A Dataset for Detecting Intrusion in Software Defined Networking Infrastructures

By
Jahid Imran
203-15-3887

FINAL YEAR DESIGN PROJECT REPORT

This Report Presented in Partial Fulfillment of the Requirements for the **Degree of Bachelor of Science in Computer Science and Engineering**

Supervised by

Md. Sazzadur Ahamed

Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Co-Supervised by

Afjal Hossan Sarower

Senior Lecturer
Department of Computer Science and Engineering
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

January 13, 2025

APPROVAL

This Project titled "**A Dataset for Detecting Intrusion in Software Defined Networking Infrastructures**," submitted by **Jahid Imran** to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 13 January, 2025.

BOARD OF EXAMINERS

Dr. Md. Zahid Hasan — **Chairman**
Associate Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Mr. Saiful Islam — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Mr. Afjal Hossain Sarower — **Internal Examiner**
Senior Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Ahmed Wasif Reza — **External Examiner**
Professor
Department of Computer Science and Engineering
East West University

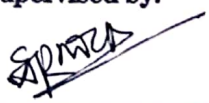
DECLARATION

I hereby declare that this project has been done by me under the supervision of **Md. Sazzadur Ahamed, Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University. I also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

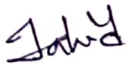
Md. Sazzadur Ahamed
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Co-Supervised by:



Afjal Hossain Sarower
Senior Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by:



Jahid Imran
Student ID: 203-15-3887
Department of Computer Science and Engineering
Daffodil International University

ACKNOWLEDGEMENTS

This work would not have been possible without the support and contributions of many individuals over the past two semesters. We are deeply grateful to everyone who has assisted us in one way or another.

First, I express my heartfelt thanks and gratefulness to the almighty for His divine blessing making it possible for me to complete the **Final Year Design Project(FYDP)** successfully.

I am grateful and wish my profound indebtedness to **Md. Sazzadur Ahamed**, **Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh. Deep knowledge and keen interest of my supervisor in the field of **Network and Information Security** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to extend my gratitude to **Afjal Hossain Sarower**, **Senior Lecturer**, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh for his continuous support and encouragement throughout this project. Additionally, I'm also indebted to the **Security and Applied Research (SAR)** lab for the technical support it provided in completing this project.

I would like to express my heartfelt gratitude to the Head of the Department of Computer Science and Engineering, for his kind help in finishing our project and also to other faculty members and the staff of the Department of Computer Science and Engineering, Daffodil International University.

I would like to thank my entire course-mates at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, I must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

Traditional networking works solely on the responsibility of hardwares (routers, switches) in moving the data between networks and calculating the routing table through various algorithms. Software Defined Network (SDN) is a networking phenomenon that moves the individual control plane of each networking device to a centralized SDN controller. Thus the decision making is moved to a centralized software system. This method is used to reduce the complexity of large infrastructures into a central controlling device. SDN allows dynamic and programmatically efficient network management of large corporations for network administrators. Due to its ease of use and implementation, this method is widely implemented into many of the top data center's networking environments. As a crucial networking component, it is of high importance to maintain security mechanisms for SDN implemented scenarios. Because of the sensitivity of the networking infrastructures it covers, cyberattacks on SDN powered data centers or companies are very common. Intrusion detection systems here can aid the way in monitoring malicious activities into the network. Traditional networking has had much research on them for IDS but their solutions don't work well when implemented in an SDN specific environment. The IDS that exists are trained on non-SDN environments mostly and the SDN specific dataset is outdated and needs more new attack type data in them. The lack of dataset collected on SDN environments is stopping researchers from making advancements into making IDS specific to SDN environments. Also, the existing dataset lacks real world scenario-based implementations and up-to-date data. This proposed research aims to mitigate such issues by creating a dataset that resembles a real world based complex scenario by attacking a simulated environment with windows, linux systems and various services that use software defined networking systems.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

In the case of traditional networking the routers are responsible for both control plane and data plane. The control plane decides the routing table and the data plane handles how the data moves into a network. In the case of software defined networks there is a central SDN controller that handles the control plane and all the switches or routers only handle how the data moves into a network. Because of the centralized control over a network SDN is widely being used and implemented but lacks proper up-to-date security features to fight modern cyber-attacks. Implementing modern Intrusion Detection Systems can help prevent and monitor such cyber-attacks but lack of SDN based datasets are creating problems for researchers in doing so. This study aims to aid researchers in making such IDS by creating a dataset that is up-to-date, contains various modern attack vectors on SDN and resembling complex networking infrastructures.

1.2 Motivation

While the evolution of networking technology has paved the way for Software Defined Networking (SDN) that enabled centralized management and control over complex networks, it also introduced the challenge of ensuring security against sophisticated cyber attacks. Existing IDS solutions don't really address the unique architecture and vulnerabilities of SDN infrastructures and the lack of dataset in making IDS solutions is also slowing the development of robust IDS solutions. Thus this research is motivated by the need to bridge the gap in IDS capabilities for SDN-specific environments. By creating a dataset that simulates real world scenarios, services and using up-to-date cyber attack patterns, this project aims to help in advancing SDN security research. Solving this problem will contribute to the broader field of cyber security and provide the next step for researchers to work on. Personally, this project aligns with my goal of becoming a cybersecurity researcher and advancing the research in protecting sensitive data and systems.

1.3 Objectives

The primary objective of this project is to develop a comprehensive and realistic dataset for intrusion detection in SDN infrastructure. This dataset will fill up the lack of publicly

available datasets specifically designed for intrusion in SDN networks. This dataset will simulate a real-world scenario using a combination of Windows and Linux systems, alongside various services running in SDN environments. It will also include a wide range of attack types and patterns reflecting contemporary cyberthreats faced by SDN-powered organizations. Another purpose of this research is to address the shortcomings of existing datasets by providing updated, diverse and complex attack vectors specific to SDN-specific architectures. By achieving these objectives, the project will support the development of efficient and reliable IDS solutions for SDN environments.

1.4 Methodology

This research involves creating a simulated environment that uses Software Defined Networking (SDN) to communicate between end devices. The environment is created inside VirtualBox using linux and windows virtual machines that depicts the end devices inside a network.

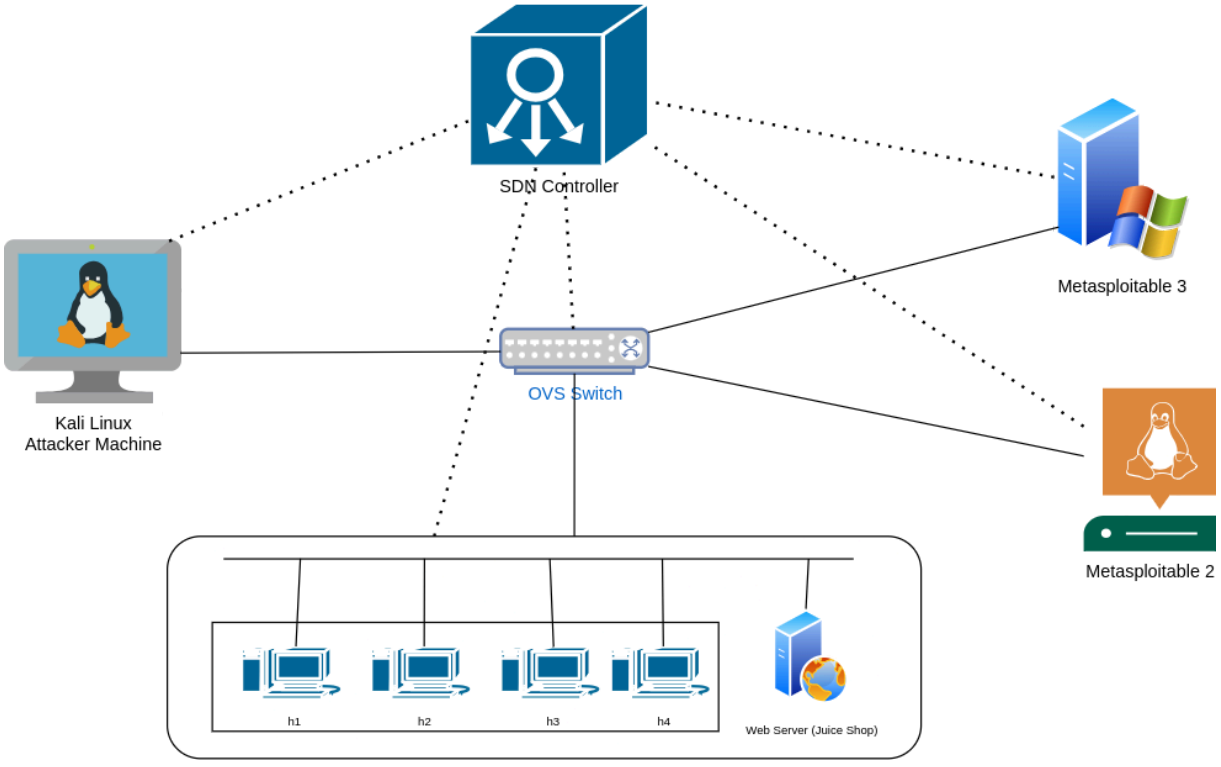


Figure 1.4.1: Research methodology overview

We've used a total of 5 virtual machines here. 2 Ubuntu 24.04 LTS, 1 Kali Linux, 1 Metasploitable 2 Linux Server and another Metasploitable 3 Windows Server. 1 Ubuntu Virtual Machine (VM) was used to set up SDN controller another to set up Open VSwitch (OVS) switch, mininet hosts and web server. The Kali Linux acts as an attacker machine here

while the Metasploitable 2 and 3 servers up services and acts as users on the network. After setting up the machines inside Virtualbox we configure the network settings to use SDN as the communication medium by connecting all of them together with OVS switch. Then using the Kali Linux machine we performed various attacks on the network including DoS, DDoS, Web Attacks based on OWASP Top 10, Remote-to-Local (R2L), Probing, User-to-Root (U2R) exploitation etc targeting various services like web, samba, ftp, ssh etc. We recorded the generated network traffic using tools like tshirk and wireshark. Then we converted the recorded network traffic pcap file into csv to create the dataset. This methodology creates a dynamic and realistic SDN environment for intrusion detection research, emphasizing modern attack scenarios and vulnerabilities to develop up-to-date dataset.

1.5 Project Outcome

The project aims to deliver a publicly available dataset designed to emulate real-world SDN environments with diverse attack scenarios and traffic patterns, enabling robust IDS development and evaluation. By addressing the gap in SDN-specific datasets, this project will enable researchers to explore advanced detection techniques, including machine learning and AI-based approaches, specifically tailored for SDN environments. The dataset will facilitate the creation of more effective and specialized IDS solutions, contributing to the overall security of SDN-powered infrastructures in organizations and data centers. This research will serve as a foundational resource for the cybersecurity community, providing insights into the unique challenges and solutions for securing SDN environments. The outcomes will have both academic and practical significance, empowering researchers, practitioners, and organizations to strengthen their defenses against evolving cyberthreats.

1.6 Organization of the Report

Chapter 1: Introduction to the research study discussing the fundamental concepts of Software Defined Networking (SDN) highlighting the challenges and research opportunities in this field. Also discusses the motivation, final outcome of this project and the contribution.
Chapter 2: Background reviews the related research works briefly discussing the existing solutions and their limitations. It also outlines the research gap that the proposed research seeks to address.
Chapter 3: Research methodology details the approach this research follows in creating the simulated SDN environment for making the data collection possible. A detailed description

for the steps followed in this study is outlined and discussed.

Chapter 4: Implementation and result presents the outcome that has been achieved through this research. It describes the dataset features, attributes and categories that has been collected during the research process.

Chapter 5: Engineering standards and design challenges examines the standards and best practices followed during this research to ensure reliability and robustness of the dataset.

Chapter 6: Conclusion summarizes the entire research, key findings, limitations and provides recommendations for future research.

Chapter 2
Background

2.1 Introduction

Software Defined Networking (SDN) was introduced not very long ago in the networking world. Even though adaptation to SDN from traditional networking was relatively slow, very little novel research has been conducted on them. The security of large enterprises was always a concern. In this review, we will explore existing research and related works in the advancement of the implication of IDS for SDN security. Existing works have been thoroughly read and analyzed for gap analysis in this section.

2.2 Literature Review

KDD '99 dataset (KDD Cup 1999 [2]) was widely used for intrusion detection (Divekar et al, 2018 [1]) having a total of 41 features but the dataset had redundancy issues. This dataset was also for traditional networking infrastructures.

A modified dataset was later introduced called NSL-KDD (Tavallaee et al, 2009 [2]) having additional 17 new attack attributes but both KDD'99 and NSL-KDD dataset lack up-to-date realistic network traffic records because they were made available decades ago.

Tang et al, 2016 [3] and Tang et al, 2018 [4] have proposed intrusion detection systems for SDNs using the NSL-KDD dataset which achieved an accuracy of 75% and 89% respectively but used only 6 raw features among 41. However, there is a lot of detection rate and a very high false alarm rate in this case as selecting appropriate malicious traffic wasn't possible.

At Kyoto University some data was collected from honeypot servers (Song et al, 2006 [5]) containing the real network traffic from November 2006 to August 2009 having a total 24 statistical features but since the traffic was received from honeypot servers majority of the data are considered malicious and the attack type is completely unknown.

Shiravi et al, 2012 [7] introduced the ISCX2012 dataset that was created by using two sets of profiles, one was used to attack the system and the other was used. The dataset contains twenty features of DoS and brute force attacks. This dataset contains only HTTP traffic

where all modern applications have moved towards HTTPS nowadays, also, the number of feature selection for SDN specific machine learning training are not enough.

The CICIDS2017 dataset (Sharafaldin et al, 2018 [8]) attack vectors containing more than 80 features but Panigrahi et al, 2018 later showed some limitations to this dataset namely 288602 labels were not found and 203 missing attributes.

The CSE-CIC-IDS2018 dataset was for Amazon Web Services platforms only but this dataset contains some of the similar problems of CICIDS2017 dataset.

A novel dataset was presented by (Elsayed et el, 2020 [14]) that introduced a total of 83 attributes and contained a more up-to-date dataset but the proposed research was made on a virtual environment containing Damn Vulnerable Web Application, a publicly available lab environment rather than on a live domain for more realistic approach. Also, the overall virtual network infrastructure doesn't really simulate real world scenarios.

Sarica et al, 2020 [15] introduced a SDN dataset for IoT devices that contained 27.9 million and 30.2 million data records, respectively.

Khanal et al, 2023 [16] have introduced NITSDN that could attain an accuracy of 99.48% using Decision Tree and this dataset contains the network traffic flow of the SDN environment.

Even though the dataset presented above was used for detection of intrusion, almost each of them were made available decades ago and were created on traditional networking infrastructures. This creates a fundamental problem because traditional network and software defined networks are different in nature so how we should approach the IDS system for each will also be different from each other.

Table 2.2.1: Summary of Literature Reviewed.

SL No	Dataset / Author Name	Year	Realistic Traffic	Label	Network Type	No. of Attributes	Network Environment
1.	KDD'99 [2]	1998	No	Yes	Small Network	41	Traditional Network
2.	NSL-KDD [2]	2009	No	Yes	Small Network	41	Traditional Network

3.	Kyoto [5]	2006-2009	Yes	Yes	Honeypots	24	Traditional Network
4.	ISCX2012 [7]	2012	Yes	Yes	Small Network	Not Known	Traditional Network
5.	CICIDS2017 [8]	2017	Yes	Yes	Small Network	83	Traditional Network
6.	CSE-CIC-IDS2018 [10]	2018	Yes	Yes	Small Network	83	AWS Platform
7.	InSDN [14]	2020	Yes	Yes	Small Network	83	SDN Network
8.	Sarica et al. [15]	2020	Yes	Yes	Small Network (IoT)	33	SDN Network
9.	ISOT Cloud IDS	2021	Yes	Yes	Cloud	130	Cloud Network
10.	Tree-SDN-DDoS Dataset	2024	Yes	Yes	Simulated Network	25	SDN Network

2.2.1 Similar Applications

Several research studies, case studies, and methodological contributions in the field of network security and Software Defined Networking (SDNs) are closely related to this thesis. These includes traditional IDS datasets like KDD'99, NSL-KDD, Kyoto and ISCX2012 datasets are outdated and designed for traditional networking infrastructures which lack relevance in SDN-specific environments.

Modern IDS dataset on the other hand includes CICDS2017, CSE-CIC-IDS2018, and Tree-SDN-DDoS dataset. While more recent, these datasets are either too small, limited to specific network scenarios, or lack key features needed for robust IDS in SDNs.

There are a few cloud specific datasets that has been made available for research like Sarica et al., 2015 [15] (IoT), ISOT cloud IDS etc. Although designed for specialized environments, these datasets do not generalize well to SDN's centralized architecture.

SDN specific dataset has been created to advance the research in this field, examples include Khanal et al. (NITSDN), InSDN dataset by Elsayed et al.. These works represent progress but often lack the scale, realism, or diversity of traffic patterns found in real-world SDN deployments.

2.2.2 Related Research

The provided literature review highlights several gaps in existing research and data resources for Intrusion Detection Systems (IDS) in Software Defined Networks (SDNs). Most publicly available datasets were created decades ago and are not representative of modern network traffic and attack vectors. This significantly hinders the development and evaluation of effective IDS for SDN environments. Existing datasets were primarily designed for traditional network infrastructures and are not suitable for the centralized control architecture of SDNs. This mismatch leads to inaccurate and unreliable detection results when applied to SDN environments. Many datasets lack a comprehensive set of features essential for training machine learning models in the context of SDN security. This limitation restricts the ability of IDS to identify and classify diverse attack types. Several studies rely on data collected from honeypots or virtual environments, which may not accurately reflect real-world network behavior and attack patterns. This lack of real-world applicability reduces the effectiveness of the proposed IDS solutions. Most datasets represent small network scenarios, failing to capture the complexity and diverse traffic patterns of real-world large-scale SDN deployments. This limits the generalizability of findings and the scalability of proposed IDS solutions. Some datasets suffer from unknown class labels for certain data points, making them unusable for training purposes. Additionally, missing attribute values further complicate the analysis and model development process.

2.3 Gap Analysis

Table 2.3.1: Gap Analysis of the Project with existing works

Features	**Traditional Datasets (e.g., KDD'99, NSL-KDD)**	**Modern IDS Datasets (e.g., CICDS2017)**	**IoT/Cloud Datasets (e.g., Sarica, ISOT Cloud)**	**Existing SDN-Specific Datasets (e.g., NITSDN, InSDN)**	**Proposed Dataset**
Up-to-date attack types	No	Partial	Partial	Yes	Yes
Realistic Traffic Patterns	No	Partial	Yes	Partial	Yes
Large-scale network scenerios	No	No	No	Partial	Yes
Specificity to SDN environments	No	No	No	Yes	Yes
Comprehensive feature set for SDNs	No	Partial	Partial	Partial	Yes

Real-world applicability	Partial	Partial	Partial	Partial	Yes
Missing or unknown labels	Yes	Yes	Yes	No	No
Diversity of attack vectors	Partial	Yes	Partial	Partial	Yes
Focus on centralized SDN architecture	No	No	No	Yes	Yes
Data Labelling	No	Partial	Partial	Yes	Yes

2.4 Summary

In conclusion, the current research landscape presents a clear need for up-to-date, comprehensive, and SDN-specific datasets that accurately represent real-world network traffic and attack scenarios. Addressing these gaps is crucial for the development and deployment of robust and effective Intrusion Detection Systems in Software Defined Networks.

Chapter 3
Research Methodology

3.1 Methodology/Requirement Analysis & Design Specification

3.1.1 Overview

This research study explores the dataset creation for automated detection of intrusion in software defined networking systems. It describes the complexity of SDN networks, obtaining a good enough dataset and making it publicly available for further analysis and use by other researchers. It also highlights the importance of information security in large enterprises.

3.1.2 Proposed Methodology/ System Design

The proposed methodology is to use five virtual machines using VirtualBox on Windows or Linux platforms. The first virtual machine is Kali Linux representing the attacker server that will be used to attack the network. The secondary machine is Ubuntu 24.04 and acts as the SDN controller using OpenDayLight. Thirdly, another Ubuntu 24.04 machine will be used to serve a Mininet and OVS switch. Mininet is a network emulation platform that allows researchers, developers, and network engineers to create and test virtual networks on a single machine. It provides a way to simulate a realistic network environment, complete with hosts, switches, links, and controllers, without the need for physical hardware. Another virtual machine will be a Linux based Metasploitable 2 server to provide vulnerable services for demonstrating common vulnerabilities. Lastly, Metasploitable 3 is a windows server which is also providing vulnerable services.

The attacker will attack the network using various tools and techniques on the Kali Linux machine. The attack will go through the live domain and the SDN. We will continuously monitor the network via wireshark and other network monitoring tools and generate the dataset from the network log files.

The setup process of above-mentioned methodology is as follows:

1. Firstly, installing the OVS Switch and Mininet on the same virtual machine.
2. Then we create four adapters in the OVS-VM which represent 5 completely different network subnets.
3. Inside the OpenFlow switch we create 2 OVS bridges and each data plane interface gets assigned to its proper bridge.
4. The IP address from each data plane gets removed and gets assigned to 0.
5. Then we also connect the Kali VM, Metasploitable 2 and Metasploitable 3 with the network.
6. On the OVS Linux Machine we enable IP Forwarding.
7. We create 4 virtual hosts in the Mininet topology.
8. Lastly, we connect all bridges with the ONOS controller and we are able to ping all hosts residing in different subnets

After the initial setup of the overall environment is completed we can move to the phase of dataset generation. For generating the dataset, attackers might try various attacks and utilize different techniques to compromise the network. As now we've shifted to a centralized component it creates new attack vectors for the attacker to try. These attacks are targeted for the SDN controller and are different from the attacks of traditional networking. Besides, applications of SDN can have other vulnerabilities that are not limited to buffer overflows, Cross Site Scripting, SQL Injection etc. These vulnerabilities might serve as an opening for attackers to gain access to the SDN controller or run a malicious script. The below table contains the information of the attack classes and vulnerabilities and the tools that were used to perform them along with the IP addresses.

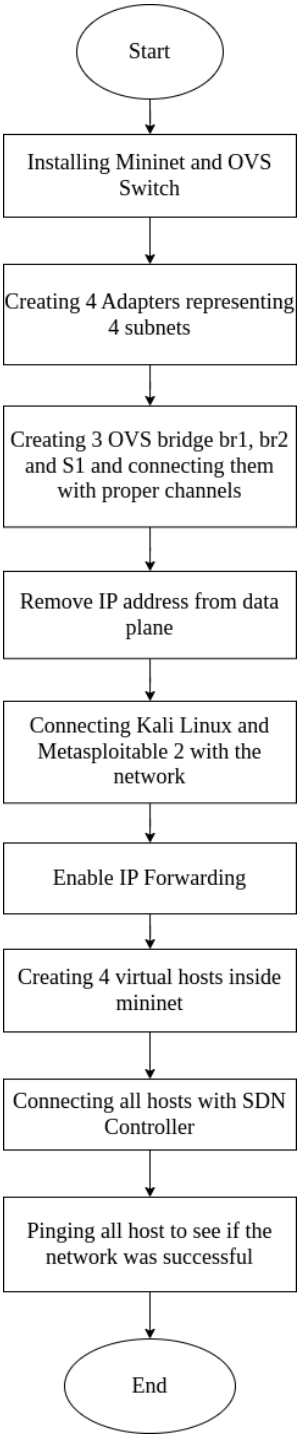


Figure 3.1.2.1: Methodology Flowchart

Table 3.1.2.1: Attack type and used tools overview

Attack Classes	Description of Activities	Attack Tools	Attacker Machine	Victim Network
DoS	TCP-ACK, UDP, HTTP, HTTP POST	LOIC slowhttptest HULK torshammer	Kali-Linux	h4
	Slowris, TCP Flood	Slowloris.py, Nping	Kali-Linux	Metasploitable2
	Slowris, TCP Flood	Slowloris.py, Nping	Kali-Linux	Metasploitable3
DDoS	TCP-SYN Flood UDP Flood ICMP Flood	Hping3	h1, h2	h4, Metasploitable2, Metasploitable3
Web Attacks	OWASP Top 10 (SQLi, XSS, Broken Auth, XXE, Redirects, Insecure Deserialization etc.)	Metasploit Framework, and other tools	Kali-Linux	Juice Shop
R2L	Password Guessing Attack	ffuf	Kali-Linux	Juice Shop
		Metasploit Framework, Hydra	Kali-Linux	Metasploitable2 Metasploitable3
Probe	Version scan, Port scan, Discover services	Nmap	Kali Linux	h1, h2, h3, h4, Metasploitable2, Metasploitable3
	Vulnerability scan (WMAP)	Metasploit Framework	Kali Linux	Metasploitable2
U2R (Exploitation)	Vsftpd, IRCd, Samba, Ingress, distcc	Metasploit Framework	Kali-Linux	Metasploitable2
	Apache Struts, Tomcat, Jenkins, WInRM etc.			Metasploitable3

We monitor the network using a network monitor tool called WireShark. Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is very similar to tcpdump, but has a graphical front-end and integrated sorting and filtering options. Inside the WireShark packet analyzer we can see the OpenFlow protocol network traffic and other necessary information about the said network traffic. Below we can see a picture from wireshark network analysis tool.

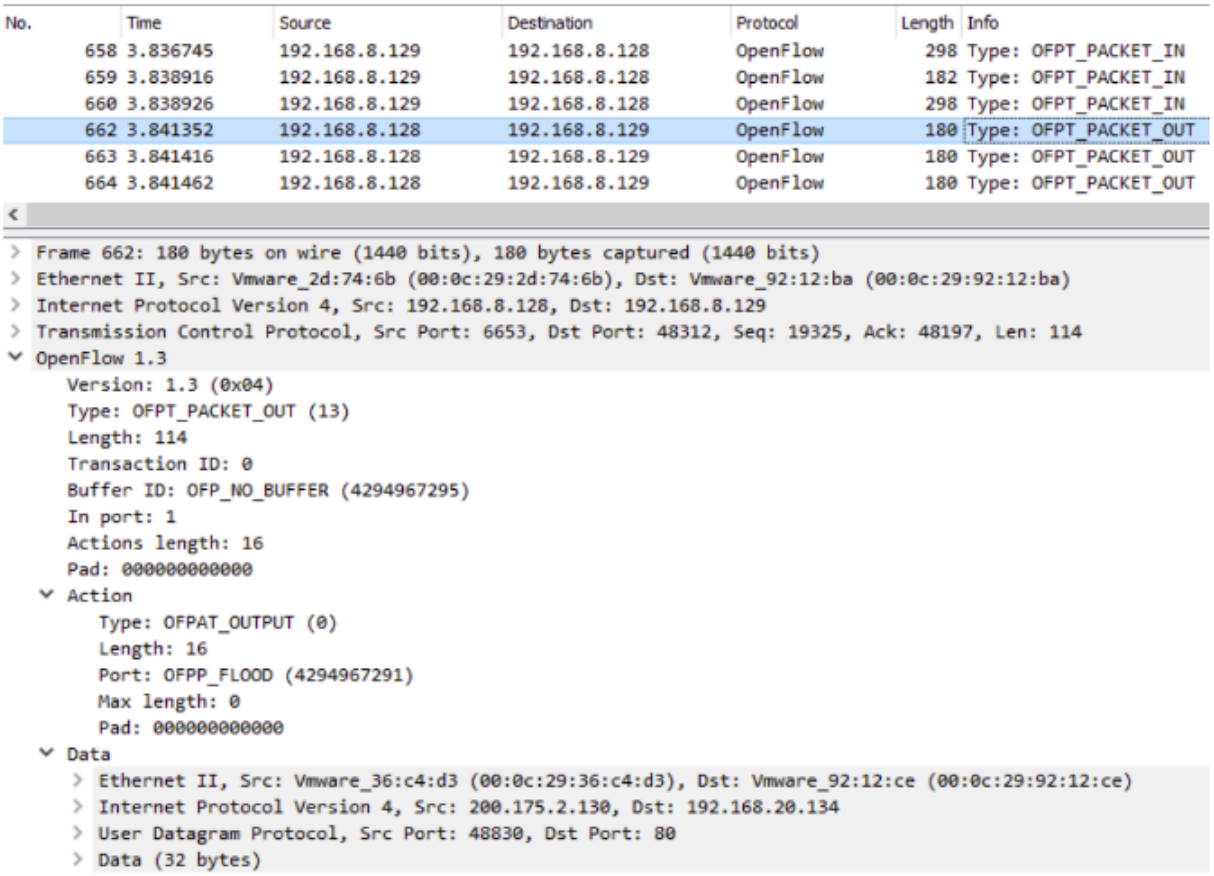


Figure 3.1.2.2: Wireshark OpenFlow Packet Analysis

3.1.3 Functional and Nonfunctional Requirements

Functional Requirements:

1. Dataset generation: Recording and analyzing network logs using tools like Wireshark to create a comprehensive intrusion detection dataset.
2. Environment setup: Simulating a network with multiple subnets using mininet and configuring opendaylight SDN controller.
3. Attack simulation: Executing various attack scenarios (DoS, DDoS, SQL Injection,

etc.) using Kali Linux tools.

4. Traffic monitoring: Monitoring and capturing network traffic in real time using tools like wireshark and openflow logs.
5. Centralized network management: Managing routing and policies centrally via the opendaylight SDN controller.
6. Vulnerability testing: Testing the network's response to vulnerabilities using Metasplioitable 2 & 3.

Non-Functional Requirements:

1. Scalability: Supporting scaling to larger virtual networks with additional hosts and subnets.
2. Reliability: Ensuring consistent and stable communication across all virtual machines.
3. Usability: Providing a user-friendly setup for other researchers to replicate.
4. Performance: Handling high traffic volumes efficiently during attack simulations.
5. Security: Isolating attack scenarios to prevent unintended impacts beyond the virtual network.

3.1.4 Context Diagram

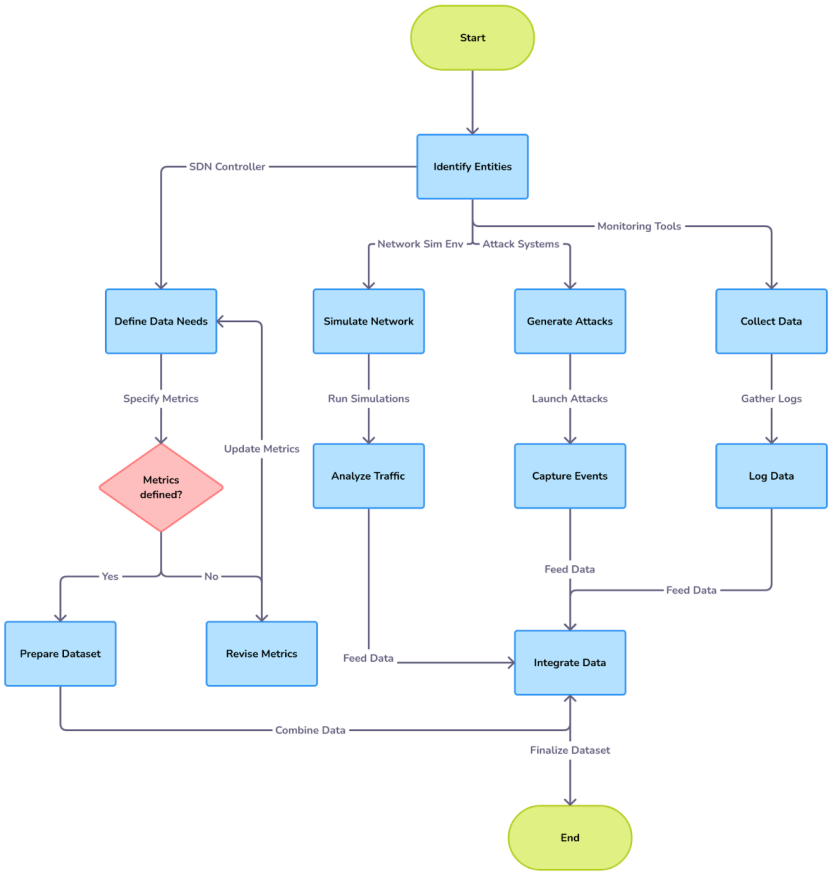


Figure 3.1.4.1: Context Flow Diagram

3.1.5 Data Flow Diagram Level 1

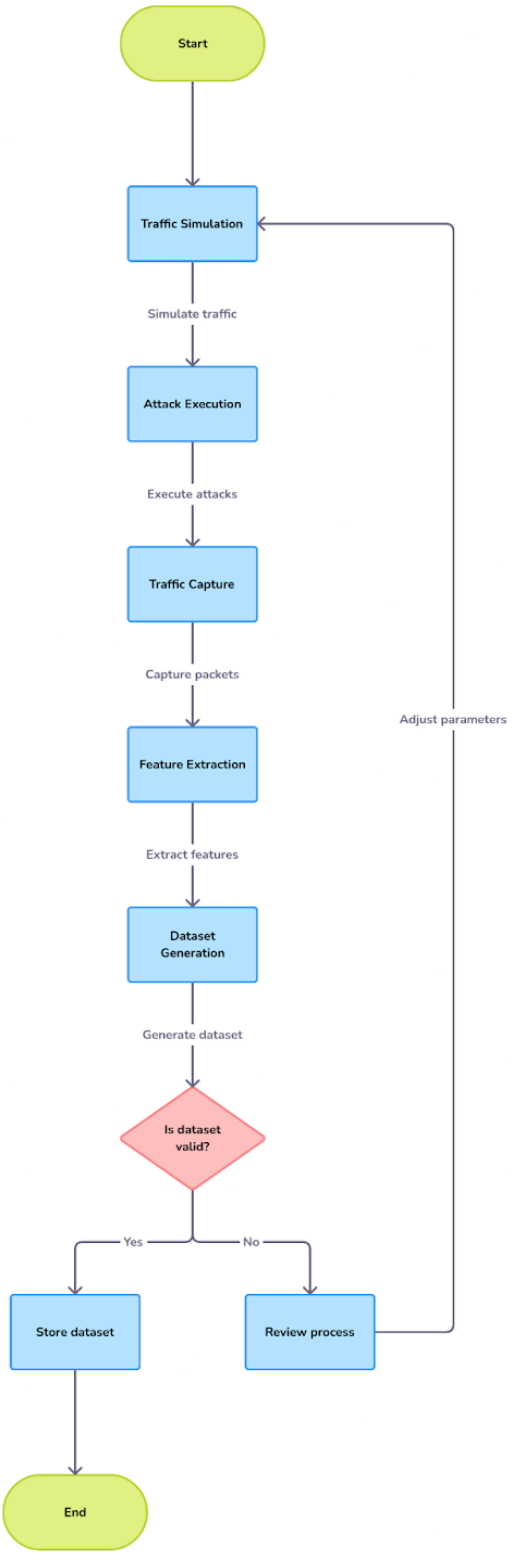


Figure 3.1.5.1: Data Flow Diagram

3.1.6 UI Design

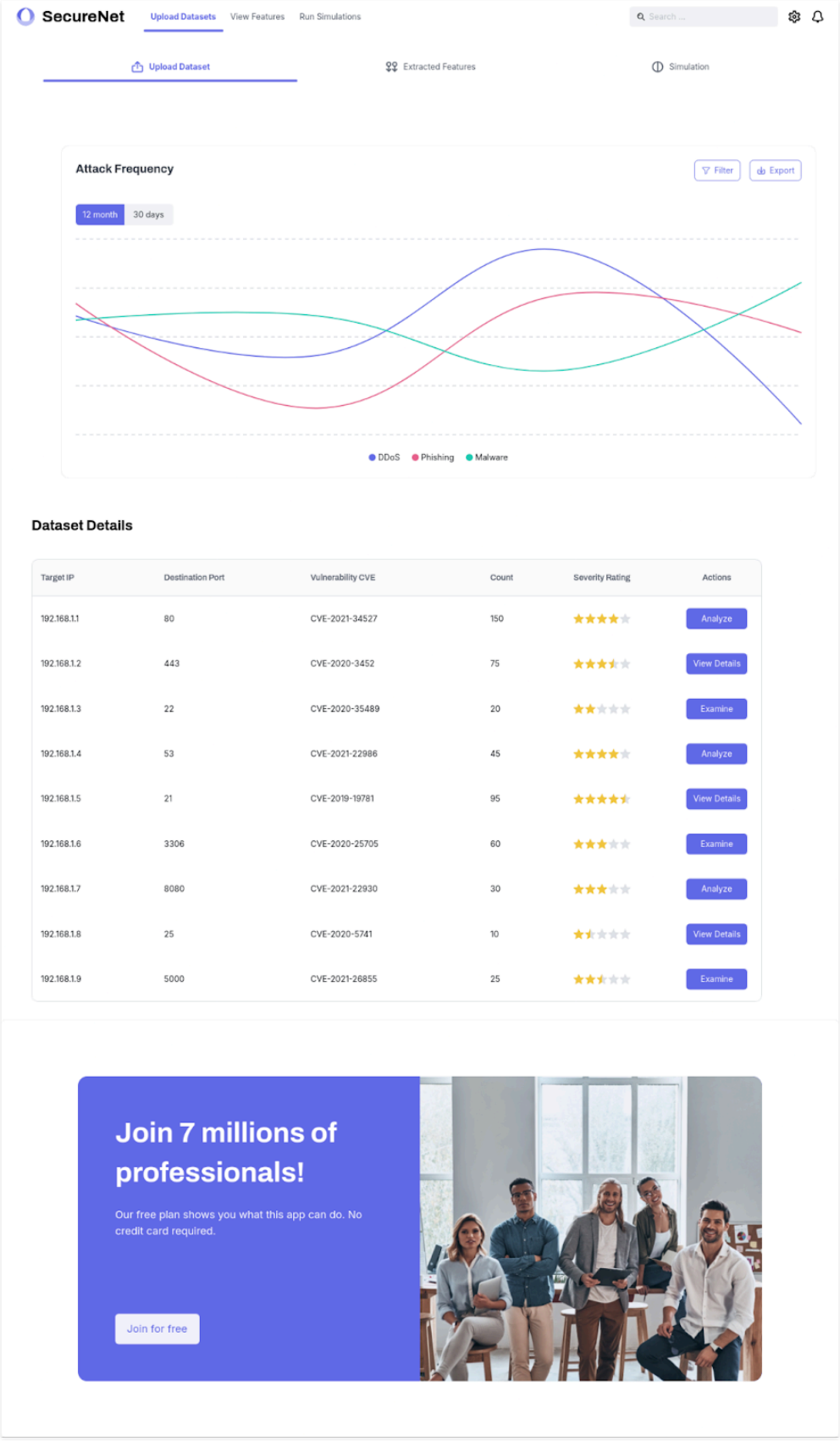


Figure 3.1.6.1: UI Design of the Interface for IDS

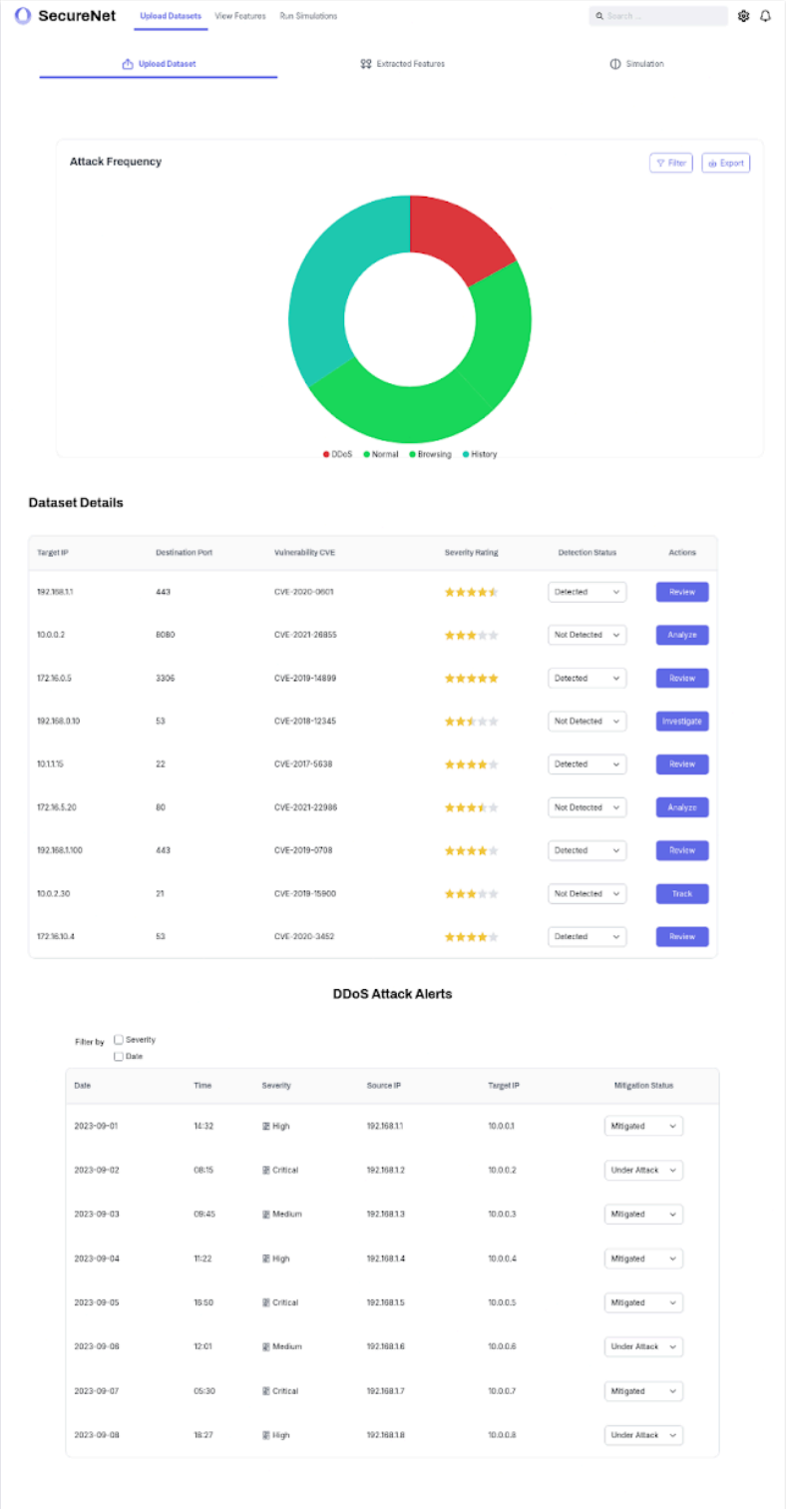


Figure 3.1.6.2: UI Design Showing DDoS Attacks

3.2 Detailed Methodology and Design

The setup involved installing Open vSwitch (OVS) and Mininet on a virtual machine, creating network adapters to simulate multiple subnets, and configuring OVS bridges. We connected a Kali VM, Metasploitable 2 and 3 for attack simulations, enabled IP forwarding, and created

virtual hosts in Mininet. Finally, we linked all components to the OpenDayLight controller, ensuring centralized network management and the ability to monitor and analyze both normal and malicious traffic for intrusion detection.

We already said that this setup involved 5 virtual machines which contain multiple Networking technology. The prototype is mainly divided into 2 categories.

1. Setup Environment which includes setting up the virtual network for testing. Connecting all VMs and setting up SDN controllers with them and ensuring they can talk to each other.
2. Processing the dataset from the logs generated by attacking the network from an attacker point of view.

To set up the environment, we used Virtualbox to run the four virtual machines. Among these we've used 2 Ubuntu 24.04 VM. One for setting up the OpenDayLight SDN controller and the other for setting up Mininet and a vulnerable server called OWASP Juice Shop. The other three machines are Kali Linux, widely used by penetration testers and security professionals for attacking the network and Metasploitable 2 server for running different services like FTP, SSH, Telnet etc and Metasploitable 3 running Microsoft IIS, Jenkins etc. Those five virtual machines were interconnected with the SDN controller.

From the followed methodology,

1. Installation of OVS Switch and Mininet

Open vSwitch (OVS) and Mininet were installed on the same virtual machine (VM) to streamline management and ensure seamless integration between the virtual network switch and the network emulator.

2. Network Adapter Configuration

Five network adapters were created within the OVS-VM, each representing a distinct network subnet. This setup simulates a more complex network environment, allowing for a variety of traffic patterns and network conditions.

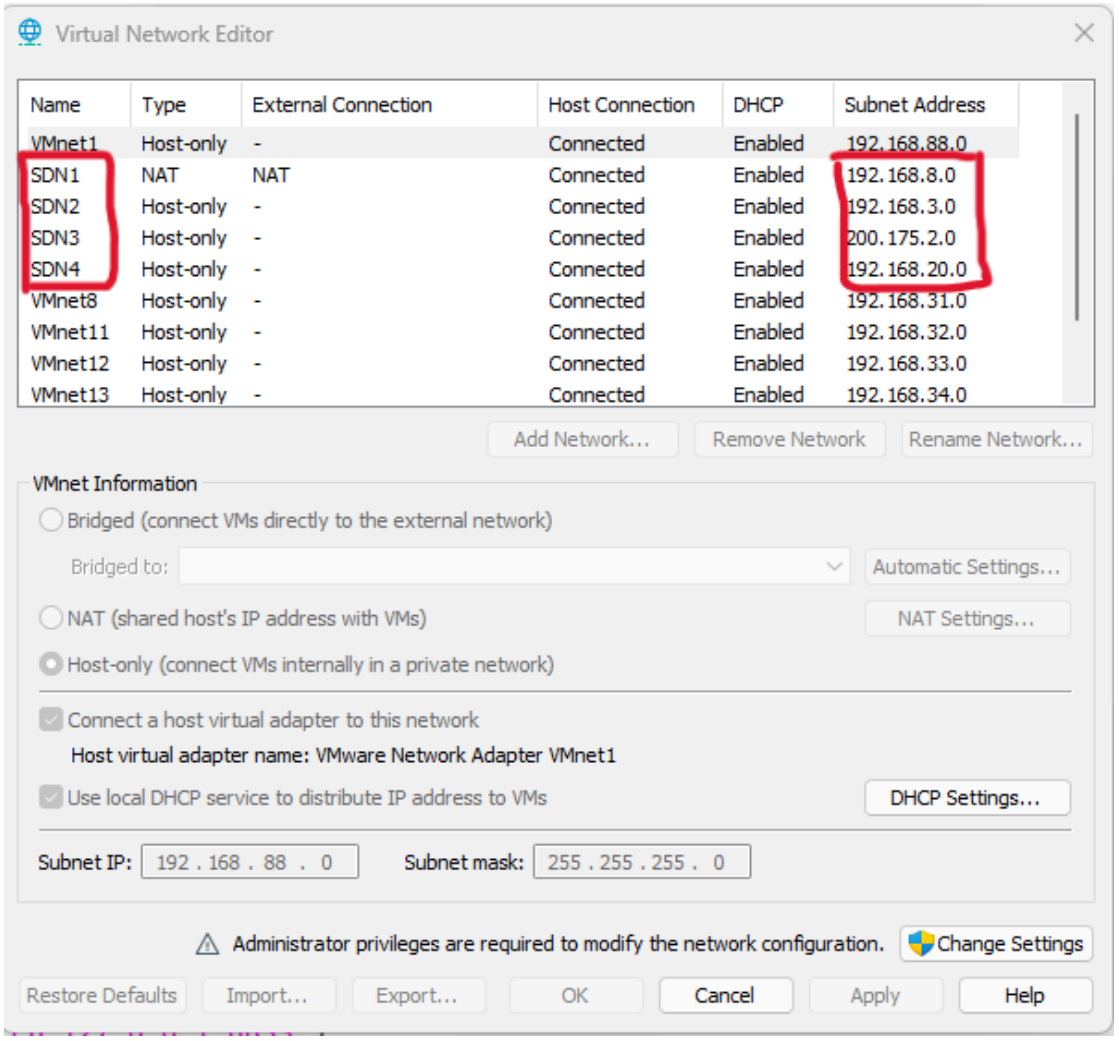


Figure 3.2.1: Subnet config inside VMware

3. Creation of OVS Bridges

Three OVS bridges were established within the OpenFlow switch named br1, br2 and br3. Each data plane interface was assigned to its respective bridge here in this case br1 with ens38 and br2 with ens39 and br3 with enp0s16 ensuring proper segregation of network traffic and enabling the isolation of different subnets. The default bridge S1 was connected with ens39. We created bridge S1 if it wasn't there.

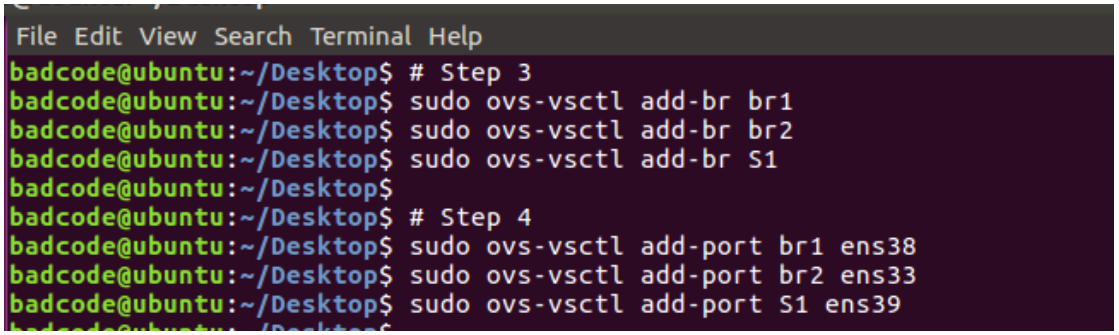


Figure 3.2.2: Adding OVS Bridge

4. IP Address Assignment

IP addresses were removed from each data plane interface and set to 0. This step ensures that routing and traffic management are controlled centrally by the SDN controller (OpenDayLight), rather than traditional IP routing methods.

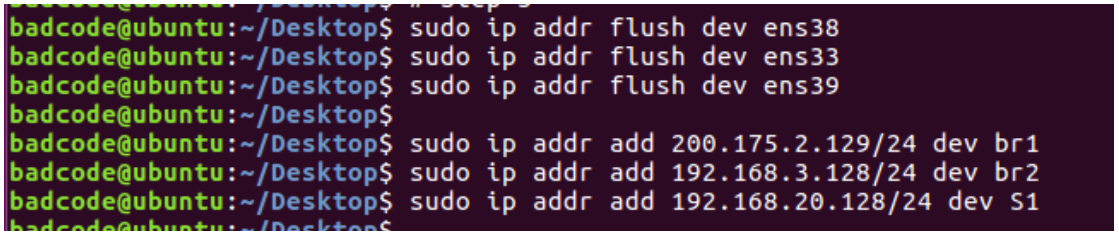


Figure 3.2.3: IP address assignment

5. Integration of Kali VM and Metasploitable 2 & 3

The Kali VM, serving as an attacker, and Metasploitable 2 & 3, acting as a vulnerable target, were connected to the network. This integration allows for the generation of realistic attack scenarios and the collection of malicious traffic for the dataset.

6. Enabling IP Forwarding

IP forwarding was enabled on the OVS Linux machine to facilitate packet routing between different interfaces and subnets. This capability is crucial for maintaining communication across the network segments.

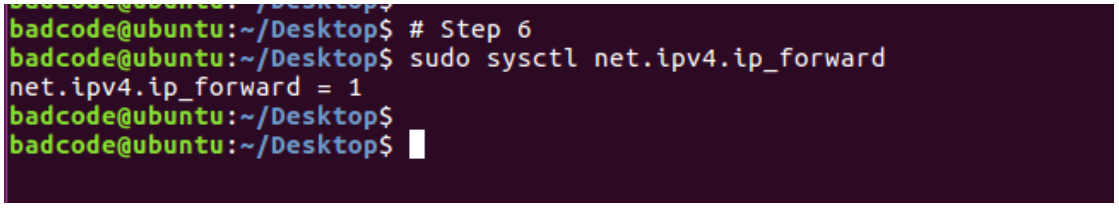


Figure 3.2.4: Enabling Port Forwarding

7. Creation of Virtual Hosts in Mininet

Four virtual hosts were created in the Mininet topology to represent end devices within the network. These hosts were used to simulate typical network traffic and interactions within and across the different subnets.

```
^C
--- 192.168.20.128 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 12998ms
rtt min/avg/max/mdev = 0.019/0.131/1.188/0.298 ms
mininet> net
h1 h1-eth0:S1-eth1
h2 h2-eth0:S1-eth2
h3 h3-eth0:S1-eth3
h4 h4-eth0:S1-eth4
S1 lo:
c0
mininet>
```

Figure 3.2.5: Virtual Host config

8. Installing OpenDayLight Controller

We have a dedicated virtual machine just for the OpenDayLight controller. It is the brain of our overall setup. To install the OpenDayLight we selected Ubuntu OS for an environment friendly setup. There are some prerequisites, among them we ensure that Java(JDK v8) is installed before we jump into OpenDayLight. We also faced some problems during the installation of OpenDayLight. After obtaining the OpenDayLight package, we successfully installed the controller into our virtual machine.

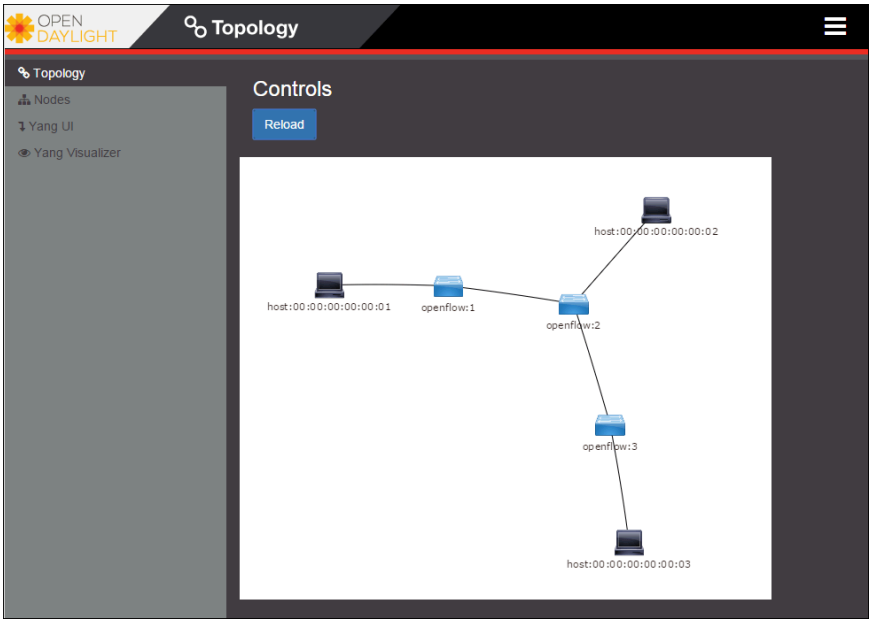


Figure 3.2.6: OpenDayLight on Ubuntu Virtual Machine

8. Connection to ONOS Controller

All OVS bridges that we created on the OVS-Switch VM, were connected to the OpenDayLight controller, for centralizing network control and management. This setup allows for the implementation and monitoring of network policies, routes, and security measures from a single, centralized point.

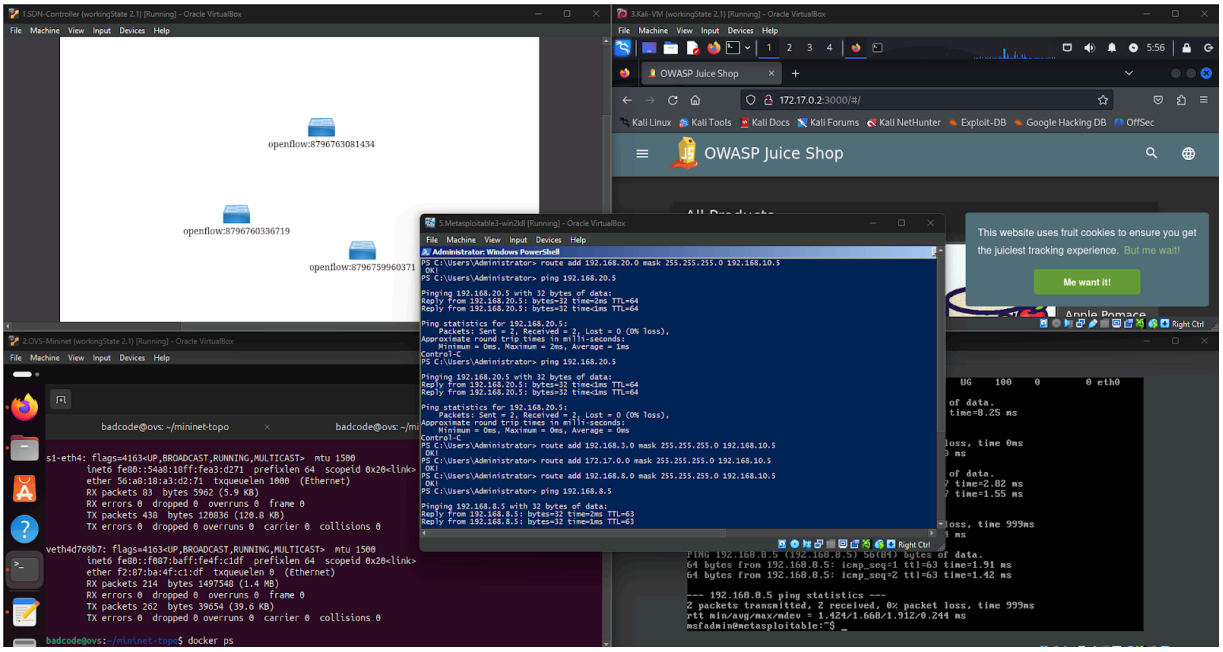


Figure 3.2.7: Complete setup with all 5 virtual machines

3.3 Project Plan

1. Project planning and background research: Defining attack scenarios and dataset requirements and identifying tools and technologies.
2. Designing and implementing the prototype.
3. Environment Setup: Installing and configuring virtual machines. Integrating SDN controller and vulnerable servers.
4. Continuously attacking and monitoring the network: Execute predefined attack scripts while monitoring and capturing network logs using Wireshark.
5. Dataset Creation from generated logs: Processing and formatting captured logs for research. Validating dataset for accuracy and completeness.
6. Documentation and Release of the dataset: Documenting methodology, dataset details, and use cases then publishing dataset for public access.

3.4 Task Allocation

Table 3.4.1: Task Allocation Table

Task Description	**Timeline**
Requirement Analysis	2 weeks
Environment setup	4 weeks
Installing and configure OVS, Mininet and OpenDayLight	3 weeks
Configure and testing network topology	1 week
Developing and executing attack scenarios	4 weeks
Traffic Monitoring and data capture	2 weeks
Dataset generation and validation dataset	2 weeks
Perform system testing of the dataset	1 week
Documentation and publication of findings	1 week

3.5 Summary

This research involves setting up a robust SDN environment, simulating realistic attacks, and creating a publicly available dataset. The methodology employs centralized management via OpenDayLight and uses advanced monitoring tools to analyze network behavior under attack conditions. This setup serves as a valuable resource for researchers aiming to improve intrusion detection in SDN systems.

Chapter 4
Implementation and Results

4.1 Environment Setup

The simulated environment was created by following the below steps:

1. Downloading required files: Ubuntu 24.04 LTS (.iso) file, Kali Linux (.ova) file and metasploitable 2 & 3 server images.
2. Creating and import Virtual Machines: Creating two virtual machines for Ubuntu in VirtualBox and import Kali Linux, Metasploitable 2 & 3.
3. In virtualbox network manager, create 5 NAT network with the below subnet: 192.168.8.0/24, 192.168.3.0/24, 200.175.2.0/24, 192.168.20.0/24, 192.168.10.0/24
4. Then in the settings of each virtual machine set the network type to NAT network and assign appropriate NAT networks for each virtual machine.
    a. SDN Controller VM: 192.168.8.0/24
    b. OVS/Mininet VM: 192.168.8.0/24, 200.175.2.0/24, 192.168.3.0/24, 192.168.10.0/24, 192.168.20.0/24
    c. Kali VM: 200.175.2.0/24
    d. Metasploitable 2 VM: 192.168.3.0/24
    e. Metasploitable 3 VM: 192.168.10.0/24
5. Enable Network Adapters on OVS Switch VM: Add five network adapters on the OVS Switch VM, each connected to a different NAT network. Allow Promiscuous Mode for all adapters.
6. Install SDN Controller: Open the SDN Controller VM and install the OpenDayLight SDN Controller.
7. Install OVS Switch and Mininet: Open the second Ubuntu VM and install Open vSwitch (OVS) and Mininet software.
8. Create Mininet Topology by following the below steps:
9. Create a Mininet topology with four virtual hosts connected to the s1 OVS bridge.
10. Assign the s1 bridge to the 192.168.20.0/24 subnet.
11. Configure the IP address of each host to match the 192.168.20.0/24 subnet.
12. Set the IP address of the s1 bridge as the default gateway for each host.
13. Test connectivity by pinging all hosts from each other.
14. Add Additional OVS Bridges:
15. Create three additional bridges (br1, br2, br3) on the OpenFlow switch.

16. Assign network interfaces to the respective bridges:
    a. enp0s9 → br1
    b. enp0s3 → br2
    c. enp0s16 → br3
    d. enp0s10 → s1
17. Update IP Address Configuration: Remove the IP address from each NIC interface or set it to zero. Assign the removed IP addresses to their respective OVS bridges.
18. Enable IP forwarding on the OVS machine to allow traffic routing between subnets.
19. Link the SDN Controller to all created bridges (br1, br2, br3, and s1).
20. Update the routing table for all hosts with the subnets and gateways. Assign the appropriate gateway for each subnet.
    a. 192.168.8.0/24
    b. 192.168.3.0/24
    c. 200.175.2.0/24
    d. 192.168.20.0/24
    e. 192.168.10.0/24
21. Ensure that all hosts can ping one another to confirm successful configuration.

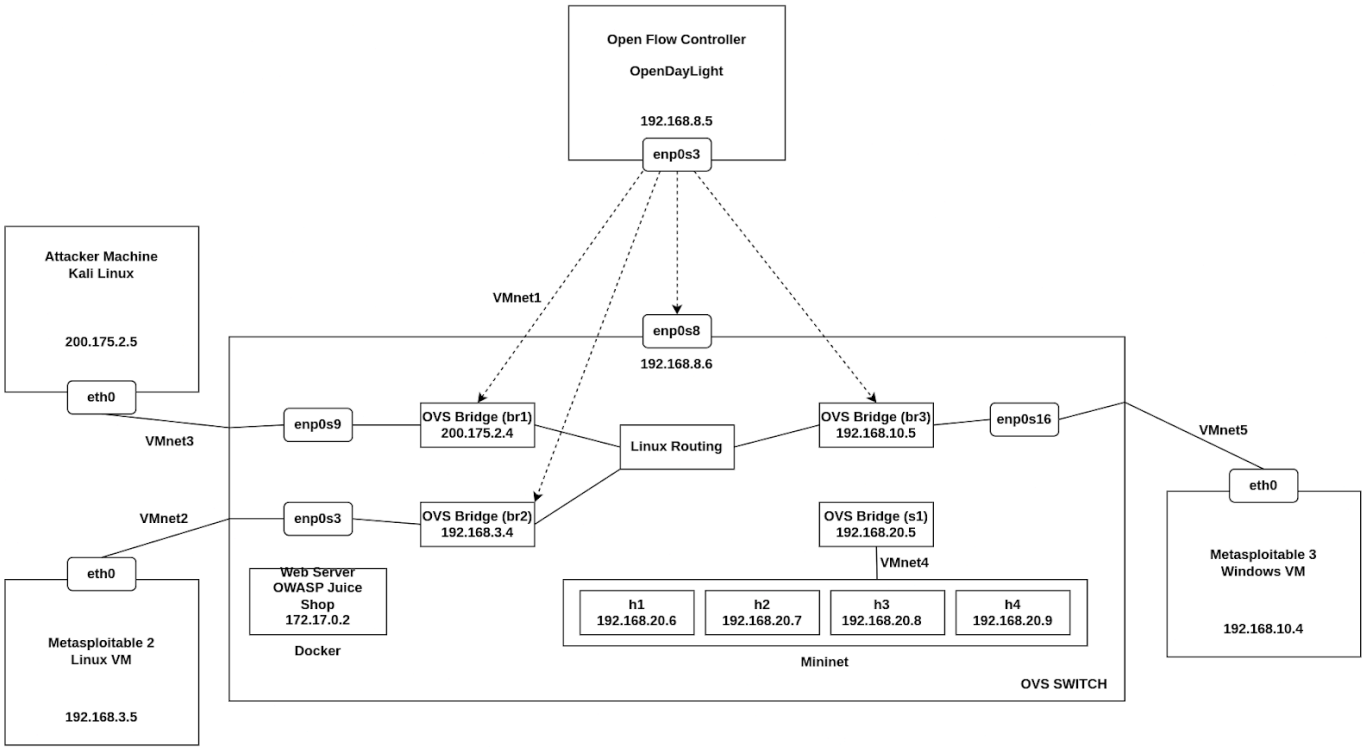


Figure 4.1.1: Architecture of the methodology

4.2 Testing and Evaluation/Performance/ Comparative Analysis

Since precise comparisons cannot be made by using total accuracy, we evaluate the model we propose using the most significant performance metrics, including precision, recall, precision, and F-score. These metrics, which are defined as follows, are frequently utilized in intrusion detection systems:

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

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

$$F - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

The values that are accurately predicted are represented by True Positive (TP) and True Negative (TN). On the other hand, incorrectly identified events are indicated by False Positives (FP) and False Negatives (FN). In addition, we took into account the training time to indicate how long the classifier algorithm needs to train the entire set of data.

It is obvious that all learner classifiers have extremely high overall score metrics for DoS/DDoS and probe classes. This is due to the fact that both the DoS and Probe categories often differ significantly from typical traffic patterns. Additionally, the training time increases as the size of the records increases since it is proportionate to the data records.

4.3 Results and Discussion

The acquired dataset was divided into five major groups based on the network traffic type, attack types and target machine. The first one includes all the data of normal traffic, the second and third one includes data related to Metasploitable 2 & 3 and the fourth one includes attacks on the OVS Switch. Lastly, the fifth category includes SDN controller specific attacks. We've utilized both tcpdump and CICFlowMeter tools to extract the data and packet flow from the network. The total dataset instances are 343,939 for normal and attack traffic. Where the normal data brings a total of 68424, and attack traffic contains 220,2759 instances.

Table 4.3.1: Description of the dataset

Data	Network Traffic	Instance	Total instances / Category	PCAP Size
Normal Data Group	Meet, Social Media, File, Email, Browsing etc.	13179	14280	1.74 GB
Metasploitable 2 - Linux	Denial of Service	1531	511964	482.786 MB
	Distributed Denial of Service	365638		
	Active Recon	138535		
	Password Brute Force	6116		
	User to Root	33		
Metasploitable 3 - Windows	Denial of Service	1571	12440	305.81 MB
	Distributed Denial of Service	12		
	Active Recon	7048		
	Password Brute Force	3788		
	User to Root	21		
OVS/Mininet Machine	Denial of Service	95079	1210187	1.81 GB
	Distributed Denial of Service	847283		
	Active Recon	262211		
	Password Brute Force	2070		
	Attacks on Web Server	3544		
OpenDayLight Controller	Denial of Service	19875	453888	351.4 MB
	Distributed Denial of Service	434013		
Total Dataset			**2202759**	**4.67 GB**

The dataset has a total 8 class that are labelled as:

1. DoS
2. DDoS
3. DrDos
4. Probe
5. Password Guessing
6. Web Attacks
7. User to root (U2R)
8. Normal Traffic

We've collected 84 features from this research which are described below:

Table 4.3.2: List of the collected features

Feature no	Feature Name	Feature No	Feature Name	Feature No	Feature Name
1	Flow ID	29	Fwd IAT Tot	57	ECE Flag Cnt
2	Src IP	30	Fwd IAT Mean	58	Down/Up Ratio
3	Src Port	31	Fwd IAT Std	59	Pkt Size Avg
4	Dst IP	32	Fwd IAT Max	60	Fwd Seg Size Avg
5	Dst Port	33	Fwd IAT Min	61	Bwd Seg Size Avg
6	Protocol	34	Bwd IAT Tot	62	Fwd Byts/b Avg
7	Timestamp	35	Bwd IAT Mean	63	Fwd Pkts/b Avg
8	Flow Duration	36	Bwd IAT Std	64	Fwd Blk Rate Avg
9	Tot Fwd Pkts	37	Bwd IAT Max	65	Bwd Byts/b Avg
10	Tot Bwd Pkts	38	Bwd IAT Min	66	Bwd Pkts/b Avg
11	TotLen Fwd Pkts	39	Fwd PSH Flags	67	Bwd Blk Rate Avg

12	TotLen Bwd Pkts	40	Bwd PSH Flags	68	Subflow Fwd Pkts
13	Fwd Pkt Len	41	Fwd URG Flags	69	Subflow Fwd Byts
14	MaxFwd Pkt Len Min	42	Bwd URG Flags	70	Subflow Bwd Pkts
15	Fwd Pkt Len Mean	43	Fwd Header Len	71	Subflow Bwd Byts
16	Fwd Pkt Len Std	44	Bwd Header Len	72	Init Fwd Win Byts
17	Bwd Pkt Len Max	45	Fwd Pkts/s	73	Init Bwd Win Byts
18	Bwd Pkt Len Min	46	Bwd Pkts/s	74	Fwd Act Data Pkts
19	Bwd Pkt Len Mean	47	Pkt Len Min	75	Fwd Seg Size Min
20	Bwd Pkt Len Std	48	Pkt Len Max	76	Active Mean
21	Flow Byts/s	49	Pkt Len Mean	77	Active Std
22	Flow Pkts/s	50	Pkt Len Std	78	Active Max
23	Flow IAT Mean	51	Pkt Len Var	79	Active Min
24	Flow IAT Std	52	FIN Flag Cnt	80	Idle Mean
25	Flow IAT Max	53	SYN Flag Cnt	81	Idle Std
26	Flow IAT Min	54	RST Flag Cnt	82	Idle Max
27	Fwd IAT Tot	55	PSH Flag Cnt	83	Idle Min
28	Fwd IAT Mean	56	ACK Flag Cnt	84	Label

After the dataset was collected we did some experimental evaluation using 6 supervised learning techniques to evaluate the usability and quality of our dataset. Primarily, our focus is to demonstrate the quality of the acquired dataset when it is used for classification of different attack types. We've mainly used precision, recall, f-score and support to evaluate the efficiency of the performance in our used machine learning models.

Firstly, we preprocess the data and drop the columns containing socket information such as Source IP, Destination IP, flow ID, etc. The final dataset includes 77 features after we drop the unnecessary features. We standardize the values of different features between 0 and 1. We also used a label encoder for multi class encoding scheme on our dataset.

We've used 6 common supervised learning algorithms to evaluate our dataset. Specifically, Logistic Regression, Random Forest, Support Vector Machine, K-Nearest Neighbour, Decision Tree and Naive Bayes. The training and testing data was splitted into a 80:20 ratio and we assumed the value of K=5 for the evaluation. The below table shows the performance of different classifiers using a fully-featured version of our dataset.

Table 4.3.3: Metrics performance for the dataset using various ML classifiers

Attack Name	Matrices	Algorithm					
		LR	RF	SVM	KNN	DT	NB
DoS	Precision	0.6541	0.9976	0.8263	0.9963	0.9977	0.9793
	Recall	0.5713	0.9971	0.9951	0.9962	0.9966	0.2719
	F1 - Score	0.6099	0.9974	0.9029	0.9962	0.9971	0.4257
	Support	23463.0	23463.0	2271.2	23463.0	23463.0	23463.0
DDoS	Precision	0.9922	1.0000	0.9996	0.9922	0.9999	0.8268
	Recall	0.9999	0.9999	0.9997	0.9999	0.9999	0.9974
	F1 - Score	0.9960	0.9999	0.9997	0.9999	0.9999	0.9041
	Support	329824.0	329824.0	2521.6420	329824.0	329824.0	329824.0
DrDoS	Precision	0.6920	0.9980	0.9996	0.9959	0.9977	0.3055
	Recall	0.6005	0.9982	0.9997	0.9936	0.9988	0.9357
	F1 - Score	0.6430	0.9981	0.9650	0.9947	0.9983	0.4606
	Support	11199.0	11199.0	11199.0	11199.0	11199.0	11199.0
Password Guessing	Precision	0.5055	0.9885	0.9650	0.9588	0.9829	0.4799
	Recall	0.1509	0.8533	0.2548	0.8533	0.8528	0.4040
	F1 - Score	0.2324	0.9159	0.4032	0.9029	0.9133	0.4387
	Support	2426.0	2426.0	106.2	2426.0	2426.0	2426.0
Web Attacks	Precision	0.9454	0.9772	1.0000	0.9707	0.9727	0.9062
	Recall	0.3247	0.9293	0.0313	0.9091	0.9264	0.4040
	F1 - Score	0.4834	0.9527	0.0606	0.9389	0.9490	0.5589
	Support	693.0	693.0	10.6	693.0	693.0	693.0

Probe	Precision	0.9231	0.9960	0.9231	0.9957	0.9960	0.8714
	Recall	0.9703	0.9997	0.9991	0.9991	0.9997	0.2013
	F1 - Score	0.9461	0.9978	0.9596	0.9974	0.9978	0.3270
	Support	94819.0	94819.0	8244.8	94819.0	94819.0	94819.0
U2R	Precision	0.0000	0.6667	0.0000	0.6000	0.4546	0.0011
	Recall	0.0000	0.3636	0.0000	0.2727	0.4546	0.1818
	F1 - Score	0.0000	0.4706	0.0000	0.3750	0.4546	0.0021
	Support	11.000	11.000	7.784	11.000	11.000	11.000
Normal	Precision	0.7832	0.9753	0.9731	0.9794	0.9817	0.5582
	Recall	0.5603	0.9877	0.9576	0.9873	0.9807	0.0893
	F1 - Score	0.6533	0.9815	0.9801	0.9834	0.9812	0.1540
	Support	2843.0	2843.0	2843.0	2843.0	2843.0	2843.0
Merged (all types)	Precision	0.9545	0.9987	0.9631	0.9984	0.9987	0.7875
	Recall	0.9545	0.9987	0.9976	0.9984	0.9987	0.7875
	F1 - Score	0.9545	0.9987	0.9801	0.9984	0.9987	0.7875
	Support	0.9545	0.9987	25837.9	0.9984	0.9987	0.7875

The performance metrics for linear classifiers indicate that the overall scores are significantly higher for DoS/DDoS and Probe attack classes, while U2R consistently demonstrates poor results. This discrepancy arises because the DoS and Probe categories exhibit patterns that are distinctly different from normal traffic, whereas U2R attacks closely resemble normal connections. Additionally, the U2R flow records are much smaller in size compared to the normal flow within the same dataset. Both recall and F1-scores for the Web Attack and U2R classes are notably low across linear classifiers and SVM. On the other hand, classifiers such as KNN, Decision Tree (DT), and Random Forest (RF) achieve reasonable training times and good performance for most attack classes. However, they also struggle with lower scores for the U2R attack class. The Naive Bayes (NB) classifier, while efficient in terms of training and prediction time, performs poorly on three attack types: Brute Force, Web Attack, and U2R. An important observation is that strong performance metrics on the merged dataset can overshadow poor results for less frequent attack classes, as the dataset is dominated by samples from DoS/DDoS and Probe attacks. Moreover, training time scales with the size of the dataset, with larger datasets resulting in longer training durations. Among the classifiers, SVM had the longest training time for DoS/DDoS and Probe attacks.

Result Comparison with other published works:

Table 4.3.4: Result comparison with other published research

Reference	Used ML Algorithms	Dataset(s)	Features used	SDN Environment	Accuracy
Chen et al., 2024	XGBoost and two-stage collaborative classifier	CICIDS2018, NDSec-1	76, 63	No	99.93 % (CICIDS 2018 dataset) and 89.13 % (NDSec-1 dataset)
Alashhab et al., 2024	Ensemble Model (BernoulliNB, Passive-Aggressive, SGD Classifier and MLP Classifier)	Trained on Private Dataset. Tested on InSDN, CICDDoS2019, slow-read-DDoS	22	Yes	99.20%
Garba et al., 2024	SVM, LR, DT, KNN	Private Dataset.	19	Yes	DT: 99%
Jaraba et al., 2024	Not Applicable. Introduced and analyzed 3 scenarios to test DDoS attacks mitigation.	Not applicable.	Not Applicable.	Yes.	Not Applicable. Analyzed 3 scenarios and how they affect the mitigation of DDoS.
Mzibri et al., 2023	DT, RF, Adaboost	InSDN	78	Yes	99.80%

Towhid et al., 2023	RF, XGBoost	InSDN	Features used: 9	Yes	99.69%
Qiu et al., 2023	Modified Golden Jackal Optimization (mGJO)	Four UCI Dataset, NSL-KDD, InSDN	48	Yes (InSDN)	99.04%
AbdulRaheem et al., 2023	XGBoost	DDoS attack SDN Dataset	23	Yes	97.20%
Liu et al., 2023	RF, SVM, XGBoost, DT, KNN	CSE-CIC-IDS 2018	26	No	RF: 99.13%
Proposed Research	LR, RF, SVM, KNN, DT, NB	Private Dataset	77	Yes	DT: 99.87 %

4.4 Summary

This chapter has provided a detailed analysis of the performance and comparative analysis on the dataset. The findings underscore the importance of choosing the right classifier based on the specific requirements of accuracy, training time, and attack type coverage. This also outlines the dataset description in detail and the features that were extracted from the network pcap file that was generated.

Chapter 5
Engineering Standards and Design Challenges

5.1 Compliance with the Standards

Creating a dataset for detecting intrusions in Software-Defined Networking (SDN) environments includes the use of specific software tools, hardware configurations and communication protocols that are critical to the functionality of SDN systems, network simulation, and intrusion detection. Below is a discussion of the relevant standards, along with their alternatives, pros, cons and rationale for selection.

5.1.1 Software Standards

SDN Controller: OpenDaylight (ODL) is a widely used open-source SDN controller platform that adheres to SDN standards such as OpenFlow and REST APIs for communication between the control plane and data plane. Alternatively to ODL, Open Networking Operating System (ONOS) is also a popular choice but is more complex to set up and maintain. OpenDaylight was selected because of its broader adoption and extensive documentation and support.

Network Simulation Software: Mininet is selected as the network simulation tool. It is compliant with SDN standards like OpenFlow, which allows for accurate emulation of SDN environments. Alternatively, GNS3 (Graphical Network Simulator-3 supports real network devices and configurations, more realistic simulations.

Wireshark and CICFlowMeter: Wireshark, an industry-standard network protocol analyzer, is used for capturing network traffic, while CICFlowMeter converts captured PCAP data into flow-based features in CSV format, which complies with flow-based traffic analysis standards. By using these widely adopted tools, this project ensures high-quality data collection and feature extraction that meets the requirements for effective network traffic monitoring and analysis.

5.1.2 Hardware Standards

In this project, the hardware standards pertain to the physical devices or virtual environments used to create and simulate the SDN network, capture traffic, execute attacks.

Virtual Machines (VMs) and Containers: The project uses Ubuntu 24.04 VMs for the Mininet, OpenDaylight controller, and attack platforms (Kali Linux, Metasploitable). Alternatively, Containers can be used for simulating SDN switches or even entire network services. May lack some of the flexibility and configurability that full VMs offer for complex network topologies. VMs were chosen because they provide full network isolation and are easier to configure for testing complex network setups with tools like Mininet, OpenDaylight, and Kali Linux. Containers could be considered for lightweight deployments, but VMs are more suited for the project's complexity.

Network Interfaces and Capture Hardware: Network interfaces for capturing traffic from virtual environments or physical NICs for real-world testing. USB Ethernet Adapters are useful for connecting physical devices to virtual environments. For the scope of this project, virtual interfaces provided by the VMs (such as veth pairs in Mininet) are sufficient. Hardware interfaces are not necessary unless expanding to real-world attack simulations.

5.1.3 Communication Standards

SDN Protocol: OpenFlow is the communication protocol between the SDN controller (OpenDaylight) and the data plane devices (OVS switches in Mininet). OpenFlow was selected because it is the most widely used protocol for SDN control and is specifically designed for managing flow tables in SDN switches, which fits the project's requirements perfectly. OpenFlow is more straightforward to implement in this project, as it is already supported by Mininet, OpenDaylight, and OVS.

Network Traffic Capture Protocols: PCAP (Packet Capture) is used for capturing raw network traffic in Wireshark. Limited detail compared to PCAP, lacks packet-level inspection. Since the project focuses on analyzing detailed packet-level data to detect intrusions, PCAP is the more appropriate choice for traffic capture. PCAP is preferred for capturing traffic at a fine-grained level necessary for creating the flow-based features and generating the dataset.

5.2 Impact on Society, Environment and Sustainability

5.2.1 Impact on Life

The creation of a dataset for detecting intrusion in SDN (Software-Defined Networking) systems holds significant potential to enhance cybersecurity measures and protect critical

infrastructures. By enabling more accurate and timely detection of intrusions, this dataset can contribute to the safety and security of individuals and organizations.

Enhanced Cybersecurity: Improved intrusion detection can prevent data breaches, safeguarding personal information and sensitive data from malicious actors.

Protection of Critical Services: Ensuring the integrity of networks that support essential services (e.g., healthcare, finance, utilities) helps maintain their availability and reliability, directly impacting the quality of life.

Increased Trust: As security measures improve, trust in digital systems grows, encouraging broader adoption of technology and digital services.

5.2.2 Impact on Society & Environment

The societal and environmental impacts of developing an intrusion detection dataset for SDN systems are multifaceted, offering both direct and indirect benefits.

Societal Impact may include the following,

1. Economic Stability: Enhanced network security reduces the risk of economic losses from cyberattacks, supporting economic stability and growth.
2. Public Safety: Protecting public infrastructure from cyber threats ensures the continued operation of essential services, contributing to public safety.
3. Innovation and Education: Availability of a robust dataset fosters innovation in cybersecurity solutions and supports educational initiatives in network security.

Environmental Impact may include the following,

1. Reduced Resource Consumption: Improved security reduces the need for reactive measures and system repairs, leading to more efficient use of resources.
2. Sustainable Development: By protecting critical infrastructure, we support sustainable development goals that depend on reliable digital services.

5.2.3 Ethical Aspects

Developing and utilizing a dataset for detecting intrusion in SDN systems raises several ethical considerations that must be addressed to ensure responsible use and development.

Privacy Concerns: It is crucial to ensure that the dataset does not contain sensitive or personally identifiable information that could compromise user privacy.

Bias and Fairness: The dataset must be representative and free from biases that could lead to unfair treatment of certain groups or individuals in intrusion detection practices.

Transparency and Accountability: Clear documentation and transparency about the dataset's creation, usage, and limitations are essential to maintain accountability and trust.

5.2.4 Sustainability Plan

To ensure the long-term viability and effectiveness of the intrusion detection dataset, a comprehensive sustainability plan is essential.

Continuous Updating: Regular updates to the dataset are necessary to keep pace with evolving threats and ensure its continued relevance and accuracy.
Community Engagement: Engaging with the cybersecurity community for feedback, contributions, and collaboration helps enhance the dataset's quality and applicability.
Open Access and Collaboration: Providing open access to the dataset encourages widespread use and collaboration, fostering innovation and improvement in intrusion detection technologies.
Funding and Resources: Securing ongoing funding and resources to support maintenance, updates, and enhancements is vital for the dataset's sustainability.

5.3 Project Management and Financial Analysis

The SDN or Software Defined Network relies on the Software. Of course We need hardware to run the software but to build this project we needed four virtual machines. Two for setupping the Software Defined Network and one for the attack machine, and another for monitoring the network to create logs. To run all the VM in a single computer, we needed a highly configured computer. We also spend money on an e-learning platform to cover our knowledge about the project. We also spend money on software where we will get the best desired output.

Table 5.3.1: Estimated Cost for this project

SN	Components	Estimated Cost (BDT)
01	Infrastructure (Upgrading Hardware)	80,000
02	Domain and Hosting	10,000
03	Software	22,000
04	Training for Learning New Technology	10,000
05	Communication	4,000
06	Transportation	2,000
07	Cloud Server	5,000
08	Miscellaneous	10,000
	Total Estimated Cost:	1,43,000

5.4 Complex Engineering Problem

5.4.1 Complex Problem Solving

This section maps the alignment of this project with complex problem solving, knowledge profiles and engineering activities below:

Table 5.4.1.1: Mapping with complex problem solving.

EP1 Depth of Knowledge	EP2 Range Of Conflicting Requirements	EP3 Depth of Analysis	EP4 Familiarity of Issues	EP5 Extent of Applicable Codes	EP6 Extent Of Stake-holder Involvement	EP7 Interdep endence
✓	✓	✓	✓	✓	✓	✓

This project ensures EP1: Depth of knowledge by focusing on SDN-specific dataset generation, advanced feature engineering, and robust analysis for IDS research. Requires expertise in SDN, networking, intrusion detection systems (IDS), and feature engineering. By ensuring a practical SDN specific dataset creation process that balances data accuracy, diversity, and resource efficiency this project also ensures EP2: Range of Conflicting Requirements. This project also includes detailed evaluation of traffic features, attack vectors and dataset structure. Conducting in-depth experimentation and feature extraction, ensuring

comprehensive IDS testing and benchmarking. EP4: Familiarity of Issues is availed by addressing known challenges like lack of SDN-specific datasets and realistic traffic capture leveraging existing tools and techniques while adapting them for SDN-specific environments. EP5: Extent of Applicable Codes is utilized through standard tools like Wireshark, CICFlowMeter, and OpenDayLight and similar established tools and protocols for network traffic capture and SDN simulation. This project also involves researchers, practitioners and network administrators for dataset evaluation and benchmarking. Improves practical applicability and usability of the dataset in real world IDS development that ensures EP6: Extent of Stakeholder Involvement. Lastly, the EP7: Interdependence integrates SDN components (controller, switches, hosts) and tools (Wireshark, CICFlowMeter) ensuring smooth data collection, feature extraction, and dataset generation in a highly interconnected SDN environment.

Mapping with Knowledge Profile for EP1

Table 5.4.1.2: Mapping with knowledge Profile.

K3 Engineering Fundamentals	K4 Specialist Knowledge	K5 Engineering Design	K6 Engineering Practice	K8 Research Literature
✓	✓	✓	✓	✓

This project maps K3 by focusing on fundamental networking and feature engineering concepts including traffic analysis. It also requires K4: Specialist knowledge by using advanced methods tailored to SDN specific IDS. Expertise in SDN based network simulation, attack generation and feature processing. K5: Engineering design is available because we've designed a dataset with realistic SDN traffic and diverse attack scenarios which balances diversity and real-world applicability in dataset design. Practical use of established tools for SDN simulation and traffic analysis and applications like OpenDayLight, Mininet, Wireshark, VirtualBox, Ubuntu maps to the use of K6: Engineering practice. Lastly, K8: Research Literature is also addressed by finding research gaps in SDN dataset and investigating SDN security trends, feature engineering and intrusion detection

5.4.2 Engineering Activities

Table 5.4.1.3: Mapping with complex engineering activities.

EA1 Range of resources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for society and environment	EA5 Familiarity
✓	✓	✓	✓	✓

For EA1 this project utilized tools like OpenDayLight, Mininet, Wireshark, CICFlowMeter, Ubuntu OS, linux and windows server. Leverages diverse tools and resources for realistic traffic simulation and attack scenario creation. EA2: Level of Interaction is ensured by the interaction between SDN controller, switches, hosts, and attack tools. High level integration of components for seamless traffic generation and dataset collection. EA3: Innovation is also checked by generating an updated SDN specific dataset with new attack scenarios and diverse features. This project thus addresses the lack of real world SDN datasets with novel traffic and attack generation techniques. Improving the SDN security and enabling better intrusion detection system this project also contributes to reducing cyberattacks on critical SDN infrastructures, ensuring safer networking thus EA4: Consequences for Society and Environment was also availed. Lastly, EA5 was tackled by using familiar tools and using them with advanced SDN integration for dataset creation. Builds on established tools while introducing complexity with SDN-specific scenarios.

5.5 Summary

This project develops an updated SDN-specific intrusion detection dataset using tools like OpenDaylight, Mininet, and Wireshark. It adheres to industry standards, tackles real-world challenges, and promotes cybersecurity innovation. The dataset's design ensures practicality, ethical considerations, and sustainability, contributing to safer and more efficient SDN environments.

Chapter 6
Conclusion

6.1 Summary

This research has successfully developed a comprehensive dataset for detecting intrusions in Software-Defined Networking (SDN) systems. The dataset aims to enhance cybersecurity measures by providing a reliable resource for testing and improving intrusion detection algorithms. Key conclusions from this research includes, firstly, dataset Creation. We have curated a robust dataset that includes a wide range of normal and malicious network activities, enabling the development and testing of advanced intrusion detection systems (IDS). The dataset has shown promise in improving the accuracy and efficiency of IDS, allowing for timely and precise detection of potential intrusions. By focusing on SDN systems, the dataset addresses a critical area in modern networking, providing valuable insights and tools for securing these flexible and dynamic environments. The dataset serves as a foundational resource for future research, fostering further advancements in the field of network security and intrusion detection.

6.2 Limitation

Despite the achievements of this research, there are certain limitations and potential conflicts of interest that need to be acknowledged:

The dataset, while comprehensive, may not cover all possible intrusion scenarios and might need continuous updates to stay relevant. The data was generated in a controlled simulation environment, which might not capture all the nuances and complexities of real-world network traffic. Here only the OpenDayLight SDN controller was used. This research does not account for security analyses of other SDN controllers, which could have different security models and countermeasures as suggested by prior studies. This limitation means our findings may not be fully generalizable across all SDN environments. Limited computational resources may impact the ability to process and analyze large volumes of data, potentially affecting the results.

By acknowledging these limitations, this research opens avenues for further work to enhance the robustness, applicability, and accuracy of SDN intrusion detection methodologies.

6.3 Future Work

While this research has made significant strides, there are several areas where further work is recommended. Continuously updating and expanding the dataset to include new types of attacks and evolving threat patterns will enhance its applicability and usefulness. Deploying the dataset in real-world scenarios and environments to validate its effectiveness and refine it based on practical insights. Exploring and developing advanced machine learning and AI techniques to improve the accuracy and speed of intrusion detection. Investigating the applicability of the dataset and detection techniques across different domains and industries to ensure broader relevance and impact. Encouraging collaborative efforts among researchers, industry practitioners, and academic institutions to foster innovation and share knowledge,

References

[1] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," in Proc. IEEE 3rd Int. Conf. Comput., Commun. Secur. (ICCCS), Oct. 2018, pp. 1–8.

[2] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl., Jul. 2009, pp. 1–6.

[3] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM), Oct. 2016, pp. 258–263.

[4] T. A. Tang, L. Mhamdi, D. McLernon, and M. Zaidi, "Deep recurrent neural network for intrusion detection in sdn-based networks," in Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft), 2018, pp. 202–206.

[5] J. Song, H. Takakura, and Y. Okabe. (2006). Description of Kyoto University Benchmark Data. Accessed: Mar. 15, 2016. [Online]. Available: http://www.takakura.com/Kyoto_data/BenchmarkData-Descriptionv5.pdf

[6] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," J. Netw. Comput. Appl., vol. 87, pp. 185–192, Jun. 2017.

[7] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," Comput. Secur., vol. 31, no. 3, pp. 357–374, May 2012.

[8] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. ICISSP, 2018, pp. 108–116.

[9] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," Future Gener. Comput. Syst., vol. 100, pp. 779–796, Nov. 2019.

[10] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," Int. J. Eng. Technol., vol. 7, no. 3.24, pp. 479–482, 2018.

[11] Canadian Institute of Cybersecurity. (2018). CSE-CIC-IDS2018. Accessed: Feb. 10, 2020. [Online]. Available: https://www.unb.ca/ cic/datasets/ids2018.html

[12] J. Wang and I. C. Paschalidis, “Botnet detection based on anomaly and community detection,” IEEE Trans. Control Netw. Syst., vol. 4, no. 2, pp. 392–404, Jun. 2017.

[13] Khanal, B., Kumar, C., Ansari, M.S.A. (2023). NITSDN: Development of SDN Dataset for ML-Based Intrusion Detection System. In: Borah, S., Gandhi, T.K., Piuri, V. (eds) Advanced Computational and Communication Paradigms . ICACCP 2023. Lecture Notes in Networks and Systems, vol 535. Springer, Singapore. https://doi.org/10.1007/978-981-99-4284-8_8

[14] Elsayed, Mahmoud Said et al. “InSDN: A Novel SDN Intrusion Dataset.” IEEE Access 8 (2020): 165263-165284.

[15] A. Kaan Sarica and P. Angin, "A Novel SDN Dataset for Intrusion Detection in IoT Networks," 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2020, pp. 1-5, doi: 10.23919/CNSM50824.2020.9269042. keywords: {Servers;Internet of Things;Tools;Intrusion detection;Real-time systems;Performance evaluation;Data models;Software-defined networking;machine learning;dataset;security}

[16] Mutalib, N.H.A., Sabri, A.Q.M., Wahab, A.W.A. et al. Explainable deep learning approach for advanced persistent threats (APTs) detection in cybersecurity: a review. Artif Intell Rev 57, 297 (2024). https://doi.org/10.1007/s10462-024-10890-4

[17] Zhiyan Chen, Murat Simsek, Burak Kantarci, Mehran Bagheri, Petar Djukic, Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier, Computer Networks,Volume 250,2024,110576,ISSN 1389-1286,https://doi.org/10.1016/j.comnet.2024.110576

[18] A. A. Alashhab et al., "Enhancing DDoS Attack Detection and Mitigation in SDN Using an Ensemble Online Machine Learning Model," in IEEE Access, vol. 12, pp. 51630-51649, 2024, doi: 10.1109/ACCESS.2024.3384398.

[19] Usman Haruna Garba, Adel N. Toosi, Muhammad Fermi Pasha, Suleman Khan, SDN-based detection and mitigation of DDoS attacks on smart homes,Computer Communications,Volume221,2024,Pages29-41,ISSN0140-3664,https://doi.org/10.1016/j.comcom.2024.04.001.

[20] Hill, W., Acquaah, Y.T., Mason, J. et al. DDoS in SDN: a review of open datasets, attack vectors and mitigation strategies. Discov Appl Sci 6, 472 (2024). https://doi.org/10.1007/s42452-024-06172-x

[21] M. S. Towhid and N. Shahriar, "Early Detection of Intrusion in SDN," NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 2023, pp. 1-6,

doi: 10.1109/NOMS56928.2023.10154272. keywords: {Training;Radio frequency;Deep learning;Intrusion detection;Focusing;Telecommunication traffic;Feature extraction;Real-time intrusion detection;SDN;machine learning;flow-based features},

[22] O. Rahman, M. A. G. Quraishi and C. -H. Lung, "DDoS Attacks Detection and Mitigation in SDN Using Machine Learning," 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 184-189, doi: 10.1109/SERVICES.2019.00051. keywords: {Control systems;Computer crime;Machine learning;Floods;Support vector machines;Network topology;SDN, DDoS, Machine Learning, J48, Weka},

[23] Jalal Bhayo, Syed Attique Shah, Sufian Hameed, Awais Ahmed, Jamal Nasir, Dirk Draheim, Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks, Engineering Applications of Artificial Intelligence,

Volume 123, Part C, 2023, 106432, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2023.106432.

[24] Vishal Giraddi, Shantala Giraddi, Narayan D G, Anupama Bidaragaddi, Suvarna G Kanakareddi, Machine Learning Approach to Intrusion Detection: Performance Evaluation, Procedia Computer Science, Volume 235, 2024, Pages 1851-1859, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2024.04.176.

[25] R. Sebopelo, B. Isong, N. Gasela and A. M. Abu-Mahfouz, "A Review of Intrusion Detection Techniques in the SDN Environment," 2021 3rd International Multidisciplinary Information Technology and Engineering Conference (IMITEC), Windhoek, Namibia, 2021, pp. 1-9, doi: 10.1109/IMITEC52926.2021.9714581. keywords: {Privacy;Intrusion detection;Telecommunication traffic;Machine learning;Denial-of-service attack;Security;Proposals;SDN;DDoS attack;IDS;Machine learning;Flow-based Anomaly.},

OriginalityReport FL24D0100

ORIGINALITY REPORT

11%	8%	6%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	3%
2	researchrepository.ucd.ie Internet Source	2%
3	Submitted to Pandit Deendayal Petroleum University Student Paper	1%
4	Mahmoud Said Elsayed, Nhien-An Le-Khac, Anca D. Jurcut. "InSDN: A Novel SDN Intrusion Dataset", IEEE Access, 2020 Publication	1%
5	hdl.handle.net Internet Source	1%
6	Jose Gomez, Elie F. Kfoury, Jorge Crichigno, Gautam Srivastava. "A survey on network simulators, emulators, and testbeds used for research and education", Computer Networks, 2023 Publication	1%