier work related to securing SDN and encryption methods.
- Research Methodology: Step-by-step configuration of the GNS3 environment, the installation, and configuration of OVS and Ryu; generation of SSL certificates; setting up TLS.
- Results and Discussion: The evaluations on encrypted communication analysis and security benefits are presented here.
- Conclusion and Future Work: Summary of findings and potentials of areas that can be targeted for future research.

e) Overview of key technologies

GNS3: GNS3 is network software emulating a combination of virtual and real devices to simulate very complex networks. It provides a GUI to design and test network topologies, thus offering an excellent tool to network administrators and researchers for testing their networks [4].

Open vSwitch(OVS): OVS is a multilayer virtual switch designed for network automation through programmatic extensions while supporting standard management interfaces and protocols. It is highly utilized in virtualized environments to offer switching services [5].

Wireshark: Wireshark is a very popular, open-source network analyzer that captures and displays real details regarding network traffic in real time. It's especially useful for troubleshooting network problems, analyzing network protocols, and ensuring network security. Networks have to be monitored for smooth operations and security [6].

f) Challenges and Considerations

Setting up encrypted OpenFlow communication between the controller and switches has challenges with respect to compatibility, certificate management, and performance overhead. The main issue I faced during the whole setup process was my device being short of RAM, even though I had a 16GB DDR5 Ram chip. In this report, we focus on dealing with such challenges by detailing configuration steps, troubleshooting, and performance analysis to get a robust and secure SDN setup.

The testbed and the methodologies described in the report will allow network administrators and researchers to set up the same environment, further improve the security of their SDN environments, and continue contributing to developing safe and efficient network management.

2 RELATED WORK

a) Security threats and strategies in the SDN control plane

In the paper "A Detailed Examination of Security Threats and Strategies to Counter Them in Next Generation SDN Controllers" by Han et al. (2019), the authors focus on the security risks that have been inbuilt in the control plane within SDN. The major objective of this research is the identification of vulnerabilities and the proposition of mitigation strategies for enhancing the security in the plane of control [7].

- Main Goals

This paper presents the security risks in the SDN control plane, focusing on vulnerabilities in the southbound interface and proposes ways to handle them. The authors have classified various kinds of attacks together with their corresponding mitigation techniques, like encryption and protocol-specific defenses.

- Approach

The paper categorizes the possible attacks against the SDN control plane and discusses their mitigation strategies, such as custom encryption methods and protocol-specific defenses. The authors aim to make the southbound interface safe by putting in place mTLS for security during the transmission of data from the SDN controller to other network components.

- Findings

It has identified mTLS as among the most important measures in verifying the identity of the SDN controller and network components and securing data transmission between these two. This approach has improved security at the southbound interface tremendously by preventing unauthorized access and possible data breaches.

- Summary

The research shows a drastic enhancement in securing the southbound interface against unauthorized access attempts and potential data breaches by adopting mTLS. The findings underline the necessity of mutual authentication and encryption for the protection of the SDN control plane.

b) Enhancing Flow Table Management in SDNs

Management Efficiency in Software Defined Networks" by Correa Chica, Imbachi, and Botero Vega in 2020 [8].

- Key Aims

The primary objective of this study is to come up with strategies that optimize flow table management in an SDN environment to counteract TCAM overflow incidents. Several strategies have been considered in this study for enhancing flow table utilization efficiency.

- Methodology

The authors present techniques for flow table compression and dynamic strategies in flow entry management. These methods increase the packing of the flow tables to avoid TCAM overflow and ensure efficient network operation.

- Findings

These mechanisms are proved to be very effective against TCAM overflow by compressing flow tables and dynamically reordering flow entries. Combining these methods with periodic table maintenance and consolidating the flow rules makes the management of flow tables very efficient.

- Summary

This study concludes that with techniques for flow table management, a system applying methodologies for installing flow entries and proactive methods to prevent overflow makes the vulnerability of TCAM overflow attacks very minimal. The research contributes to the maintenance of the best SDN flow table performance.

c) Machine Learning for Enhanced Threat Detection in SDNs

In the paper "A Survey on Machine Learning Approaches for Anomaly Detection in SDN" by Farooq, Riaz, and Alvi, 2023, the authors have talked about harnessing machine learning techniques in enhancing threat detection within the SDN controllers [9].

- Primary Objectives

This research primarily seeks to assess machine-learning-driven anomaly detection systems in improving threat recognition and mitigation of the SDN controllers. During the study, the interest of using advanced machine learning models in the identification of network anomalies and responding accordingly drives the researcher.

- Approach

The authors have surveyed various kinds of machine learning models—some being deep learning and reinforcement learning—to analyze traffic patterns and user activities for anomaly detection. This approach to identification exploits subtle anomalies in network traffic that traditional methods would probably miss.

- Results

It has been shown that especially deep and reinforcement learning do very well in detecting subtle anomalies in network traffic. These models significantly improve the detection of, and response to, network threats relative to more traditional methods.

- Conclusion

It improves the detection rate and accelerates responses to network threats through the integration of machine learning systems into an SDN controller. Improved overall security in a network is achieved accordingly. This study underlines the potential of machine learning toward transforming threat detection and mitigation in the environment of SDNs.

d) Centralized Security Enhancement for TLS Flows in SDNs

The paper "An SDN-based Approach to Enhance End-to-End Security: SSL/TLS Case Study" suggests an original approach for the improvement of the security of data flows, based on analysis in the centralized SDN controller of the TLS handshake messages [10].

- Summary

The paper proposes a new solution toward enhancing the security of data flows by inspecting TLS handshake messages through a centralized SDN controller. Unlike traditional SDN firewall-based solutions, filtering flows using access control lists, this paper focuses on verifying vital parameters of the TLS handshake protocol, including protocol version, cipher suites, and certificates.

SDN Firewall Solutions: Traditional SDN firewall solutions forward the first packet of each flow to the controller, which enforces predefined ACLs in the same way. This includes solutions implemented using the Floodlight controller and approaches that detect and resolve conflicting firewall rules based on header-space analysis algorithms.

Other solutions, such as that of Enrique et al. (2016), prevent man-in-the-middle attacks by validating TLS certificates with Bayesian networks. Such solutions, however, remain silent regarding the validation of other important parameters of the handshake protocol.

- Comparative Approaches

Snort is one of the popular solutions monitoring flows and setting rules for the allowance of only certain versions and states of TLS. While Snort works well, the proposed SDN-based approach has the advantage of enforcing network-wide policies from a centralized control point.

- Conclusion

The solution proposed in this paper, based on SDN, enhances the security of a communication session by analyzing TLS handshake parameters within a centralized controller. Not only does it allow for the application of security policies throughout the network, but it also allows for finer-grained security against some well-known attacks on the TLS protocol. Experimental results show that such analysis adds a manageable overhead, which justifies the benefits coming from central control over compliance with the policy.

e) FLOWGUARD: Building robust firewalls for software-defined networks

This paper by Hongxin Hu, Wonkyu Han, Gail-Joon Ahn, and Ziming Zhao, "FLOWGUARD: Building Robust Firewalls for Software-Defined Networks," calls for the implementation of FLOWGUARD: a universal framework to enhance security against adversaries in dynamic OpenFlow-based networks. The primary objective of FLOWGUARD is to offer robust firewall capabilities to properly identify and resolve policy violation incidents in real time due to the changing nature of the network state and the indirect security violation scenarios, where both centralized and distributed firewall architectures perform these functions correctly. The methodologies used by FLOWGUARD to track network flows for detection of violation scenarios are that of flow path space analysis and firewall authorization space partitioning, excellently classified into complete or partial violations [11].

This paper uses multiple resolution strategies, which include dependency breaking through flow rerouting and tagging, update rejecting, flow removing, and packet blocking to maintain network utility while enforcing security policies. Implemented in the Floodlight SDN controller and evaluated with a real-world topology based on the Stanford backbone network bound FLOWGUARD exhibited the effectiveness in terms of efficient performance and scalability, successfully handling dynamic updates and policy violations. The performance of the framework had good proximity to already existing solutions, just adding some flexibility and robustness. Stateful packet inspection and extension of support to other SDN controllers and visualization tools are some of the future works to further increase applicability and effectiveness in securing software-defined networks.

f) Performance Evaluation of Ryu Controller in Software Defined Networks

In the paper "Performance Evaluation of Ryu Controller in Software Defined Networks," Alaa Taima Albu-Salih investigates the performance of the Ryu controller within the environment of an SDN-based network. In the study, the key metrics were latency and throughput. The Cbench tool was used to assess how efficiently a Ryu controller manages network traffic by controlling OpenFlow switches [12].

- Introduction

SDN stands for Software-Defined Networking and is realized with a significant development in network architecture; that is, decoupling the control plane from the data plane for enhanced flexibility, manageability, and performance of the network. A fully Python-based open source SDN controller, Ryu, has been provided and is based on the latest OpenFlow protocols in programmable management of the network.

- Objective:

One of the main objectives of this paper is to study the performance of the Ryu controller in latency and throughput. This evaluation has helped to quantify the efficiency of the controller in handling network traffic and its suitability for different network applications.

- Methodology

Testing was done on Mininet, an emulation environment for creating virtual network topologies, and Cbench, a benchmarking tool designed to test the performance of SDN controllers. Mininet was used to simulate the network environments, while Cbench was used to probe the controller for throughput and latency. In the setup, there was a variation of the number of switches in the network to see how the performance scaled for the Ryu controller.

- Results:

Latency: The study measured the time taken by the controller to process packets from packet_in to flow_mod. The results indicated that latency increases along with a number of switches due to a higher processing load on the controller.

Throughput: This was measured in terms of the number of flow_mod messages the controller could send back after receiving packet_in messages. Results: its throughput decreased as the number of switches increased until it hit a point where the controller's queue was saturated, hence the huge drop in performance.

- Conclusion

The performance of the Ryu controller was inversely proportional to the number of switches on a network. Both latency and throughput thus take negative hits as the number of switches increases. These results clearly show the significance of consideration of the performance limitations of the controller in design and on-field deployment of SDN-based networks. In the future, this evaluation will be extended to other performance metrics like the packet drop rate, RTT, and jitter to give a more comprehensive performance profile of the Ryu controller.

3 RESEARCH METHODOLOGY

This section details the research methodology employed to secure Software-Defined Networking (SDN) OpenFlow messages. The setup involved the use of GNS3 for network emulation, Open vSwitch (OVS) for virtual switching, and the Ryu controller for SDN management. The objective was to create a secure SDN environment by ensuring the integrity and confidentiality of OpenFlow messages between the controller and the switches. The methodology is broken down into several stages: environment setup, network topology design, security configuration, and validation of security measures. network management practices.

a) Environment Setup

Setting up the environment is a very critical stage in the project since it makes sure that everything works properly and cohesively. This section details installation and configuration of GNS3, Open vSwitch, and the Ryu controller.

i) GNS3 Installation and Configuration
GNS3 (Graphical Network Simulator-3) is network software emulator that offers a graphical interface used in emulating complex networks.

- Installation: GNS3 was installed on a dedicated server running a Linux-based operating system to leverage its stability and performance. Equipped with ample resources of CPU, memory, and storage, this server could easily handle the computational load of a number of virtual devices. This installation entailed downloading the GNS3 software package and its dependencies, followed by the execution of installation scripts provided by GNS3.

- Configuration: After installation, the GNS3 server was configured for advanced performance and resource management in a network. This is where the setting of virtual network interfaces took place, isolation of CPU cores, and memory allocation for optimum performance. This kind of integration with the host operating system helped virtual devices communicate with each other and external networks if there was such a need. The final step in this process was the configuration of the GNS3 GUI to ensure efficient management and visualization of the network topology.

ii) OVS Setup and configuration
Open vSwitch or OVS is a virtual switch that is high-performance and programmable to manage packet switching in the virtualized infrastructures.

- Deployment: Open vSwitch is deployed on each virtual instance across the GNS3 platform. Each instance of the switch can route packets using the rules it receives from the SDN controller.

- Bridge Configuration: Virtual bridges were configured inside OVS to connect the virtual network interfaces. It developed an internal communication path of various network segments in OVS; this also formed a pathway for the forwarding of packets between the virtual hosts.

- Integration with GNS3: OVS instances were integrated into the GNS3 GUI, which meant it could control and monitor the OVS switches from the GNS3 interface.

iii) Ryu Controller Setup
Ryu is an SDN controller developed in Python that allows a wide range of network management capabilities to be implemented.

- Installation: A Ryu controller installation was completed on a VM inside the GNS3 environment. This involved setting up a Python environment and the installation of Ryu software, together with all its dependencies, through package management tools like pip.

- Configuration: Configuration of the Ryu controller for use with OVS switches over the OpenFlow protocol. Configuration files were edited, defining the IP address of the controller, its listening port, and other parameters to establish OpenFlow sessions with the switches. Loading some Ryu applications necessary in this project into the controller was also done. These include topology discovery, traffic management, and security modules.

b) Network Topology Design

Design of a network topology is an important step in setting up a secure SDN environment. It deals with where to place switches, hosts, and controllers in terms of their configuration so that all entities are able to communicate effectively and securely. The topology was designed to be very close to a real-world SDN deployment and provided a strong framework for testing the security measures in place.

i) Topology Structure

The network topology was designed with the following structure to emulate a real-world enterprise network:

- Core Switches: These essentially act as the backbones of the networks to which high-speed interconnectivity and data routing primary layers are attached.

- Distribution Switches: These switches aggregate the traffic from the access switches before connecting to core switches for efficient traffic management and policy enforcement

- Access Switches: These are used to connect end-user devices in a network, normally hosts; they deal with local traffic and provide the first level of data forwarding.

- Router: A cisco router was installed to route internal traffic to the internet as it was connected to NAT device in the GNS3 environment. It also acted as a default gateway for all the devices in the network

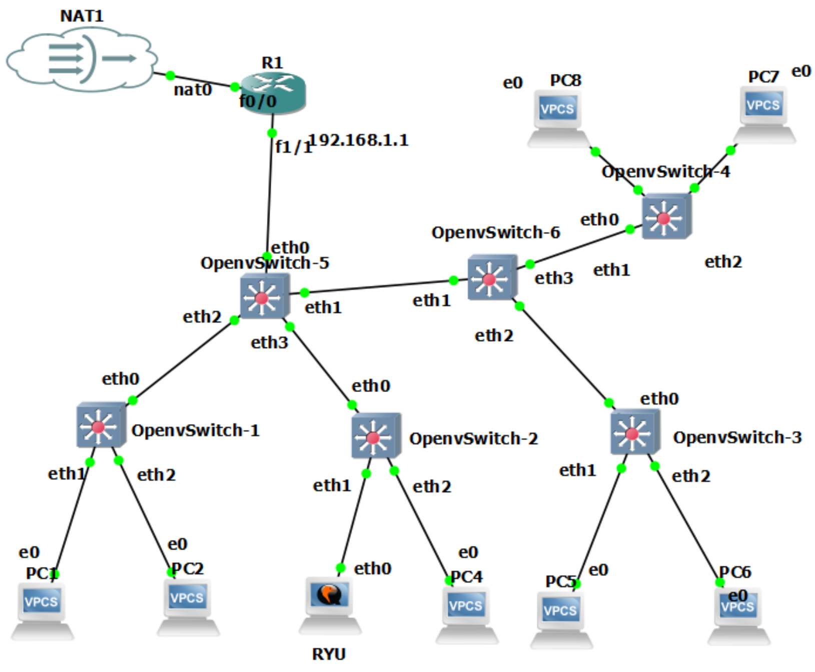

Figure 2. Network Topology

ii) Switch Deployment

- Deployment of core switches: Two OVS switches were used for core switches, which act as the central hub of the network. Configuration was done for high-capacity links in order to carry huge volumes of traffic and provide redundancy for fault tolerance.

- Deployment of distribution switches: Four OVS switches were used as distribution switches. Every distribution switch was then connected to both core switches to make sure that there were multiple pathways through which data would travel, hence adding resilience to the network.

- A number of OVS switches were used as access switches and then had a direct connection to the distribution switches. These access switches were supposed to connect all the virtual hosts to the network and thereby provide the required ports and bandwidth for the end-user devices.

iii) Host Integration

- Configurations of Virtual Hosts: Virtual hosts were implemented in the GNS3 environment to model all devices that link to the network from the end-users. Connections to the access switches were made, and they were allowed to communicate in the SDN environment to build network traffic to be used for testing purposes.

- Selection of the Operating System: Lightweight operating systems, such as Linux Ubuntu 20.04, were installed on the virtual hosts to save resources. These operating systems provided all the relevant networking tools and utilities for generating traffic and participating in network tests.

- IP Address Allocation: Every virtual host was statically configured with an IP address from a predefined range of IPs. Uniformity in addressing made it easy and quite convenient for management and troubleshooting purposes across the network.

iv) Controller Connectivity (Ryu)

- Establishing an OpenFlow Session: Every OVS switch was configured to establish a secure OpenFlow session with the Ryu controller. This involved putting the IP address and the controller's port number in configuration files for OVS.

- Secure Communication Setup: Against this backdrop, considering the need for secure communication between the switches and the controller, TLS was enabled. This demanded the generation of installation on both the controller and switches with SSL/TLS certificates, detailed in the security configuration section.

- Control Plane Management: The Ryu controller configuration, managing and monitoring the OVS switches, was performed. This included loading particular Ryu applications that provided topology discovery, link management, and traffic engineering functionalities. The exposed interface to the controller allowed for real-time monitoring and control of the network, hence enforcing security policies and rules on the traffic.

v) Link configuration

The network topology was configured to ensure efficient data flow and secure communication between all components.

- Virtual Ethernet Interfaces: The network links between switches were created using virtual Ethernet interfaces GNS3 provided. The links thus made were configured to allow OpenFlow communication; hence, data transfers between switches and the controller will be reliable and efficient.

- Configuring Bandwidth and Latency Settings: The bandwidth and latency settings of these virtual links were changed to very practical network conditions. This is where the speeds of the links were configured to standard values—for instance, 1 Gbps on access links and 10 Gbps on core links—and latency added where appropriate to simulate distance and load on the network.

vi) Ip addressing

- Subnet Allocation: Different subnets were allocated to different segments of the network, such as core, distribution, and access layers. Partitioning of the network helped in managing traffic effectively with much-enhanced security by separating different types of traffic.

- Static IP Assignments: Static IP addresses were assigned to each OVS switch and the Ryu controller. This provided stable and predictable communication between them so that there would be no hassle in managing or troubleshooting the network.

- Configured routing tables for directing appropriate traffic in a network. Static routes were added to guarantee that the flow of traffic moves efficiently to the destination without any unnecessary delay within the network.

vii) Security Configuration
Securing the SDN environment involved implementing measures to protect OpenFlow messages and ensuring the integrity and confidentiality of the control plane communication.

- Configuration of TLS: The configuration targets, having OpenFlow messages between the Ryu controller and the OVS switches encrypted. It comprises generation for SSL/TLS certificates, certificate management using a CA, and deployment of such to target devices.

- Certificate Management: For this, OpenSSL was used to generate certificates and sign them through a reliable CA. The root certificate of the CA is distributed to all devices to allow the chain of trust to be established.

- Signed certificates, regarding the corresponding private keys, were deployed on the Ryu controller and on instances of OVS. Configuration files were updated by setting paths to these certificates and keys; OpenFlow TLS parameters were set to enforce secure communication.

4 RESULTS

a) Successful Implementation and Running of the Whole Topology in GNS3

The aim of the project was to provide a secure environment for Software-Defined Networking using GNS3, Open vSwitch, and the Ryu controller. Successful running of this topology in GNS3 proved the efficacy of the tool for simulation purposes in a complex network environment. GNS3 offered a flexible platform to design, configure, and test the network for ensuring that all the components work together in harmony.

b) Network Topology and Configuration

Design: A network topology was designed using GNS3, wherein there were multiple instances of OVS connected with routers and end devices like PCs. Therefore, various OVS instances are configured with bridges and ports for the working out of communication between different devices in a more efficient way. Integration: The Ryu controller has been integrated into this topology, which has managed the OVS instances by following the OpenFlow protocol.

The project proved to work with the creation of a functional and secure SDN environment through detailed planning and configuration.

c) DHCP server implementation

The final component for the network setup was running a DHCP server in order to assign IP addresses to devices in the network automatically. In this automation, there was a reduced need for IP configuration manually in each device. This eased the setup process and allowed the devices to communicate effectively.

d) Configuration of the DHCP Server:

IP Address Assignment: Addresses were to be assigned by the DHCP server within a certain range to the various end devices connected to the OVS instances.

Configuration Settings: Other configurations of the network, such as gateway and DNS server details, were configured through the DHCP server.

IP Configuration: ipconfig under Windows and ifconfig under Linux are facilities through which devices can obtain IP addresses and other network configurations from the DHCP server.

Connectivity: Ping tests between devices were successfully conducted, which proved the DHCP server was working properly, and devices were configured accordingly.

e) Controller Connectivity

It was necessary to check that the Ryu controller was connected to each instance of OVS in its role of performing dynamic control of flow rules on switches. This made the control plane of the network centralized.

Connection Setup: Using the ovs-vsctl command, each connection setup of an instance of OVS to the Ryu controller was done, only by stipulating the IP address and the port of the controller, 6653.

Management: The Ryu controller managed the OVS instances' dynamic flow rules according to network policies.

Check Status: In this part, the ovs-vsctl show command was used to check whether all OVS instances were successfully connected to the Ryu controller.

Controller Logs: The logs of the Ryu controller showed successful connection and management of the OVS instances. Active communication and updates of rules were evidenced in the logs.

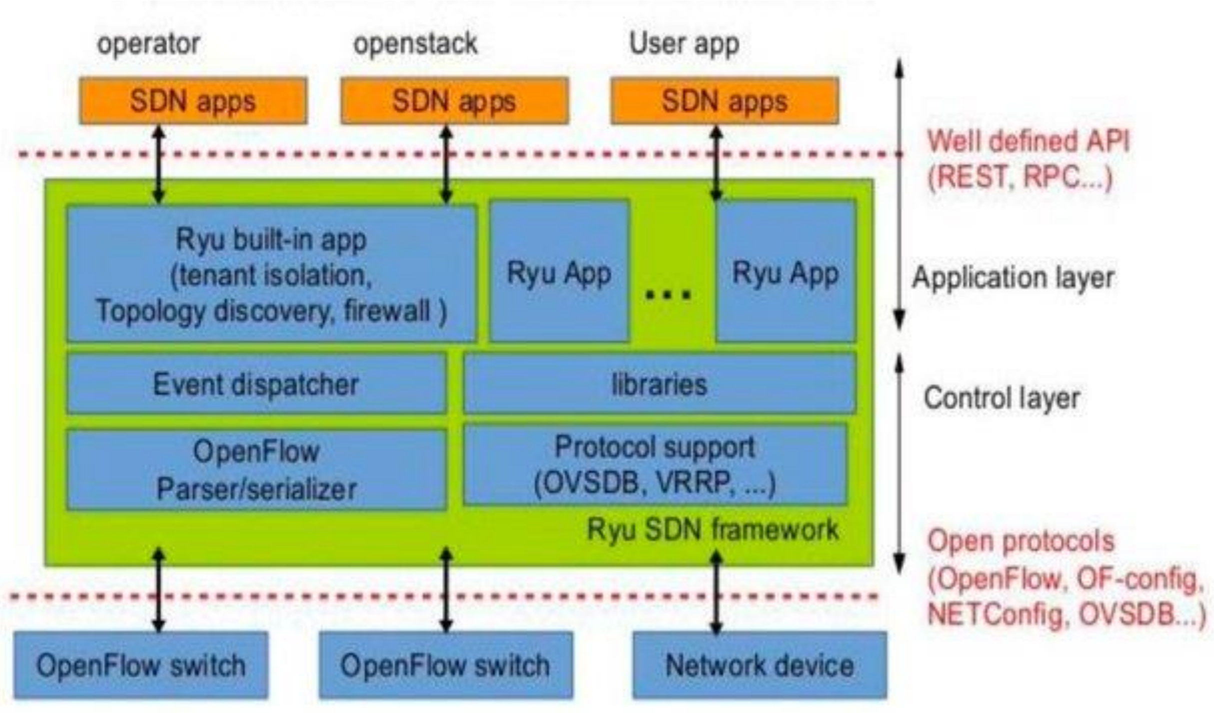
operator
openstack
User app
SDN apps
SDN apps
SDN apps
Well defined API (REST, RPC...)
Ryu built-in app (tenant isolation, Topology discovery, firewall)
Ryu App
...
Ryu App
Application layer
Event dispatcher
libraries
Control layer
OpenFlow Parser/serializer
Protocol support (OVSDB, VRRP, ...)
Ryu SDN framework
Open protocols (OpenFlow, OF-config, NETConfig, OVSDB...)
OpenFlow switch
OpenFlow switch
Network device

Figure 3. Ryu Architecture [14]

f) Implemented TLS

The OpenFlow communication from the Ryu controller to the OVS instances was secured by implementing TLS encryption. This exercise entailed generating SSL certificates, configuring the OVS instances to use these certificates, modifying the Ryu controller to deal with encrypted connections, and distribution.

Generation and Distribution of Certificates;

CA Certificate: First, a Certificate Authority was created, and then SSL certificates for Ryu controller and instances of OVS generation; these were distributed to the relevant devices and configured accordingly.

Configuration and Verification;

Setup SSL by issuing ovs-vsctl set-ssl command on the instances of OVS.

Controller adjustment: Setting up Ryu controller application to support TLS.

Traffic Analysis: By using Wireshark, it could be confirmed that OpenFlow messages had been encrypted. This is evident from the captured traffic, which includes TLS handshakes and encrypted application data.

g) Network Stability

During the setup and configuration, it could be noted that the stability of the network was very sensitive to correctness regarding configurations on both the switches and routers. Even a small misconfiguration could bring about significant problems in connectivity, and the problems were spread over the functionality of the whole network.

Sensitivity in Configuration Changes:

Configuration Integrity: The wrong setting of OVS instances or routers might lead to loss of connectivity and instability of the network in general.

It had detailed logging, which helped in swiftly identifying and rectifying any configuration issues. This impacted network operations in the following ways:

Precise Configuration: Correct, accurate configurations avoided disruptions of the network's stability.

Troubleshooting Efforts: Detailed verification and troubleshooting steps quickly identified and rectified misconfigurations that might have caused the network to fail.

Settings on the NAT of the router were made for translation of the internal IP addresses into the public IP address so that access to the internet could be provided for the devices inside. The rules under the firewall are modified to allow outgoing traffic and back response.

The devices across the network could access the internet without any issues, thus proving that the setup of NAT works fine. Functional testing, in terms of web browsing, pinging external websites, and other internet activities, proved that everything regarding the NAT configuration was successful and the entire functionality of the network. Summary The SDN topology set up in GNS3 with OVS instances, the Ryu controller, DHCP server, and NAT was finally implemented. Security of the OpenFlow messages was improved with the introduction of TLS encryption. The project proved that the network was sensitive to configuration changes, hence it requires exact and accurate settings in order to remain stable. Finally, devices could access the internet by the successful setup of NAT; therefore, it completed the requirements of a functional and secure SDN environment. Results show how effective GNS3 can be as a tool for simulating SDN, and secure communication has become critical in today's networking environment.

5 DISCUSSIONS

This demonstrates that GNS3, Open vSwitch, and Ryu Controller can be used together to simulate a network environment and secure it. Key findings, challenges, and implications for this project will be discussed in the next sections.

- Successful Implementation and Running of the Whole Topology in GNS3

GNS3 was a very important tool in design and simulation of the SDN environment. With the flexibility to build up a real network topology and integrate different network devices, GNS3 was used to simulate these interactions. Since all components of the network are visible and can be manipulated in a graphical interface, it became easier to set up and debug, thus helping in discovering the problems and their solution.

Successful configuration of the OVS instances and integration with the Ryu controller proved that GNS3 was effective in conducting SDN simulations. The project proved that even very complex, large network set-ups can be managed and controlled by GNS3 robustly and is a good platform for testing and validation.

- DHCP Server Running

This was in respect ensured by the implementation of a DHCP server in the network, which automated the assignment of IP addresses to all connected devices. This introduced automation that reduced administrative overhead and allowed devices to communicate in the test without manual configuration. Checking the functionality of the DHCP server—by successful IP address assignment and connectivity tests—underlines the importance of automated network configuration in large-scale deployments.

- Ryu Controller

Therefore, this step of connecting all OVS instances to the Ryu controller was a very critical point in this proof-of-concept exercise. Having done so, the authors would be in a position to exploit other major advantages of SDN: centralized control and dynamic management of a network. By doing this, it would be possible to prove how SDN will dynamically update the flow rules based on network conditions in order to facilitate efficient usage of resources and to optimize network performance.

Setting up the connection between OVS instances and the Ryu controller by using the ovs-vsctl command was pretty smooth and very effective. Controller logs and network behavior proved the success of the connection.

- TLS Implemented

This involved another major accomplishment: securing the communication between Ryu's controller and OVS instances by TLS. That is basically generation of SSL certificates, distribution, and configuration of a controller and switches to use those very certificates. This has been able to show that secure communication is important in order to prevent unauthorized access and guarantee data integrity.

Wireshark traffic analysis has shown that OpenFlow messages were successfully encrypted, which proves that the implemented TLS is effective. This encryption is critical in modern networks to counter eavesdropping and man-in-the-middle attacks, improving the security configuration for the SDN environment.

- Network Stability and Sensitivity to Configuration Changes

One of the interesting observations made during the project exercise was the sensitivity of the network to configuration changes. Even slight misconfigurations in OVS instances or routers caused huge connectivity disruptions, thus proving that settings have to be very precise and accurate.

This sensitivity also brought home the need for careful planning and checking in network configuration. It was constant monitoring and detailed logging that helped quickly identify and remedy all problems, keeping the network stable and up. The project proved that sometimes network functionality can behave like an on/off switch; one misconfiguration and a whole connectivity can be lost.

- NAT Successful

It was necessary to perform Network Address Translation to allow devices within the SDN environment to access the Internet. With no issues, communications between the internal network and the outside were allowed by the successful configuration of NAT on the router. It was important for testing how the network works at a practical level, where access to the Internet is usually required.

The success meant that one could be able to browse the web, ping sites external to the network, and do other activities over the internet. Such success showed that the configured network was practical and ready for deployment in production environments.

6 CONCLUSION & FUTURE WORK

Testing the Ryu controller in an SDN setup was very instrumental in bringing out its operational strengths and weaknesses. The study revealed that even though the Ryu controller efficiently controlled network traffic by implementing policies through the OpenFlow protocol, it is immensely affected by the number of switches available within the network. Quantitatively, key metrics, such as latency and throughput, have shown that with an increase in the network, controller efficiency drops, raising critical scalability challenges. These findings are of paramount importance to network designers and operators who seek to deploy SDN in large-scale environments.

The paper confirms that Ryu works well for smaller-to-medium-sized networks where the number of switches and traffic load remains within manageable bounds. However, in replicating this into the larger and more dynamic networks, the controller performance problems have to be resolved for the proper functioning of the network. Latency surge and drop in throughput with the addition of more switches bring out the requirement for improved strategies in scalability and performance optimization.

Several research directions and practical implementations can be aimed at enhancing the performance and scalability of the Ryu controller and other SDN controllers. These efforts can be oriented not only to mitigate the limitations identified in the present work but also toward the discovery of new functionalities and optimizations for SDN environments.

Dynamic management of flow table: Dynamic algorithms that prioritize and manage flow entries based on usage patterns may avoid TCAM overflow. Techniques such as flow table compression and aggregation can be applied to optimize TCAM space.

Offloading flow rules: Place the less-critical flow rules on software-based flow tables. This will free TCAM space that can be filled with high-priority entries. This will be a hybrid way by which to balance performance and resource utilization.

Automated Network Management: Developing automation tools for routine network management tasks—like updating flow rules and changes in network topology—may reduce the administrative overload and human errors.
Orchestration Frameworks: Integration of the Ryu controller with orchestration frameworks like Kubernetes or OpenStack will allow managing network resources in cloud environments in a seamless way, harnessing the power of scalability and flexibility.

Intrusion Detection Systems (IDS): Adding the capabilities of IDS to the Ryu controller will result in an enhancement in terms of networking security. It can be done for machine-learning-based anomaly detection in real time, for the identification of threats so that it can be rectified.

Controller Clustering: Multiple instances of the Ryu controller running in a clustered configuration can share the processing load, making it more scalable and fault-tolerant. This will ensure high availability and reliability for large networks.

Load Balancing Algorithms: More complex load-balancing algorithms can be designed to uniformly distribute network traffic across controllers, optimizing resource utilization and overall performance.

Hardware Acceleration: Hardware acceleration using technologies such as Field Programmable Gate Arrays or Graphics Processing Units can increase processing capabilities of Ryu controller by a huge amount with tremendous reduction in latency and increased throughput.

Optimized packet processing: By developing efficient algorithms to process packets with the least processing overhead, the controller's responsiveness can be increased and latency reduced.

REFERENCES

[1] ieeexplore.ieee.org. (n.d.). *Software-Defined Networking (SDN): A Review | IEEE Conference Publication | IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/document/9972067.

[2] Jammal, M., Singh, T., Shami, A., Asal, R. and Li, Y. (2014). Software defined networking: State of the art and research challenges. *Computer Networks*, 72, pp.74–98. doi:https://doi.org/10.1016/j.comnet.2014.07.004.

[3] Asadollahi, S., Goswami, B. and Sameer, M. (2018). Ryu controller’s scalability experiment on software defined networks. *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*. doi:https://doi.org/10.1109/icctac.2018.8370397.

[4] Gns3.com. (2020). Available at: https://www.gns3.com/.

[5] www.openvswitch.org. (n.d.). *Open vSwitch*. [online] Available at: https://www.openvswitch.org/.

[6] Wikipedia Contributors (2019). *Wireshark*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Wireshark.

[7] Han, T., Jan, S.R.U., Tan, Z., Usman, M., Jan, M.A., Khan, R. and Xu, Y. (2019). A comprehensive survey of security threats and their mitigation techniques for next-generation SDN controllers. *Concurrency and Computation: Practice and Experience*, 32(16). doi:https://doi.org/10.1002/cpe.5300.

[8] Correa Chica, J.C., Imbachi, J.C. and Botero Vega, J.F. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, 159, p.102595. doi:https://doi.org/10.1016/j.jnca.2020.102595.

[9] Farooq, M.S., Riaz, S. and Alvi, A. (2023). Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review. *Electronics*, [online] 12(14), p.3077. doi:https://doi.org/10.3390/electronics12143077.

[10] Ranjbar, A., Miika Komu, Salmela, P. and Aura, T. (2016). An SDN-based approach to enhance the end-to-end security: SSL/TLS case study. doi:https://doi.org/10.1109/noms.2016.7502823.

[11] Hu, H., Han, W., Ahn, G.-J. and Zhao, Z. (2014). FLOWGUARD. *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN ’14*. [online] doi:https://doi.org/10.1145/2620728.2620749.

[12] Alaa Taima Albu-Salih (2022). Performance Evaluation of Ryu Controller in Software Defined Networks. *Magallaẗ al-qādisiyyaẗ li-ʿulūm al-ḥāsibāt wa-al-riyāḍiyyāt,* 14(1). doi:https://doi.org/10.29304/jqcm.2022.14.1.879.

[13] Google.com. (2024). *Redirect Notice*. [online] Available at: https://www.google.com/url?sa=i&url=https%3A%2F%2Favinetworks.com%2Fglossary%2Fsoftware-defined-networking%2F&psig=AOvVaw16J4G4lBPwMQiAA-

5axQu0&ust=1723509158784000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCOCG_Iqa7ocDFQAAAAAdAAAAABAE [Accessed 12 Aug. 2024].
[14]	ryu-sdn.org. (n.d.). *Ryu SDN Framework*. [online] Available at: https://ryu-sdn.org/.