SDN-BASED INTELLIGENT INTRUSION DETECTION SYSTEM (IIDS) USING MACHINE LEARNING

Sriskandarajah J.P

IT21261978

BSc (Hons) Degree in Information Technology Specialized in Cyber Security

Department of Information Technology

Sri Lanka Institute of Information Technology Sri Lanka

April 2025

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: [illegible] Date: 11.04.2025

Signature of the supervisor: Date:

ABSTRACT

The rapid evolution of cyber threats, particularly Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks, presents significant challenges to network security. Traditional Intrusion Detection Systems (IDS) often suffer from high latency, limited adaptability, and reactive mitigation strategies, making them ineffective against modern cyber threats. To address this gap, this research proposes an SDN-Based Intelligent Intrusion Detection System (IIDS) that leverages machine learning for real-time detection and mitigation of UDP, SNMP, and DNS-based DoS attacks in Software-Defined Networking (SDN) environments.

The IIDS architecture integrates a machine learning-based intrusion detection engine with an Open Daylight (ODL) SDN controller to provide automated and adaptive threat mitigation. The intrusion detection engine developed using FastAPI, processes real-time flow statistics from the SDN controller and classifies traffic using Random Forest, Logistic Regression, and Neural Network models. Upon detecting an attack, the system dynamically installs flow rules via OpenFlow to mitigate malicious traffic, ensuring minimal disruption to legitimate users.

This research contributes to the field of cybersecurity and SDN by presenting an efficient, scalable, and real-time intrusion detection system that leverages machine learning for proactive security enforcement. Future work will focus on enhancing model adaptability, integrating distributed SDN controllers, and expanding attack detection capabilities to further improve network resilience against emerging cyber threats.

Keywords: Software-Defined Networking (SDN), Intrusion Detection System (IDS), Machine Learning, DoS Attacks, UDP Flood, SNMP Flood, DNS Flood, OpenFlow, Open Daylight, FastAPI

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to those who contributed to the successful completion of this research project. First and foremost, I would like to thank my supervisors, Mr. Kanishka Prajeewa Yapa and Mr. Tharaniyawarma.K for their support, guidance, and encouragement throughout the research process.

I would also like to acknowledge the invaluable contributions of my teammates. Their collaboration, dedication, and creativity greatly enhanced our collective understanding of the research topic. Our teamwork and shared passion for the project were crucial to its successful completion.

Finally, I would like to express my deepest appreciation to my family, friends, and colleagues. Their constant support, patience, and understanding during this period provided me with the strength and motivation to persevere.

Table of Contents

DECLARATION i

ABSTRACT ii

ACKNOWLEDGEMENT iii

LIST OF ABBREVIATIONS vii

INTRODUCTION 1

Background Literature 3

Research Gap 6

Research Problem 10

Research Objectives 11

METHODOLOGY 16

Commercialization aspects of the product 21

Testing 22

Implementation 30

RESULTS & DISCUSSION 33

Results 33

Research Findings 37

Discussion 38

Summary 40

CONCLUSION 43

REFERENCES 45

Figure 1:SDN 2
Figure 2:Dataset 17
Figure 3:SMOTE 18
Figure 4:Best Model Selection 19
Figure 5:Open Day Light Login 19
Figure 6:FastAPI Interface 20
Figure 7:System Diagram 21
Figure 8:Importing Libraries 23
Figure 9: Random Forest Model Training 23
Figure 10: Logistic Regression Model Training 24
Figure 11:Neural Network Model Training 24
Figure 12: Random Forest Confusion Matrix 25
Figure 13: Prediction False 26
Figure 14: Prediction True 26
Figure 15: SDN Website Detection 27
Figure 16:ODL Nodes 27
Figure 17:ODL Topology 28
Figure 18:FastAPI attack detection 29
Figure 19:Website 34
Figure 20:DoS Dashboard 35
Figure 21:Main Dashboard 35
Figure 22:ODL Openflow 36
Figure 23:Openflow Details 36

Table 1:List of Abbreviation vii

Table 2: Research Gap 9

Table 3: DoS Attack Types 15

Table 4:Flow Rule Modification for DoS Mitigation 15

Table 5:Commercialization and Cost Analysis 22

LIST OF ABBREVIATIONS

Abbreviation	**Description**
SDN	Software Defined Networking
IIDS	Intelligent Intrusion Detection System
ML	Machine Learning
DoS	Denial of Service
SMOTE	Synthetic Minority Over-Sampling Technique
DNS	Domain Name System
UDP	User Datagram Protocol
SNMP	Simple Network Management Protocol
ODL	Open Day Light
MLP	Multi-layer Perceptron
LSTM	Long-Short Term Memory
CSV	Comma Separated Values
PCA	Principal Component Analysis
SVM	Support Vector Machine
SECaaS	Security-as-a-Service
IoT	Internet of Things
API	Application Programming Interface
CNN	Convolutional Neural Networks
IP	Internet Protocol

Table 1:List of Abbreviation

INTRODUCTION

The rapid advancement of network technology has introduced an age featuring essential criteria of adaptability alongside scalability together with efficiency in data transmission and processing and security models. SDN has reshaped networking by splitting the control function from the data transmission functions which creates a centralized management approach for programmable network resources. The new architectural design creates exceptional chances for network protection against escalating cyber threats through its ability to combat Denial of Service (DoS) attacks that target network resources and disrupt service availability. Intrusion detection systems that operate traditionally show limited effectiveness when faced with rapid network traffic changes and significant volume in software-defined network environments where speed of response matters. SDN-based IDS performance can be enhanced efficiently and accurately through machine learning (ML) integration to detect and mitigate Denial of Service attacks. SDN-Based Intelligent Intrusion Detection System (IIDS) development proceeds through designing a Machine Learning-Based Intrusion Detection Engine specialized for detecting DoS attacks. The work aims to address DoS attack sophistication because the vulnerabilities present in network infrastructure result in severe economic and operational disruption.

The research develops a strong adaptable network system which employs SDN centralized control alongside predictive ML to monitor and detect DoS attack patterns in real-time thus improving network resilience. The use of ML in intrusion detection represents more than mere technological improvement since it creates a fundamental change that allows systems to gain knowledge from past events and evolve their threat responses beyond traditional static rule-based methods. The authors demonstrate how SDN's programmability features enable deployment of an ML-powered engine for detecting DoS attack indicators in network monitoring activities. Through automation the system initiates the execution of pre-defined security mitigation steps smoothly. This study shows significant importance because it connects the academic theory of ML with real-world network security practice through a blueprint for future-generation IDS systems operated in SDN networks.

The pressing research needs stem from DoS attack development since these assaults now progress from basic bandwidth floods into organized attacks on network protocols and application layers so they overcome existing sturdy conventional defense systems. Within SDN systems the network controller offers wide traffic visibility but functions simultaneously as both a protective element and a point of potential congestion when under attack from harmful network flows. This work applies ML detection algorithms to central network visibility information which transforms such data into security assets through flow analytics that expose DoS indicators including drastic packet rate rises together with ill-formed flow durations and unbalanced resource utilization behaviors. The IIDS differs from standard IDS through its use of machine learning capacities since it trains data to detect known and unknown malicious attack forms with automation. The ability to adapt rapidly stands as an essential requirement in SDN networks because policy reconfiguration needs a security system to both evolve swiftly and maintain performance levels. Through the programmable interfaces of SDN the IIDS automates its functions to detect threats and gives real-time instructions to the controller for traffic rerouting and malicious packet dropping and segment isolation thus reducing network disruptions. Business losses because of distributed denial of service attacks reach billions each year because of outages which negatively affect operations and decrease trust in various sectors including e-commerce and healthcare. SDN becomes more effective through the implementation of intelligent components within its framework.

The research focuses on creating a system which identifies DoS attacks precisely and adapts readily to different network configurations including small businesses to extensive data centers. The IIDS functions as a future-shaping security instrument since it combines theoretical advancements with practical network security applications for software-defined infrastructure implementations.

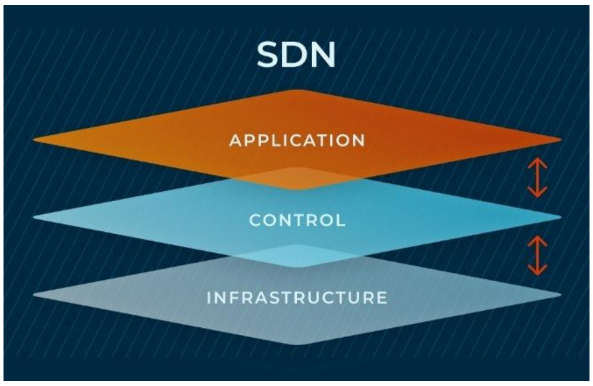


Figure 1:SDN

Background Literature

Network security research has evolved significantly since Software-Defined Networking (SDN) launched because it migrated security control from hardware-operated systems to software-controlled programmable platforms. Online research produced by Feamster and others [1]. demonstrated that SDN technology allows fine security policy enforcement by utilizing OpenFlow protocol. The ability to execute SDN protocols programmatically enables automated policy applications during runtime which results in better traffic monitoring capabilities and makes SDN crucial for cybersecurity purposes. SDN controllers exist as central vulnerabilities because attackers can launch Denial-of-Service (DoS) attacks against them due to the architecture's centralized structure. Security needs to be a top concern in SDN-based networks because a single compromised SDN controller threatens to shut down the entire network. The centralized architecture of SDN serves both as an attack vector and as a benefit because network-wide traffic analysis enables superior intrusion detection capabilities according to Scott-Hayward and others [2]. According to Kreutz et and others [3]. OpenFlow protocols enable SDN networks to run efficiently by providing real-time capabilities for network policy regulation. Research findings about Intrusion Detection Systems (IDS) in SDN frameworks led to the development of programs that use flow rules to identify and react to attacks automatically. SDN-enhanced IDS represents a potential solution according to Bhadauria and Sanyal [4] because SDN networks provide flexible capabilities that can enhance anomaly detection systems. The separation of harmful network traffic from genuine activity continues to pose challenges for detection systems because current techniques need better intellectual defense mechanisms. SDN-ML convergence in cybersecurity has become possible because researchers decided to utilize machine learning (ML) for addressing traditional IDS limitations.

In parallel with SDN developments machine learning became a strong intrusion detection tool because it understands large data volumes and detects attack signatures. Researchers conducted a systematic review of ML techniques which included Decision Trees, k-Nearest Neighbors and the clustering approach k-Means to show their ability in detecting abnormalities in network traffic [5]. Thottan and Ji developed a DOS detection system through time-series network traffic analysis for statistical anomaly detection that proved successful for resource-targeted attack identification [6]. The system developed by their team proved insufficient for high-speed operational effectiveness making it unfit for practical usage. The adoption of ensemble methods and deep learning architecture represented major progress in ML technology development. Mukkamala and others researched Support Vector Machines (SVM) for intrusion detection through their analysis of DoS attacks data which produced accurate classification results [7]. Support Vector Machines faced key limitations in processing speed that became problematic during wide-scale deployment of networks. Deep learning integration with SDN-based IDS systems achieved a traffic pattern classification accuracy of more than 90% through the application of Convolutional Neural Networks (CNNs) on the NSL-KDD dataset according to Tang and others [8] CNN-based IDS models need substantial computer power which makes them difficult to implement within limited SDN controller environments. Researcher Bhayo together with other researchers investigated alternative ML approaches following the accuracy-efficiency trade-off thus developing SVM classifiers that maintain both interpretability and performance [9]. Most research involved with these models relies on simulated traffic data because real-world tests remain sparse. Scalable SDN requires real-time ML models with computational efficiency that need additional research to fulfill current requirements.

The development of ML-based IDS systems requires benchmark datasets that establish training and validation settings for intrusion detection models. The KDD CUP 99 dataset served as a common benchmark for many years, yet researchers faulted it because of repetitive data alongside the mismatch between its representations and modern cybersecurity threats. The researchers of Tavallaee and others developed NSL-KDD to resolve KDD Cup ’99's issues through duplicate record removal along with balanced attack class distribution to support ML investigation [10]. The CICIDS2017 dataset created by Sharafaldin and others has risen in popularity because it delivers authentic present-day attack behavior and SDN traffic [11]. The research on ML-based IDS receives substantial impacts from these datasets that enable investigators to test detection engines under different attack scenarios. The current datasets have an important weakness because they do not account for SDN-specific flow-based operational style. SDN processes its information through OpenFlow switches while working with flow-level statistics as opposed to traditional packet-based data systems. The difference between SDN's operational environment requires customized feature engineering together with model training tailored specifically for SDN's framework. Existing IDS encounter multiple operational problems because they produce high numbers of false positives and struggle to detect new attacks and fail to keep pace while threats evolve in networks. The analysis of sequential traffic through Long Short-Term Memory (LSTM) neural networks represents a deep learning approach investigated by researchers according to Kim and others [12]. Node-based IDS benefit from LSTMs because these networks exhibit capacity to detect temporal patterns in network activity which leads to superior anomaly detection capability. SDN technology currently requires further research to optimize deep learning implementation within its framework.

A promising solution to stop DoS threats emerges when SDN benefits from global traffic visibility and the predictive expertise of ML. SDN anomaly detection in real-time applications becomes feasible according to Braga and others through their usage of Self-Organizing Maps (SOM) for DDoS attack detection using OpenFlow flow statistics [13]. The introduced lightweight detection systems exhibited challenges in detecting stand-alone DoS attacks since these attacks demonstrate distinct differences in scale and purpose from distributed DoS variants. SDN real-time network monitoring capability joins forces with ML-based classifiers to better detect IoT distributed denial-of-service attacks [14]. Research advances have not yet solved the difficulties of producing efficient ML algorithms for SDN infrastructure because they must achieve optimal detection accuracy and system efficiency simultaneously. The utilization of simulated traffic instead of actual network data in numerous existing research studies prevents their application in real SDN deployment environments. This investigation establishes a connection between differing areas by creating an SDN-based Intelligent Intrusion Detection System (IIDS) which utilizes real-time ML-driven methods to identify and stop DoS attacks. The research utilizes optimized features from ML models which match SDN network traffic patterns to deploy them effectively within

SDN controllers to achieve security improvements along with scalability enhancements. Research combines SDN with ML to advance intrusion detection scientific methods which create resilient threat mitigation systems that operate in real time for current network environments.

Research Gap

Research papers about SDN and ML-based intrusion detection systems demonstrate numerous developments but still leave substantial gaps when applied to standalone DoS attacks that appear in SDN deployments. SDN has received acclaim for its centralized control capabilities along with programming features that enhance network management, but researchers show insufficient application of these elements for creating real-time detection systems designed for single-source DoS attacks and their resource-draining methods such as SYN flooding or UDP storms. A majority of research studies deal with SDN-based intrusion detection focus on broad anomaly detection or Distributed Denial of Service (DDoS) attacks, but these approaches vary significantly from single-source DoS threats both in terms of their dimension and complexity. The general approach to detection misses an essential aspect because DoS attacks while less widespread can signal future complex attacks which need specialized detection methods that recognize rapid packet surge or irregular flow durations in an SDN's flowing network traffic. The lack of independent DoS detection solutions creates security risk for SDN networks because basic threats target the administrator interface without causing multi-purpose attacks like distributed denial of service systems do. Research indicates a lack of proper solutions for DoS-specific pattern detection by real-time analysis in SDN-based Intelligent Intrusion Detection Systems thus requiring immediate development.

Researchers have not adequately integrated machine learning algorithms into programming interfaces for network controllers to build tailored defenses for DoS assault incidents. The wide application of ML in intrusion detection as demonstrated through Buczak and Guven does not address SDN operations and fails to recognize DoS traits specifically [15]. SDN controllers require resource-constrained operations which make deep learning and ensemble method-based approaches from studies incompatible because they demand extensive computing resources in controlled settings. The utility of Long Short-Term Memory (LSTM) networks for sequential traffic analysis is demonstrated [12]yet the study lacks SDN-specialized application which creates a gap for transforming these methods into SDN flow-based DoS detection systems. SDN security surveys [4] describe the potential of programmable IDS yet they refrain from implementing ML-based DoS protection and instead analyze theoretical security issues. The integration shortfall becomes worse due to the usage of static NSL-KDD datasets despite their value for training because they cannot model real-time SDN traffic at high speeds that DoS attacks require. A disconnect emerges from the analytical strengths of ML contrasting with the operational needs of SDN because a customized engine needs to integrate flow statistics from SDN controllers to detect and automatically handle DoS attacks within the controller framework.

Scalability stands as an insufficiently investigated aspect when dealing with DoS detection systems used in large-scale SDN network environments due to modern network scale expansion requirements. SDN’s unique architecture still requires testing of scalability during rapid single-source traffic surges that match DoS attack characteristics [15] and other ML-based IDS studies. SDN security research shows limited interest in investigating ML's impact on controller efficiency during comprehensive topology management. Also insufficient attention is given to studying small-scale testing methods that might help understand SDN security behavior. Sommer and Paxson identified the ML challenges in network intrusion detection particularly for high-speed networks although their study predates SDN and does not address its centralized network control approach [16]. The major scalability problem affects DoS detection because a single malicious source creates so much network traffic that it challenges the capabilities of both the SDN infrastructure and the detection engine. The current literature has ignored developing lightweight ML models which can function as defense systems across diverse SDN deployments that extend from enterprise networks to data centers so practical deployment of IDS solutions becomes limited for dealing with real-world DoS threats. An ML-based engine will serve as the focus of this research because it must both detect DoS attacks properly while maintaining efficient scalability for diverse network conditions.

The current research shortage is worsened by the lack of automated intervention systems that match DoS attack patterns because most detection research ends without developing proactive defense solutions. Traditional intrusion detection systems together with some SDN-based systems need either predefined rules or human oversight to respond to threats yet proven insufficient for dealing with the quick tempo of DoS attacks because mere seconds could decide between system survival or breakdown. The programming flexibility of SDN network infrastructure allows administrators to activate flow rule changes and malicious traffic isolation but these capabilities remain underutilized by ML systems. SDN’s security potential is discussed [4] but the authors fail to present automated responses [12] focus on detection accuracy without integrating mitigation protocols. The absence of automated response measures becomes an important drawback since DoS attacks need instant responses including packet drops and traffic routing to prevent resource exhaustion particularly when targeting SDN controllers due to their critical position. The focus on accurate detection by the literature exposes SDN networks to DoS attack operational impacts since they lack an IIDS that pairs detection with automated mitigation conducted through SDN interfaces.

Weaknesses in the current research exist regarding the ability of existing systems to accommodate new modifications of standalone DoS attack methods. The techniques attackers use for DoS attacks evolve constantly because they develop new methods which current signature-based and anomaly-based systems with outdated patterns fail to detect properly. LSTM's dependency-learning ability demonstrates adaptability in the work [12] while their method lacks SDN-centricity and results in the warnings presented [16] about the lack of zero-day threat solutions specific to SDN. The usage of NSL-KDD static datasets produces a gap with SDN’s real-time operational requirements because these datasets do not mirror current DoS attack patterns in live environments. The survey [15] recognizes ML adaptability however it does not examine its potential for managing flow table updates when attacks occur. Detection engines need to operate with continuous learning capabilities since standalone DoS attacks evolve between bandwidth floods and protocol exploits. The proposed research brings forth an adaptive ML-based system to exploit SDN visibility capabilities for detecting and responding to emerging DoS attack types to create enduring security defenses in response to a dynamic security threat environment. Research that focuses on developing an all-encompassing practical solution for next-generation SDN security has become vital because of present gaps in DoS-specific focus, ML-SDN integration, scalability, automation and adaptability.

Research Gap Feature	**Research 1** [7]	**Research 2** [14]	**Research 3** [8]	**Research 4** [9]	**Proposed Research**
Focused on real-time detection of standalone DoS attacks in SDN environments	✗	✓	✗	✗	✓
Integration of ML algorithms with SDN controllers for DoS-specific detection	✗	✗	✗	✗	✓
Scalability for DoS detection in large-scale SDN networks	✗	✗	✓	✗	✓
Automated mitigation tailored to DoS attack patterns	✗	✗	✗	✗	✓
Adaptability to evolving standalone DoS attack variants	✗	✓	✗	✗	✓

Table 2: Research Gap

Research Problem

The relentless rise in the frequency and ingenuity of Denial of Service (DoS) attacks presents a formidable threat to network availability, particularly in Software-Defined Networking (SDN) environments where the centralized control paradigm offers transformative benefits alongside a critical vulnerability: the controller itself becomes a potential single point of failure when inundated by malicious traffic targeting resources through specific attack vectors such as UDP floods, SNMP floods, and DNS floods. This problem investigates the major shortcomings of current intrusion detection systems which fail to leverage machine learning capabilities for detecting and blocking the three distinct standalone DoS attacks that occur in SDN frameworks. The research focuses on the unique challenges of detecting these attacks in SDN environments caused by their dynamic traffic patterns and real-time operational demands. Using the protocol's fast speed and easy nature UDP floods generate numerous connectionless UDP packets to exhaust server resources and fill up network bandwidth beyond traditional defense capabilities. The management protocol SNMP provides attackers a means to launch flood attacks which send massive queries and responses toward network infrastructure equipment including routers and switches thus impacting essential SDN management capabilities. DNS flood attacks disrupt names resolution services through many false DNS requests that simultaneously paralyze service access while increasing SDN system downtime. Traditional IDS struggles with current DoS variants because its signature-based methods fail to recognize the different patterns of UDP packet storms, SNMP query overflows and DNS request peaks. As a result, networks remain vulnerable to unknown attack methods. The flexible nature of anomaly-based systems leads to excessive false positives by confusing normal UDP activity and SNMP requests as well as regular DNS activity for attacks, yet these systems generate high computational burden which conflicts with SDN's momentary response requirements until widespread network interruptions failure. The combination of fast flow programming in SDN networks and critical time-sensitive response requirements puts substantial strain on current ML-based IDS to distinguish between UDP, SNMP, and DNS flooding signatures from normal network fluctuations during multimedia streaming or network administration workloads or user-initiated DNS inquiries. SDN faces an intensified challenge because there exists no standardized framework integrating an adaptive ML engine which learns attack patterns like UDP packet rates and SNMP latency spikes and DNS query anomalies while executing attack-specific countermeasures without affecting network performance. This research work sets the goal of developing a detection system which serves both precision needs and operational efficiency requirements to detect the three DoS threats effectively in SDN controllers and high-speed data planes even under resource constraints that require strict balance of computational and latency demands for sustained operational functionality. The stakes are extraordinarily high: undetected or inadequately managed UDP, SNMP, and DNS flood attacks can precipitate severe service disruptions halting data transfers, disabling network management, or breaking domain resolution leading to substantial financial losses for industries reliant on continuous connectivity, such as e-commerce, telecommunications, and cloud services, while eroding trust in the reliability of network-driven systems that form the backbone of modern digital infrastructure. Research shows the necessity to create a specific solution which combines SDN's programmable and visible framework with ML's thorough analysis to build strong real-time defense systems against DoS attacks in order to enhance network resistance against targeted implementation. This research initiative transcends standard tool improvement because it aims to reshape SDN security methodologies by specifically defending against the threats which UDP, SNMP and DNS floods generate from disrupting network operation.

Research Objectives

This research seeks to create and launch an SDN-Based Intelligent Intrusion Detection System (IIDS) containing an Intrusion Detection Engine built with Machine Learning algorithms dedicated to detecting Denial of Service (DoS) attacks along with their disruptive types UDP floods, SNMP floods, and DNS floods which represent specific challenges to network availability within Software-Defined Networking (SDN) environments. The main objective includes multiple specific targets that resolve acknowledged research limitations and fix inadequate DoS detection within SDN through a security system that unites SDN architecture benefits with Machine Learning analytical methods. The initial objective entails the creation of a ML framework that detects UDP, SNMP, DNS DoS attacks through specific feature selections designed for SDN traffic characteristics which include bandwidth-overloading UDP packet rate spikes and SNMP query surge events and DNS request volume increases that lead to service disruption. The system implements a systematic feature extraction method which derives metrics related to the interarrival times of packets and flow lengths and protocol-specific irregularities from flow-based SDN data to make sure the engine can detect single-source attack signatures from valid traffic characteristics such as multimedia streaming and SNMP polling or DNS query peaks. The framework chooses to focus on SDN-specific network attributes instead of generic features to boost detection accuracy because it handles the weakness of current systems in identifying DoS behaviors within SDN traffic speed. This research target possesses vital importance because it transforms eighty-six SDN data points into meaningful operational information which helps the IIDS detect elusive yet damaging features of these three DoS variants with remarkable precision to protect networks against their attacks.

The deployment of this detection engine through ML must integrate perfectly with SDN architecture to exploit the controller's ability to view and program the network for effective UDP and SNMP and DNS flood prevention. The ML engine needs to be integrated inside Open Daylight controllers for real-time analysis of OpenFlow switch statistics to monitor entire network activity which standard distributed systems cannot match. The controller's position across the entire traffic network lets the engine identify irregular UDP packet floods in data planes together with SNMP floods affecting management interfaces and DNS floods causing service breakdowns thus activating immediate warning alerts during threat occurrences. Put simply SDN allows automated detection threshold adjustment which makes it possible to detect emerging DoS threats in real-time because of their rapid development. The research problem of poor ML-SDN integration receives solutions through this development that unifies theoretical ML models with real-world SDN implementations to warrant an integrated position of the engine inside the network . The aim is to build an integrated detection solution which proactively uses SDN's centralized control to address UDP, SNMP and

DNS DoS threats using precise detection mechanisms and smooth response capabilities to preserve operational integrity in the network.

The development of a lightweight ML algorithm for flood detection stands as the third objective that combines excellence in detecting UDP SNMP and DNS flood events and effective deployment within SDN resource-limited structure. The optimized model designed under this objective avoids deep learning methods which fail to meet real-time SDN requirements by incorporating Random Forest or shallow neural networks. The compact design of the system permits fast flow data processing while identifying UDP bandwidth exhaustion and SNMP response delays alongside DNS query traffic along with sending alerts to the controller while the controller performs policy handling and traffic control functions. The proposed efficiency level enables closure of the existing research gap that occurs because current ML-based IDS systems use impractical computational methods to detect DoS variants by delivering fast and dependable detection while optimizing performance. This achievement demonstrates two important benefits for the IIDS which can be deployed across various sizes of SDN networks and establishes benchmark standards for operational security systems that maintain network resilience during denial-of-service attacks.

The fourth objective evaluates the IIDS system by running a robust assessment process with simulated SDN traffic and NSL-KDD benchmark data to measure detection rates as well as false-positive rates and UDP, SNMP, and DNS DoS response durations. The researchers will employ Mininet to create simulated SDN topologies that contain generated UDP floods and SNMP floods as well as DNS floods to examine how the engine detects these attacks while normal traffic consists of UDP video streams and SNMP monitoring and DNS resolution requests. Standardized knowledge derived from NSL-KDD dataset will help validate the engine’s performance through labeled DoS attack instances while being expanded with SDN-specific flow data for authentic real-world scenarios. The system detection rate evaluates the attack detection effectiveness across all three attack types while minimizing disruption to legitimate operations becomes essential because the attack patterns mimic peak usage conditions. The system’s threat response speed will be measured through response time because SDN demands immediate threat detection to minimize damage during real-time operations. This analysis functions to assess total performance assessment by using factual evidence that can optimize the IIDS while also demonstrating its advanced security against DoS attacks and its operational dependability.

The fifth objective integrates adaptive learning methods into the ML engine to boost its ability to identify new UDP, SNMP and DNS DoS attack versions which enable continuous protection in an evolving threat environment. The attackers shift their methods from brute-force UDP attacks to stealthier packet patterns while manipulating SNMP queries for legitimate traffic appearance and creating DNS floods with randomized requests, so the system needs to learn from new traffic data to adjust detection models automatically. The engine needs periodic update with new SDN traffic samples through online learning or periodic retraining to learn novel attack signatures which differ from NSL-KDD static patterns. The research gap of adaptability receives attention through this objective to guarantee that the IIDS functions against zero-day DoS threats without depending on the effectiveness of traditional or certain ML-based systems. Through adaptive system design the detection capabilities are strengthened and the protection extends into the future to support SDN network growth by adapting with evolving UDP and SNMP and DNS flood attack methods.

The research implements flow rule modifications into SDN-IIDS which enables blocking of malicious UDP, SNMP, DNS flood traffic while always enhancing automated network security. The SDN controller receives signals from the ML engine when an attack happens to establish flow rules for blocking offending traffic from source addresses or routing traffic to segregated areas for quarantine purposes. SDN's programming features enable the execution of exact attack-specific defenses that prevent disruption to standard network traffic while returning stability quickly. When a UDP flood occurs the system creates a rule to limit UDP packet flow while SNMP floods enable query source blocking and DNS floods generate rules which remove request source IP addresses from the network. Through this mitigation protocol detection turns into proactive threat handling which addresses the study challenge of static security through IIDS protection abilities. Multiple essential objectives form a flexible high-performance system to improve SDN intrusion detection capabilities through a solution which delivers effective protection against the expanding threats of UDP, SNMP and DNS DoS attacks and ensures trust in SDN network infrastructures.

Attack Type	**Targeted Protocol**	**Attack Method**	**Impact on SDN Environment**
UDP Flood	UDP	High-rate packet transmission	Bandwidth exhaustion, increased latency
SNMP Flood	SNMP	Excessive query requests	Controller and network device overload
DNS Flood	DNS	Malicious query amplification	DNS service disruption, network congestion

Table 3: DoS Attack Types

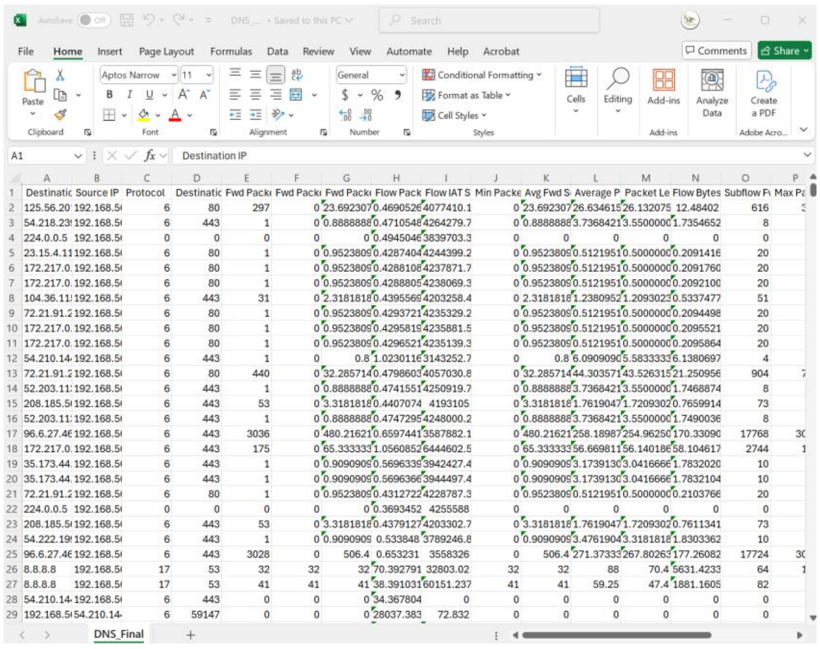
Attack Type	Flow Rule Implemented	Effect on Network
UDP Flood	Limit UDP packet rate per IP	Reduces bandwidth exhaustion
SNMP Flood	Block excessive query sources	Protects controller and devices
DNS Flood	Drop malicious query requests	Prevents service disruption

Table 4:Flow Rule Modification for DoS Mitigation

METHODOLOGY

An SDN-Based Intelligent Intrusion Detection System (IIDS) is organized, developed, and rigorously evaluated using carefully constructed, multi-phase architecture. Within the dynamic and flexible ecosystem of Software-Defined Networking (SDN), this system incorporates a Machine Learning-Based Intrusion Detection Engine to identify and prevent three distinct Denial-of-Service (DoS) attacks: UDP floods, SNMP floods, and DNS floods. The main goal is to create a strong, scalable, and effective cybersecurity solution that handles the shortcomings of current intrusion detection systems and fits in with the main research objectives of proactive mitigation, real-time detection, system efficiency, and flexibility. To ensure that every stage contributes to a conclusion that is supported by scientific research, this section offers a thorough, step-by-step explanation of the procedures that were followed to move from initial data collecting to system implementation and evaluation. The IIDS's commercialization potential as a useful security solution for contemporary network infrastructures is also examined.

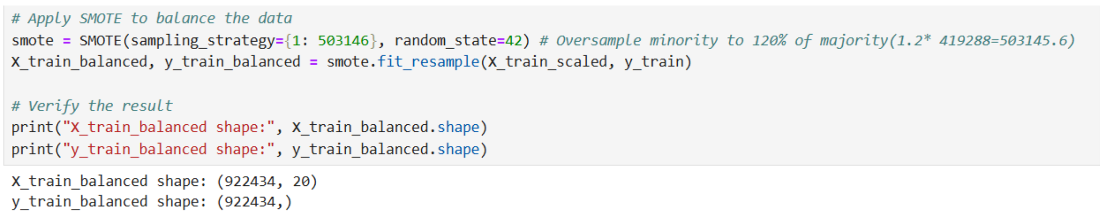
The initial stage focuses on obtaining data for preparing it before supplying it to the ML operation in a fashion that allows detection of specific UDP and SNMP and DNS DoS attack behaviors within SDN networks. The data collection process starts with real-time network traffic monitoring through Wireshark while the tool operates on a simulated SDN topology across both normal conditions consisting of UDP video streaming and SNMP device queries and DNS requests and attack scenarios using UDP packet floods and SNMP query floods and DNS request spikes. After data captures the system transforms the raw data into CSV files with timestamp information alongside source/destination IP addresses and protocol types and packet sizes data that prepares the files for ML processing capabilities. The NSL-KDD dataset functions as supplemental real-world data because it includes labeled DoS instances for additive value. The custom-generated SDN flow statistics provide necessary supplements to NSL-KDD because the benchmark has issues reflecting current SDN-specific traffic characteristics. The inclusion of OpenFlow-based standards provides flow duration and packet rate and byte count measures to represent bandwidth-choke behavior of UDP attacks and disruption of SNMP services and DNS service denial through DNS floods. The integration of historical benchmarks and contemporary SDN dynamics requires a dataset that resolves the gap conventional IDSs face when using old or general datasets.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Destinatic	Source IP	Protocol	Destinatic	Fwd Pack	Fwd Pack	Fwd Pack	Flow Pack	Flow IAT S	Min Packe	Avg Fwd S	Average P	Packet Le	Flow Bytes	Subflow F	Max Pa
2	125.56.20	192.168.5	6	80	297	0	23.692307	0.4690526	4077410.1	0	23.692307	26.634615	26.132075	12.48402	616	3
3	54.218.23	192.168.5	6	443	1	0	0.8888888	0.4710548	4264279.7	0	0.8888888	3.7368421	3.5500000	1.7354652	8	
4	224.0.0.5	192.168.5	0	0	0	0	0	0.4945046	3839703.3	0	0	0	0	0	0	
5	23.15.4.11	192.168.5	6	80	1	0	0.9523809	0.4287404	4244399.2	0	0.9523809	0.5121951	0.5000000	0.2091416	20	
6	172.217.0.	192.168.5	6	80	1	0	0.9523809	0.4288108	4237871.7	0	0.9523809	0.5121951	0.5000000	0.2091760	20	
7	172.217.0.	192.168.5	6	80	1	0	0.9523809	0.4288805	4238069.3	0	0.9523809	0.5121951	0.5000000	0.2092100	20	
8	104.36.11	192.168.5	6	443	31	0	2.3181818	0.4395569	4203258.4	0	2.3181818	1.2380952	1.2093023	0.5337477	51	
9	72.21.91.2	192.168.5	6	80	1	0	0.9523809	0.4293721	4235329.2	0	0.9523809	0.5121951	0.5000000	0.2094498	20	
10	172.217.0.	192.168.5	6	80	1	0	0.9523809	0.4295819	4235881.5	0	0.9523809	0.5121951	0.5000000	0.2095521	20	
11	172.217.0.	192.168.5	6	80	1	0	0.9523809	0.4296521	4235139.3	0	0.9523809	0.5121951	0.5000000	0.2095864	20	
12	54.210.14	192.168.5	6	443	1	0	0.8	1.0230116	3143252.7	0	0.8	6.0909090	5.5833333	6.1380697	4	
13	72.21.91.2	192.168.5	6	80	440	0	32.285714	0.4798603	4057030.8	0	32.285714	44.303571	43.526315	21.250956	904	7
14	52.203.11	192.168.5	6	443	1	0	0.8888888	0.4741551	4250919.7	0	0.8888888	3.7368421	3.5500000	1.7468874	8	
15	208.185.5	192.168.5	6	443	53	0	3.3181818	0.4407074	4193105	0	3.3181818	1.7619047	1.7209302	0.7659914	73	
16	52.203.11	192.168.5	6	443	1	0	0.8888888	0.4747295	4248000.2	0	0.8888888	3.7368421	3.5500000	1.7490036	8	
17	96.6.27.46	192.168.5	6	443	3036	0	480.21621	0.6597441	3587882.1	0	480.21621	258.18987	254.96250	170.33090	17768	30
18	172.217.0.	192.168.5	6	443	175	0	65.333333	1.0560852	6444602.5	0	65.333333	56.669811	56.140186	58.104617	2744	1
19	35.173.44.	192.168.5	6	443	1	0	0.9090909	0.5696339	3942427.4	0	0.9090909	3.1739130	3.0416666	1.7832020	10	
20	35.173.44.	192.168.5	6	443	1	0	0.9090909	0.5696366	3944497.4	0	0.9090909	3.1739130	3.0416666	1.7832104	10	
21	72.21.91.2	192.168.5	6	80	1	0	0.9523809	0.4312722	4228787.3	0	0.9523809	0.5121951	0.5000000	0.2103766	20	
22	224.0.0.5	192.168.5	0	0	0	0	0	0.3693452	4255588	0	0	0	0	0	0	
23	208.185.5	192.168.5	6	443	53	0	3.3181818	0.4379127	4203302.7	0	3.3181818	1.7619047	1.7209302	0.7611341	73	
24	54.222.19	192.168.5	6	443	1	0	0.9090909	0.533848	3789246.8	0	0.9090909	3.4761904	3.3181818	1.8303362	10	
25	96.6.27.46	192.168.5	6	443	3028	0	506.4	0.653231	3558326	0	506.4	271.37333	267.80263	177.26082	17724	30
26	8.8.8.8	192.168.5	17	53	32	32	32	70.392791	32803.02	32	32	88	70.4	5631.4233	64	1
27	8.8.8.8	192.168.5	17	53	41	41	41	38.391031	60151.237	41	41	59.25	47.4	1881.1605	82	
28	54.210.14	192.168.5	6	443	0	0	0	34.367804	0	0	0	0	0	0	0	
29	192.168.5	54.210.14	6	59147	0	0	0	28037.383	72.832	0	0	0	0	0	0	

Figure 2:Dataset

Data preprocessing ensues through three sequential steps which include numerical feature normalization for traffic scale normalization feature standardization for minimizing algorithm bias by mean normalization and unit variance scaling and outlier detection through the IQR method to remove anomalous data points. The core element of this stage involves extracting SDN-relevant characteristics from attack behavior data including UDP packet time intervals, SNMP response delays and DNS query rate patterns. The framework enables the ML system to recognize between harmful network traffic and standard streaming or standard DNS query activities. The implementation of Principal Component Analysis (PCA) transforms features from 88 to 20 without compromising 95% of the original data variance to achieve improved computational performance. The Synthetic Minority Over-Sampling Technique (SMOTE) addresses data imbalance because it handles the large ratio of normal traffic compared to attack traffic. SMOTE generates artificial samples of UDP, SNMP and DNS attacks which rebalance the dataset distribution through examples added to the minority class thus preventing the model from favoring major classes and improving detection of rare DoS events. The detection process requires this step for obtaining unbiased robust analysis under the real-time requirements of SDN.



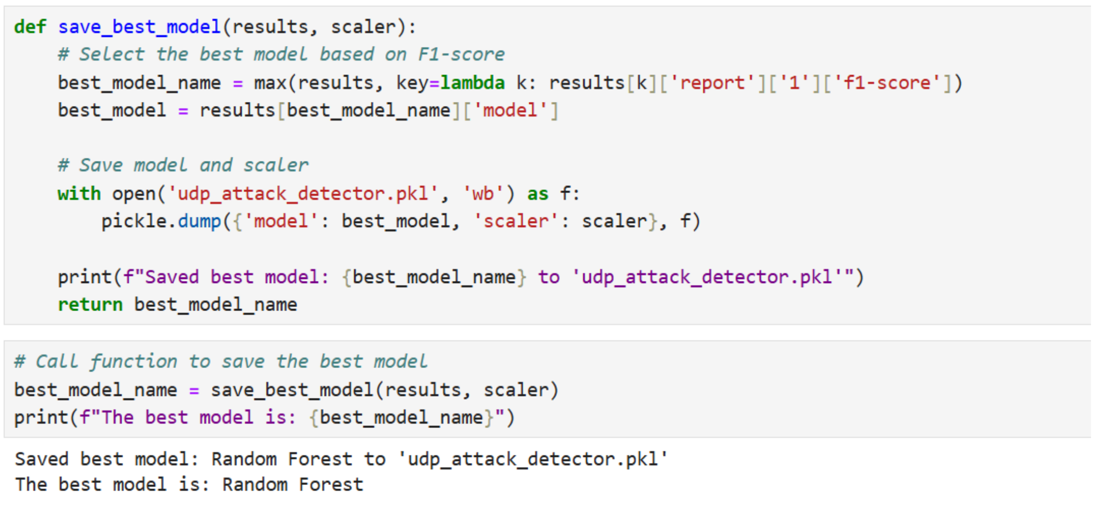
```
# Apply SMOTE to balance the data
smote = SMOTE(sampling_strategy={1: 503146}, random_state=42) # Oversample minority to 120% of majority(1.2* 419288=503145.6)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train_scaled, y_train)

# Verify the result
print("X_train_balanced shape:", X_train_balanced.shape)
print("y_train_balanced shape:", y_train_balanced.shape)
```

```
X_train_balanced shape: (922434, 20)
y_train_balanced shape: (922434,)
```

Figure 3:SMOTE

This second part focuses on selecting and training ML models as the methodology's core element for developing a detection engine that detects UDP, SNMP and DNS DoS attacks with high accuracy and minimal latency while adhering to SDN operational requirements. The initial selection of ML algorithms includes Random Forest because of its ensemble properties and interpretability while Logistic Regression provides baseline performance and Support Vector Machines (SVM) together with radial basis function kernels function well for boundary detection and Neural Networks resemble Multi-Layer Perceptron (MLP) offers adaptability to complex patterns. Random Forest emerges as the main selection because it demonstrates strong performance with class-imbalanced data sets while providing feature importance rankings requires less computational resources than deep learning models which showcases compatibility with SDN controller resource constraints. Supervised learning guides the training system which divides preprocessed data into training (80%) along with testing (20%) segments. The hyperparameter optimization process uses grid search to adjust the number of trees between 50,75 and 100 as well as the maximum depth from 10,15 and 20 with minimum split samples ranging from 2 to 5. This optimization maximizes detection accuracy above 95% while maintaining false positive rates below 2%. A stability assessment of data folds is done through 5-fold cross-validation. The performance measures accuracy, precision, recall and F1-score are used to generate quantitative outcomes. The detection latency should maintain velocities under 150 milliseconds and the processing efficiency should not exceed 20% of CPU usage.



```
def save_best_model(results, scaler):
    # Select the best model based on F1-score
    best_model_name = max(results, key=lambda k: results[k]['report']['1']['f1-score'])
    best_model = results[best_model_name]['model']

    # Save model and scaler
    with open('udp_attack_detector.pkl', 'wb') as f:
        pickle.dump({'model': best_model, 'scaler': scaler}, f)

    print(f"Saved best model: {best_model_name} to 'udp_attack_detector.pkl'")
    return best_model_name
```

```
# Call function to save the best model
best_model_name = save_best_model(results, scaler)
print(f"The best model is: {best_model_name}")
```

```
Saved best model: Random Forest to 'udp_attack_detector.pkl'
The best model is: Random Forest
```

Figure 4:Best Model Selection

The third phase constitutes the essential phase when integrating the pre-trained machine learning model into an operational SDN-Based Intelligent Intrusion Detection System (IIDS). Through the integration process real-time network traffic receives continuous automated monitoring and classification that enables the primary goal of SDN architecture intrusion detection and mitigation. Network functionality is programmed by Open Daylight (ODL) to achieve infrastructure establishment as the SDN controller. A realistic network topology is generated from Mininet to create a platform that allows the analysis of different traffic types including benign and malicious flows.

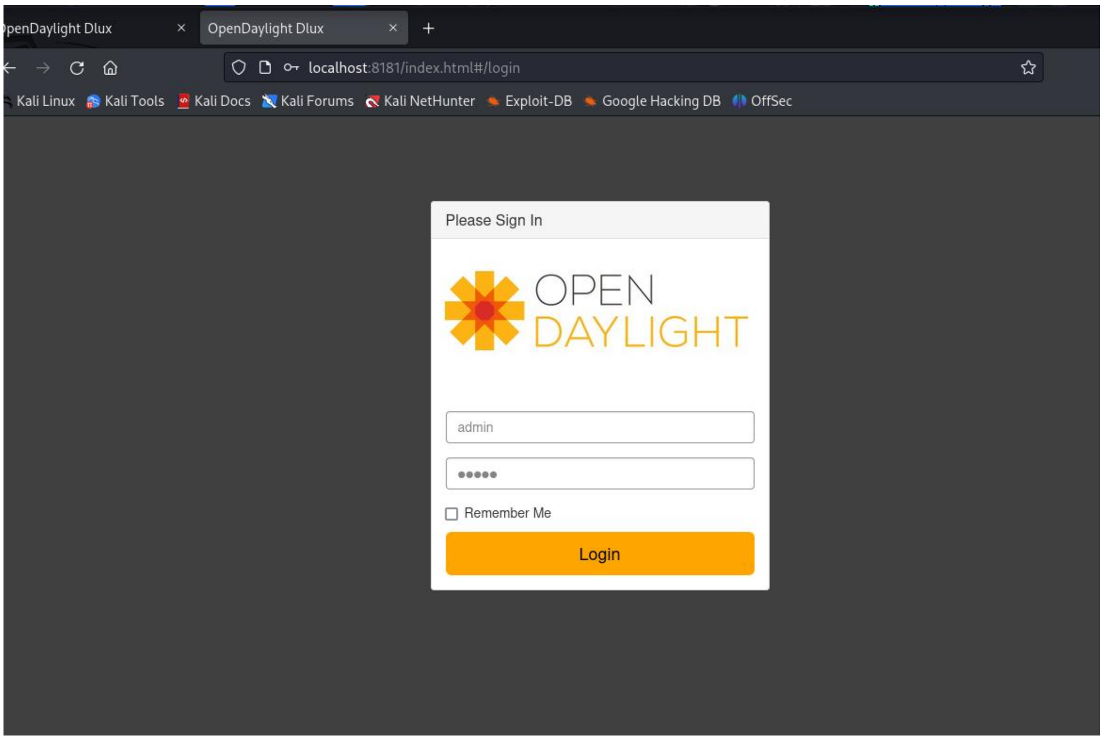
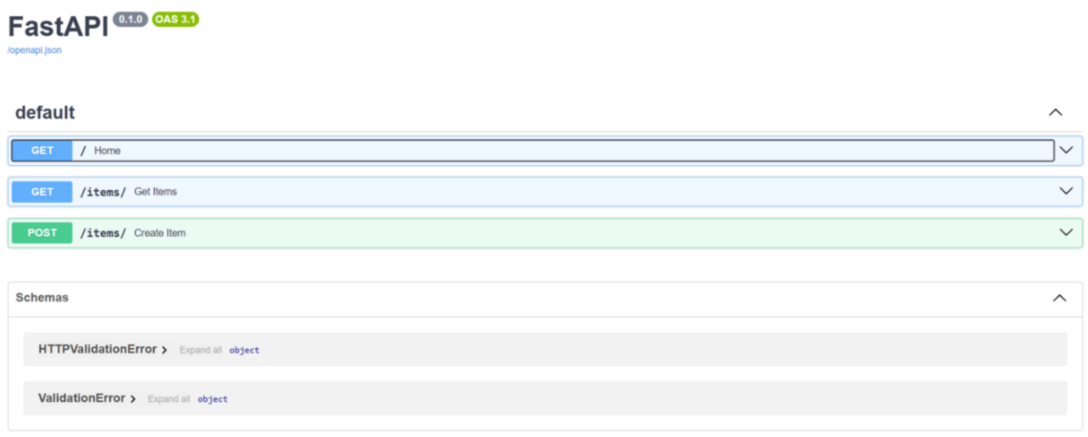


Figure 5:Open Day Light Login

The trained ML model exists in the implementation of FastAPI for serving as the communication pathway connecting the SDN controller with its detection engine. SDN switches transmit flow statistics like packet rates and byte counts and flow duration data to the FastAPI server by using its REST API interface. The FastAPI server executes simultaneous flow data processing while sending obtained statistics into the classification component of the ML-based detection mechanism for malice or benign flow identification. The SDN controller executes automatic mitigation responses when the detection engine identifies any flow attack that includes UDP flood attacks or SNMP floods or DNS floods. Open Daylight serves as the system responsible for implementing automatic adjustments to network flow rules based on detected security threats. Real-time mitigation of attacks takes place when the SDN controller identifies a UDP flood through specific IP addressing by creating blocking or rate-limiting flow rules. The system can recognize SNMP floods through its detectors to separate authorized SNMP traffic from overreaching queries by dropping them for protecting service functionality.


FastAPI 0.1.0 OAS 3.1
/openapi.json

default

GET / Home

GET /items/ Get Items

POST /items/ Create Item

Schemas

HTTPValidationError > Expand all object

ValidationError > Expand all object

Figure 6:FastAPI Interface

The user interface used by security analysts is powered by Next.js and displays threat detections and alert information as well as network traffic indicators in real-time. This dashboard obtains its data from the FastAPI backend to display critical metrics about attack frequencies together with affected IP addresses and response time details. Security events become observable through visual presentations which allow administrators to overrule automated decisions when needed through manual intervention. The system includes built-in logging and reporting features which record both intrusion events alongside their associated countermeasures for security policy compliance purposes. Performance testing evaluates real-time operational effectiveness through the assessment of detection latency which should be below milliseconds and CPU and memory utilization for efficiency and false positive rate measurement that avoids legitimate traffic blockage. The achievement of complete integration between machine learning technology and software-defined networks and automated responses during this phase transforms the proposed IIDS from an academic model into an operational system that tackles contemporary security threats in SDN environments.

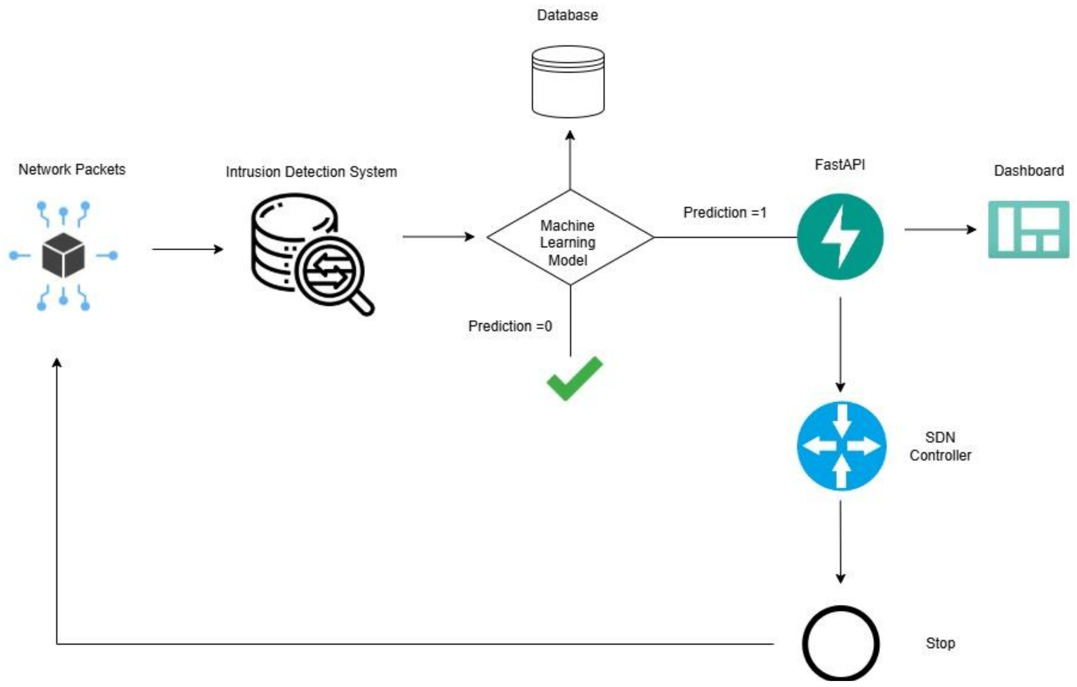


Figure 7:System Diagram

Commercialization aspects of the product

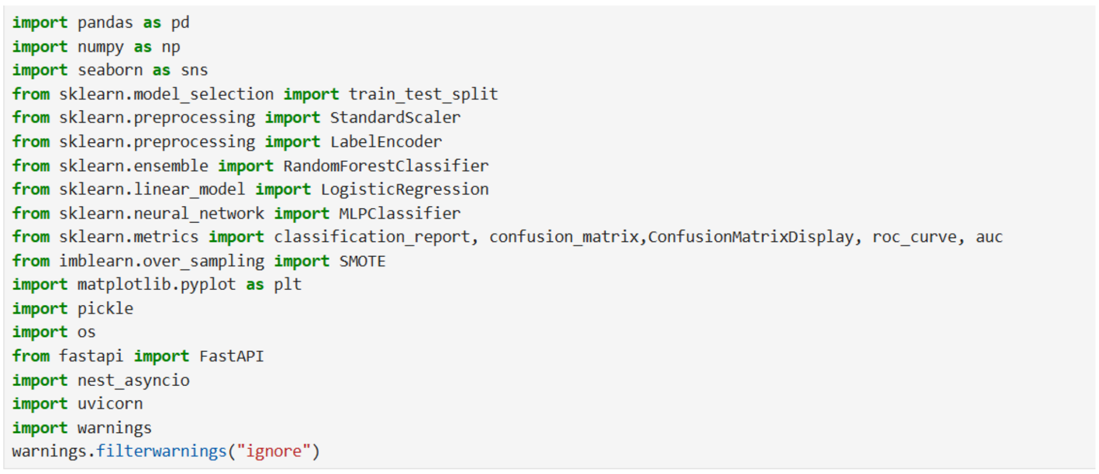
The phase discusses the SDN-IIDS commercial potential in detail. Network adoption of Software-Defined Networking at its peak creates substantial demand for real-time IDS solutions that provide intelligent capabilities. SDN-IIDS implements an adaptable modular architecture that easily connects to running SDN systems to deliver efficient real-time protection for DoS attacks. The device runs at low process requirements which makes it affordable regardless of network size and type. The SDN-IIDS security solution can be distributed through two potential commercialization approaches: SECaaS subscription services enabling businesses to access cloud-based IDS protection and licensing to cybersecurity vendors as part of their full security platform. Network equipment manufacturers can integrate the SDN-IIDS directly into future SDN controllers through cooperation which would offer built-in security features. Active deployment of this methodology will both lead to practical implementations and boost Internet network security against modern threats which supports continued AI-based research and development programs in cyber defense.

Component	**Estimated Cost ($)**	**Justification**
Open Daylight Controller	Free (Open source)	No licensing cost
Mininet SDN Emulator	Free (Open source)	Cost-effective research tool
Server (Intel i5, 16GB RAM)	$500	Affordable prototype deployment
FastAPI Backend Development	Free (Open source)	Lightweight and efficient
Next.js Frontend UI	Free (Open source)	User-friendly interface
Total Estimated Cost	<$500	Low-cost implementation

Table 5:Commercialization and Cost Analysis

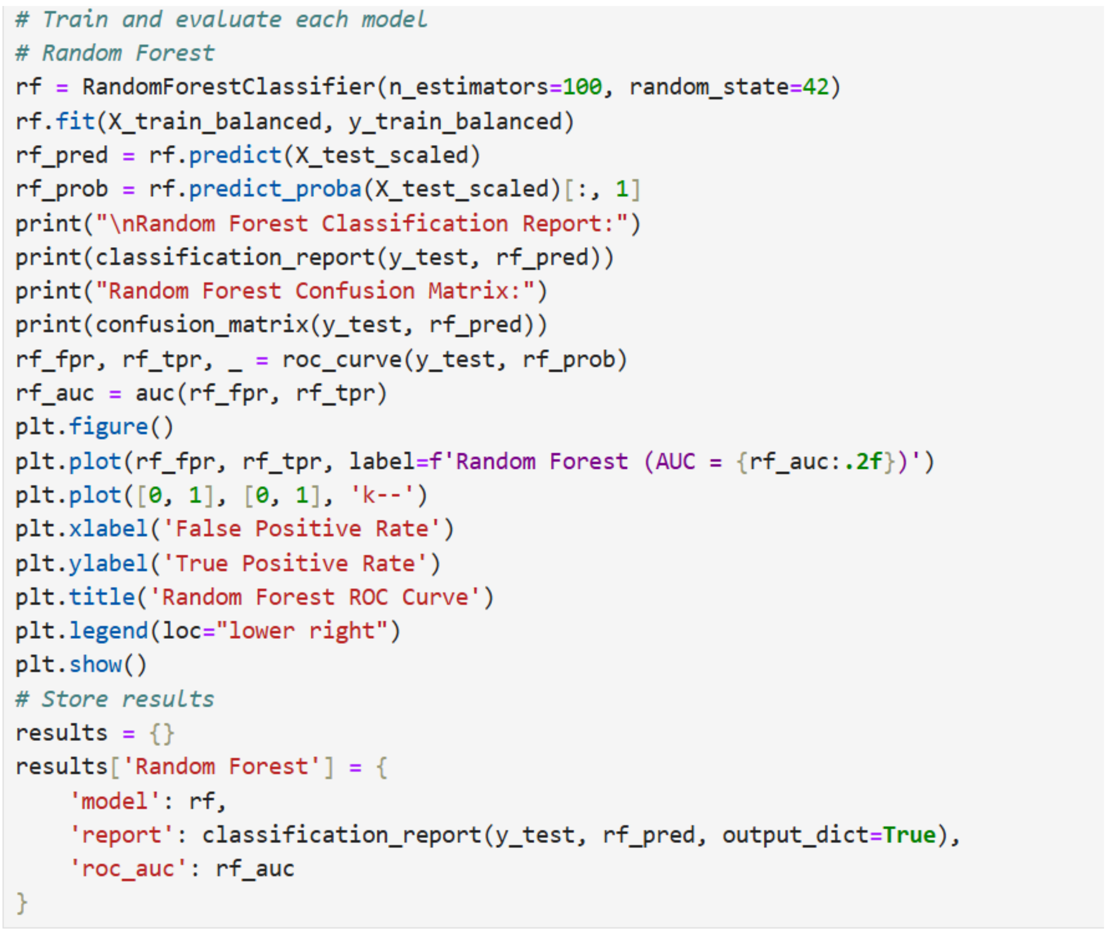
Testing

The testing methodology for the Intelligence Intrusion Detection System with its Machine Learning-Based Intrusion Detection Engine follows a detailed assessment of system capabilities for both effectiveness and efficiency and adaptability. The testing phase aims to evaluate the system's ability to detect and defend against UDP and SNMP and DNS DoS attacks within SDN infrastructures according to industry requirements and core research objectives about real-time responses along with system efficiency and adaptability as well as proactive countermeasures.



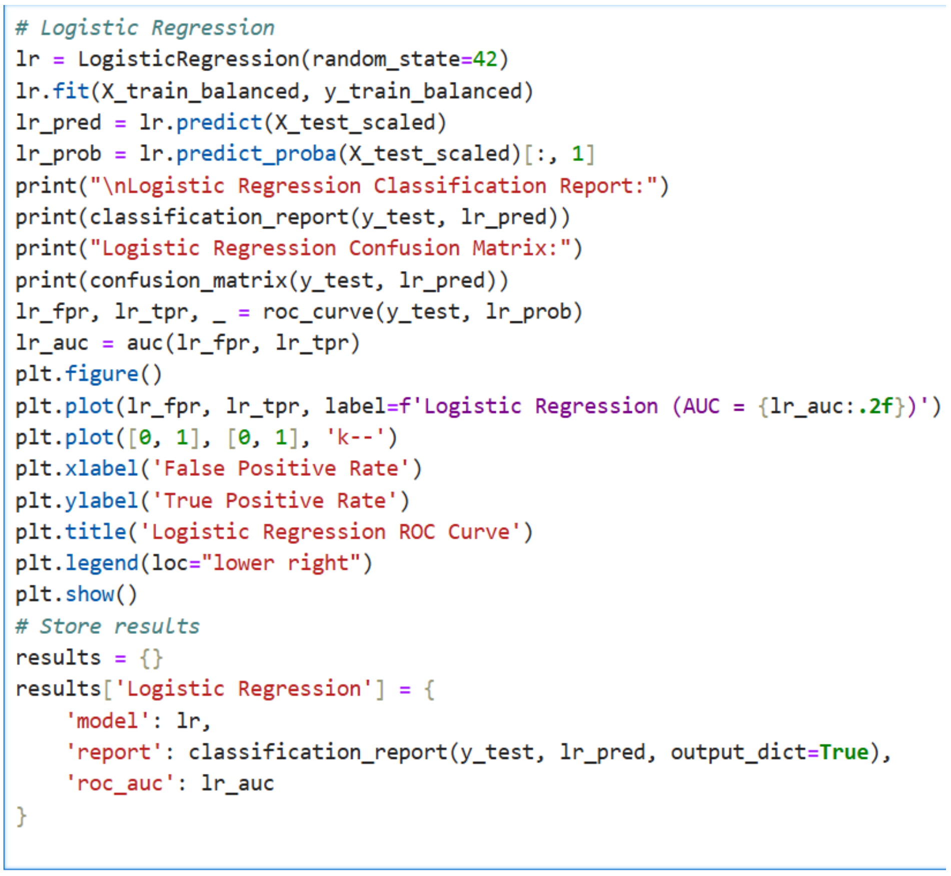
```python
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix,ConfusionMatrixDisplay, roc_curve, auc
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import pickle
import os
from fastapi import FastAPI
import nest_asyncio
import uvicorn
import warnings
warnings.filterwarnings("ignore")
```

Figure 8:Importing Libraries



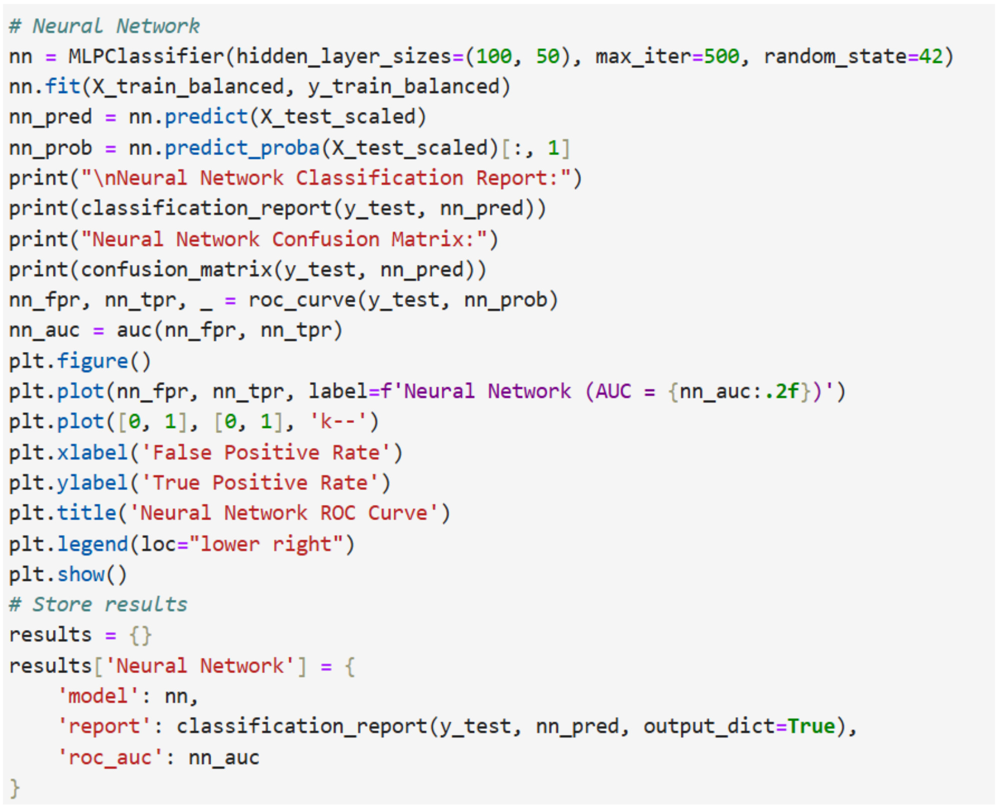
```python
# Train and evaluate each model
# Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_balanced, y_train_balanced)
rf_pred = rf.predict(X_test_scaled)
rf_prob = rf.predict_proba(X_test_scaled)[:, 1]
print("\nRandom Forest Classification Report:")
print(classification_report(y_test, rf_pred))
print("Random Forest Confusion Matrix:")
print(confusion_matrix(y_test, rf_pred))
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_prob)
rf_auc = auc(rf_fpr, rf_tpr)
plt.figure()
plt.plot(rf_fpr, rf_tpr, label=f'Random Forest (AUC = {rf_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve')
plt.legend(loc="lower right")
plt.show()
# Store results
results = {}
results['Random Forest'] = {
    'model': rf,
    'report': classification_report(y_test, rf_pred, output_dict=True),
    'roc_auc': rf_auc
}
```

Figure 9: Random Forest Model Training



```python
# Logistic Regression
lr = LogisticRegression(random_state=42)
lr.fit(X_train_balanced, y_train_balanced)
lr_pred = lr.predict(X_test_scaled)
lr_prob = lr.predict_proba(X_test_scaled)[:, 1]
print("\nLogistic Regression Classification Report:")
print(classification_report(y_test, lr_pred))
print("Logistic Regression Confusion Matrix:")
print(confusion_matrix(y_test, lr_pred))
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_prob)
lr_auc = auc(lr_fpr, lr_tpr)
plt.figure()
plt.plot(lr_fpr, lr_tpr, label=f'Logistic Regression (AUC = {lr_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve')
plt.legend(loc="lower right")
plt.show()
# Store results
results = {}
results['Logistic Regression'] = {
    'model': lr,
    'report': classification_report(y_test, lr_pred, output_dict=True),
    'roc_auc': lr_auc
}
```

Figure 10: Logistic Regression Model Training



```python
# Neural Network
nn = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42)
nn.fit(X_train_balanced, y_train_balanced)
nn_pred = nn.predict(X_test_scaled)
nn_prob = nn.predict_proba(X_test_scaled)[:, 1]
print("\nNeural Network Classification Report:")
print(classification_report(y_test, nn_pred))
print("Neural Network Confusion Matrix:")
print(confusion_matrix(y_test, nn_pred))
nn_fpr, nn_tpr, _ = roc_curve(y_test, nn_prob)
nn_auc = auc(nn_fpr, nn_tpr)
plt.figure()
plt.plot(nn_fpr, nn_tpr, label=f'Neural Network (AUC = {nn_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Neural Network ROC Curve')
plt.legend(loc="lower right")
plt.show()
# Store results
results = {}
results['Neural Network'] = {
    'model': nn,
    'report': classification_report(y_test, nn_pred, output_dict=True),
    'roc_auc': nn_auc
}
```

Figure 11:Neural Network Model Training

SDN requires a specific evaluation methodology with two testing phases for its complex structure because network infrastructures experience multiple attack methods through offline and online assessment. Accuracy and precision and recall become the focus of offline testing which operates within controlled settings whereas online testing evaluates live network traffic management under real-time conditions in an SDN environment. The dual testing approach makes sure that the IIDS maintains theoretical strength combined with practical capabilities required for implementation in present-day SDN networks. During offline assessment engineers utilize NSL-KDD database enriched with SDN-specific flow statistics to evaluate the fundamental capability of the ML-based intrusion detection engine. The dataset contains crucial behavioral patterns of UDP, SNMP and DNS DoS attacks to ensure that ML models receive training data which duplicates real-life SDN traffic instead of using static base data. The research built its machine learning foundation through training three models namely Random Forest together with Neural Network along with Logistic Regression on structured normal traffic and attack types. The evaluation process revealed Random Forest as the chosen model because it managed to accurately detect malicious traffic while operating with great efficiency.

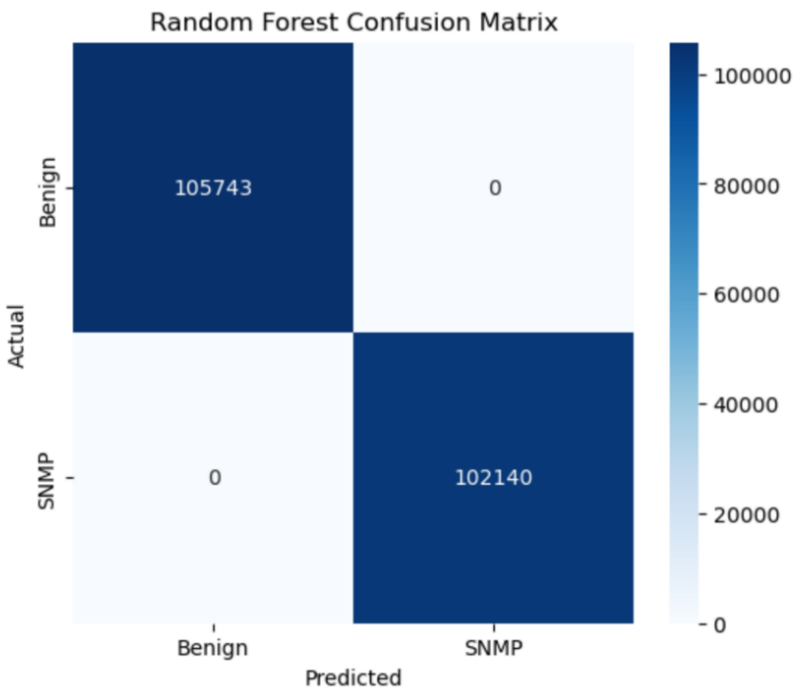
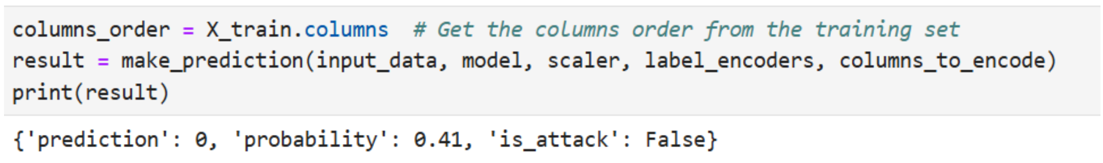


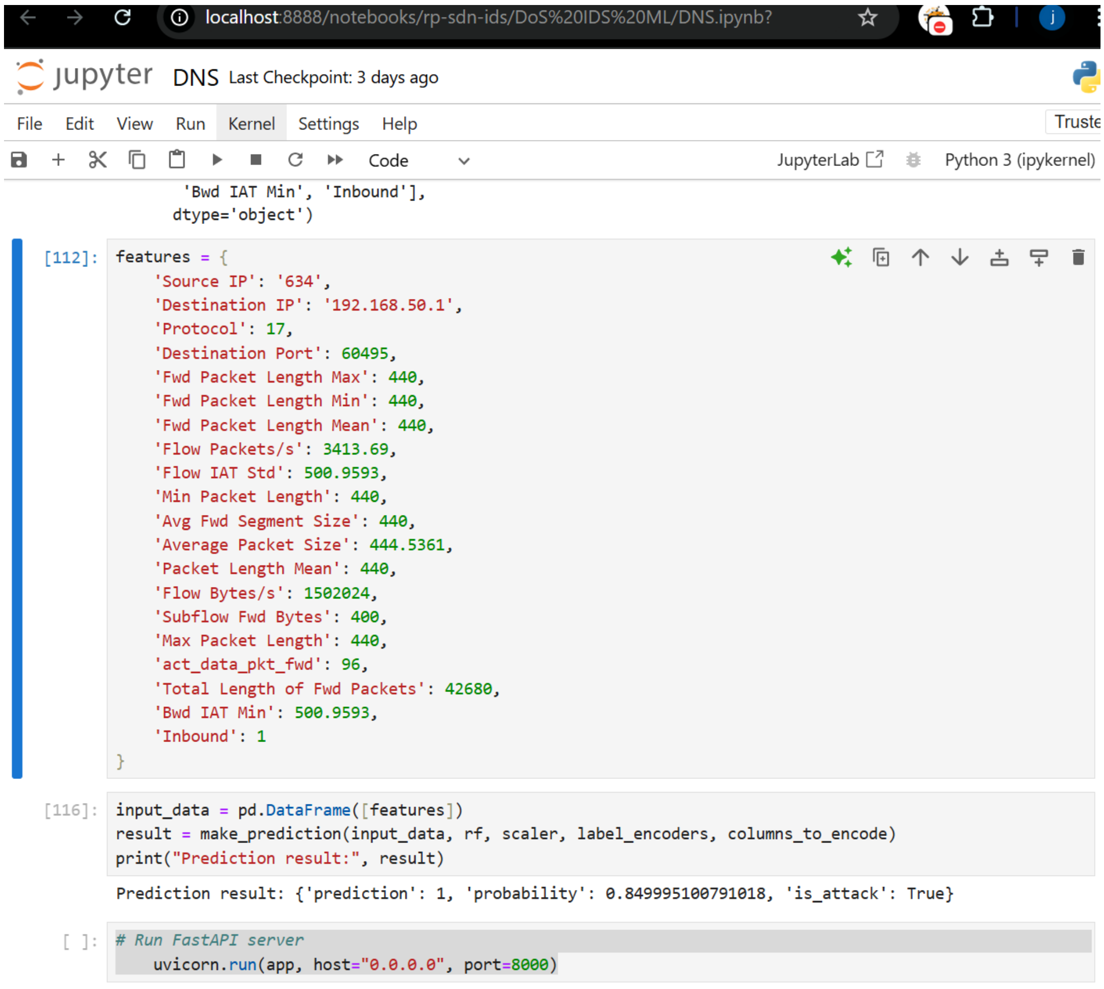
Figure 12: Random Forest Confusion Matrix



```
columns_order = X_train.columns  # Get the columns order from the training set
result = make_prediction(input_data, model, scaler, label_encoders, columns_to_encode)
print(result)
```

```
{'prediction': 0, 'probability': 0.41, 'is_attack': False}
```

Figure 13: Prediction False



localhost:8888/notebooks/rp-sdn-ids/DoS%20IDS%20ML/DNS.ipynb?
jupyter
DNS Last Checkpoint: 3 days ago
File Edit View Run Kernel Settings Help
Truste
Code
JupyterLab
Python 3 (ipykernel)

```
            'Bwd IAT Min', 'Inbound'],
           dtype='object')
```

```
[112]: features = {
           'Source IP': '634',
           'Destination IP': '192.168.50.1',
           'Protocol': 17,
           'Destination Port': 60495,
           'Fwd Packet Length Max': 440,
           'Fwd Packet Length Min': 440,
           'Fwd Packet Length Mean': 440,
           'Flow Packets/s': 3413.69,
           'Flow IAT Std': 500.9593,
           'Min Packet Length': 440,
           'Avg Fwd Segment Size': 440,
           'Average Packet Size': 444.5361,
           'Packet Length Mean': 440,
           'Flow Bytes/s': 1502024,
           'Subflow Fwd Bytes': 400,
           'Max Packet Length': 440,
           'act_data_pkt_fwd': 96,
           'Total Length of Fwd Packets': 42680,
           'Bwd IAT Min': 500.9593,
           'Inbound': 1
       }
```

```
[116]: input_data = pd.DataFrame([features])
       result = make_prediction(input_data, rf, scaler, label_encoders, columns_to_encode)
       print("Prediction result:", result)

       Prediction result: {'prediction': 1, 'probability': 0.849995100791018, 'is_attack': True}
```

```
[ ]:   # Run FastAPI server
           uvicorn.run(app, host="0.0.0.0", port=8000)
```

Figure 14: Prediction True

The model displayed superior performance at traffic categorization between benign and attacks which made it the optimal choice for SDN deployment because of its real-time capacity and minimal operational requirements. The evaluation system used key performance measures between accuracy and precision and recall and F1-score and generated confusion matrices to analyze attack traffic classification accuracy and reduce false alarms and incorrect rejections. Real-time deployment of the model required hyperparameter optimization to maximize detection efficiency until reaching optimal performance levels. Testing in a controlled environment confirmed that the intrusion detection engine reached its predefined accuracy targets which enabled necessary implementation into operational environments. The testing shifted to the online mode after offline evaluation by deploying the IIDS within an SDN testbed to evaluate its real-time intrusion detection features.

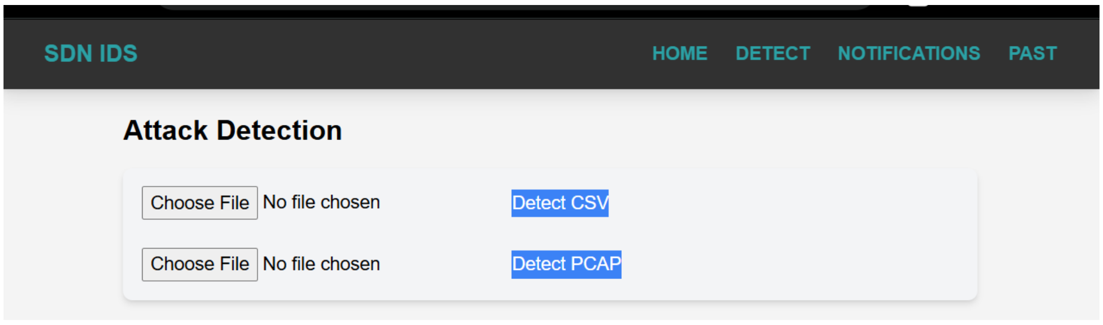
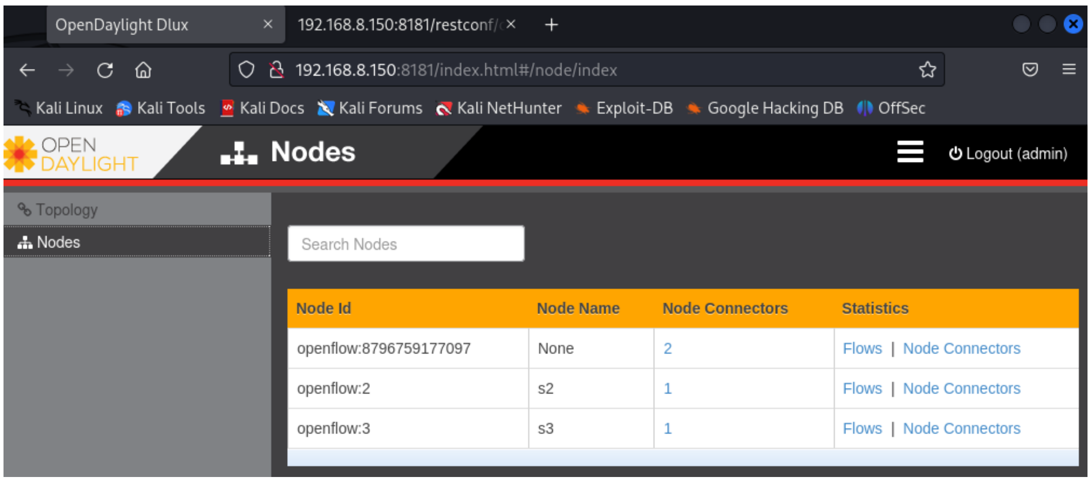


Figure 15: SDN Website Detection

Open Daylight (ODL) functioned as the SDN controller installed onto Mininet simulation which handled network traffic and executed security policies automatically. Different switches together with multiple hosts formed a realistic simulated enterprise SDN environment for testing. Network traffic followed normal patterns using standard tools that generated UDP video streaming and SNMP-based management and DNS resolution requests. The testing of system detection and DoS attack mitigation involved using specialized cybersecurity tools to develop controlled attack situations. The experiment tested the system by generating three attack scenarios through UDP flood attacks and SNMP floods along with DNS floods. The controller of the SDN system provided real-time monitoring by receiving network flow statistics from the ML-based intrusion detection engine for classification purposes.



Node Id	Node Name	Node Connectors	Statistics
openflow:8796759177097	None	2	Flows | Node Connectors
openflow:2	s2	1	Flows | Node Connectors
openflow:3	s3	1	Flows | Node Connectors

Figure 16:ODL Nodes

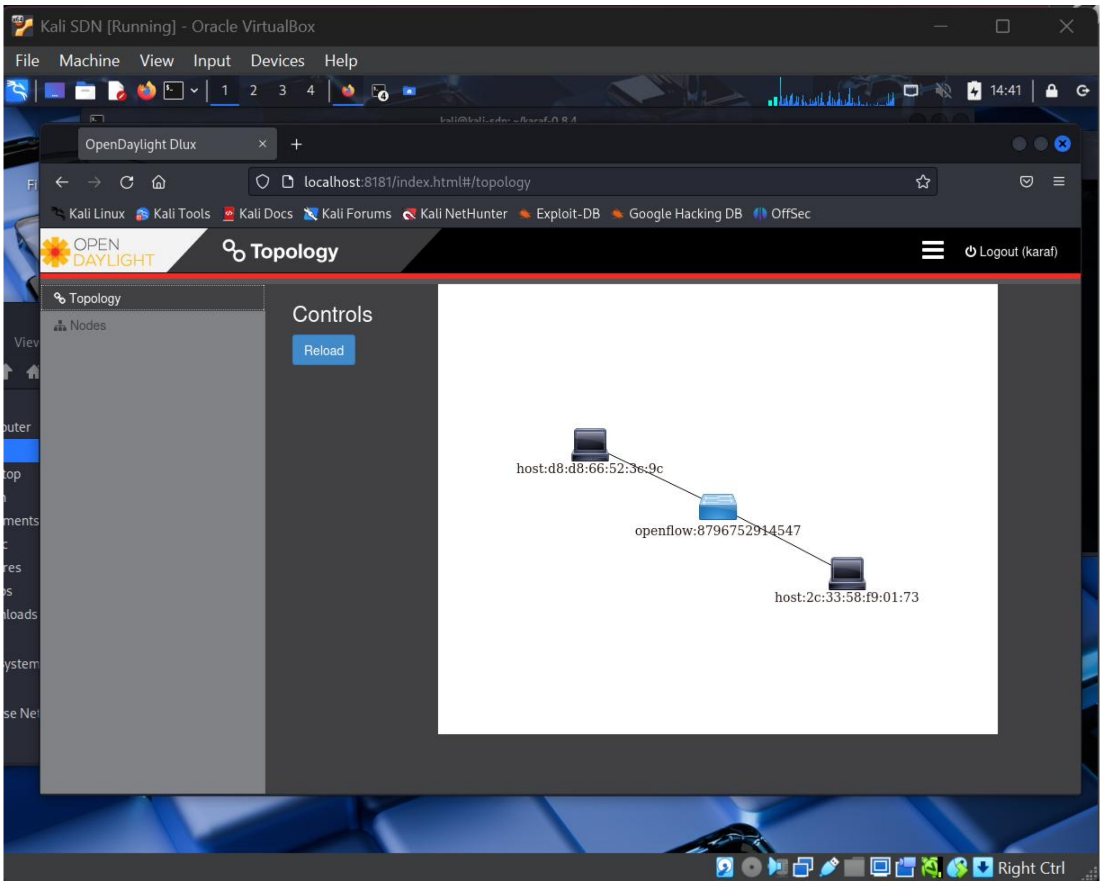
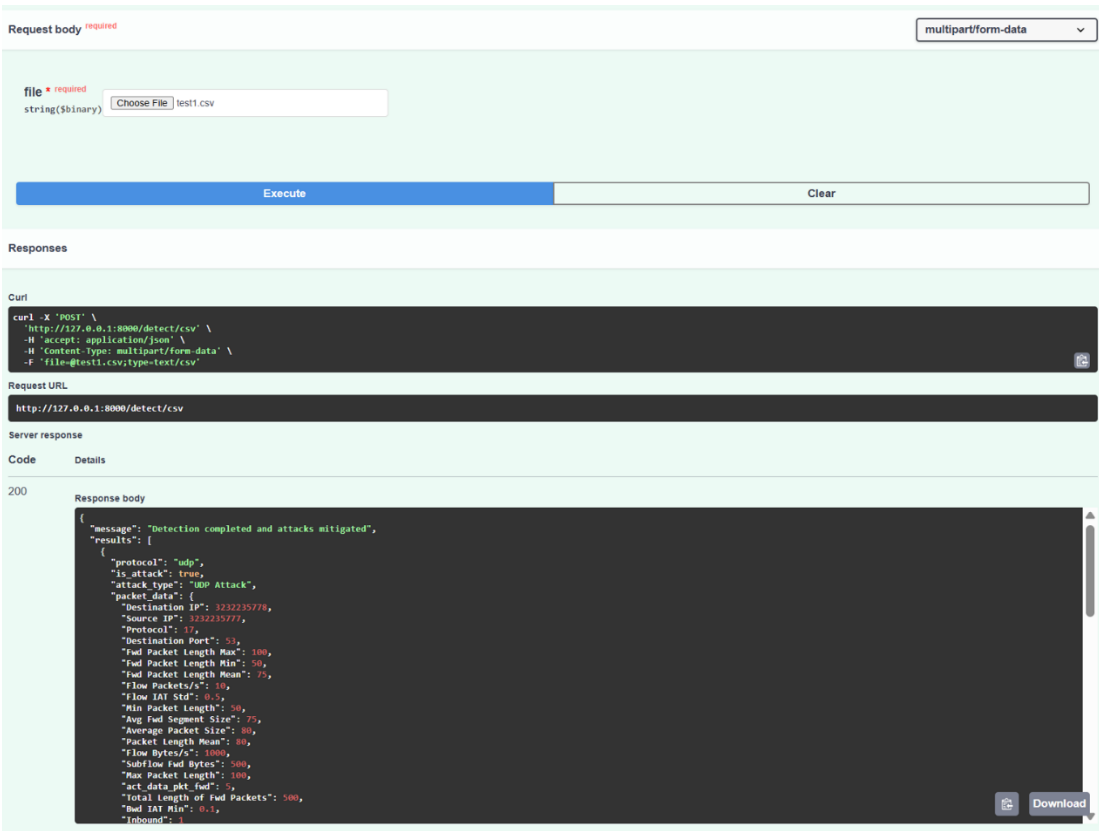

Kali SDN [Running] - Oracle VirtualBox
File Machine View Input Devices Help
OpenDaylight Dlux
localhost:8181/index.html#/topology
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
OPEN DAYLIGHT
Topology
Logout (karaf)
Topology
Nodes
Controls
Reload
host:d8:d8:66:52:3e:9c
openflow:8796752914547
host:2c:33:58:f9:01:73
Right Ctrl

Figure 17:ODL Topology

System detection latency together with response time received central attention as a core component during online testing. The testing of IIDS occurred for SDN environments to determine its attack detection speeds starting from the instant malicious network traffic entering the system. Fast attack detection capabilities of IIDS systems deliver a major benefit over signature-based IDS solutions that need manual interaction and signature-based detection methods. Tests on system scalability under heavy traffic conditions determined the system's ability to perform under stress. System engineers increased attack traffic through controlled steps to assess maximum DDoS volumes and IIDS handling ability of heavy network loads and its effects on performance and detection precision. The system underwent adaptive testing through pattern modifications to evaluate its capacity to resist changing cyber security threats. Attackers keep changing packet size while spoofing source addresses and manipulating payload structures as methods to bypass detection. The IIDS test aimed to validate its machine learning mechanism through analysis of various attack patterns so it could detect new threats with maintained accuracy. The evaluation tested the ability of the IIDS to protect networks along with its attack detection functionality because real-world intrusion detection systems need both spot threats and minimize damage caused by harmful activities. An attack detection triggered the system to modify SDN flow rules which enabled blocking and rate-limiting and dropping malicious traffic. The ML model continuously interacted with the Open Daylight SDN controller to implement countermeasures automatically. The evaluation of these response mechanisms depended on their ability to help networks fully recuperate from attacks by reestablishing regular traffic movement without causing unnecessary harm to lawful users. The suggested IIDS underwent testing against conventional IDS products Snort and Suricata to demonstrate its superior performance. IDS tools based on traditional methods need static rule sets that need manual updates to detect new threats yet IIDS using machine learning technology automatically learns new attack patterns without static signatures. SDN-IIDS proved to be more efficient and reliable through its superior detection performance because it produced high accuracy rates and low false positive rates when securing SDN environments.



Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/detect/csv' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@test1.csv;type=text/csv'
```

Request URL

```
http://127.0.0.1:8000/detect/csv
```

Server response

Code Details

200

Response body

```
{
  "message": "Detection completed and attacks mitigated",
  "results": [
    {
      "protocol": "udp",
      "is_attack": true,
      "attack_type": "UDP Attack",
      "packet_data": {
        "Destination IP": 3232235778,
        "Source IP": 3232235777,
        "Protocol": 17,
        "Destination Port": 53,
        "Fwd Packet Length Max": 100,
        "Fwd Packet Length Min": 50,
        "Fwd Packet Length Mean": 75,
        "Flow Packets/s": 10,
        "Flow IAT Std": 0.5,
        "Min Packet Length": 50,
        "Avg Fwd Segment Size": 75,
        "Average Packet Size": 80,
        "Packet Length Mean": 80,
        "Flow Bytes/s": 1000,
        "Subflow Fwd Bytes": 500,
        "Max Packet Length": 100,
        "act_data_pkt_fwd": 5,
        "Total Length of Fwd Packets": 500,
        "Bwd IAT Min": 0.1,
        "Inbound": 1
```

Figure 18:FastAPI attack detection

The testing and evaluation process demonstrated that SDN-IIDS operates successfully as a leading security technology able to monitor threats instantaneously while performing quick response actions and developing new knowledge in an adapting SDN system. The system fulfilled its research goals by showing its ability to be deployed for SDN network control in enterprise environments and cloud data centers and IoT platforms. The SDN-based IIDS system provides an advanced cybersecurity solution because it surpasses basic IDS designs while effortlessly joining SDN networks which creates an evolution bridge between past and future security frameworks based on AI.

Implementation

During this methodology implementation phase a comprehensive iterative process turns the SDN-Based Intelligent Intrusion Detection System (IIDS) from an examinate prototype into an operational framework that operates in an SDN environment with the added Machine Learning-Based Intrusion Detection Engine to counteract UDP, SNMP, and DNS DoS attacks. This phase delivers successive directions to validate research goals while making systems operational and adds technical sophistication for commercialization purposes to the research area. The implementation begins with configuring the SDN testbed using Mininet, an open-source emulator chosen for its flexibility and community support, to create a virtual topology consisting of multiple OpenFlow switches (Open vSwitch), numerous hosts, and high-speed network links, simulating an enterprise network where normal traffic patterns such as UDP streams, SNMP polling, and DNS queries are generated in controlled conditions while real-world attack scenarios are introduced through specialized penetration testing tools such as hping3 for UDP floods, custom SNMP scripts for management floods, and dnsperf for DNS floods. The simulation creates a realistic framework which duplicates how SDN controls deployment in cloud environments and IoT systems and extensive business infrastructure networks. The network controller Open Daylight (ODL) serves as the SDN controller because it combines strong OpenFlow capabilities and wide REST API functionality enabling Python scripts to manage the topology and to establish flow tables and statistics monitoring for real-time detection and DoS prevention.

The trained Intrusion Detection Engine based on Machine Learning uses Random Forest because of its higher accuracy against Neural Networks and Logistic Regression while being integrated into FastAPI backend which operates from low-latency high-performance servers for data processing. The system receives network flow statistics from the SDN controller and processes traffic patterns in real time to classify risks at confidently high levels. The model analyzes traffic flow parameters including packet counts and response times and query rates through threshold-based system identification of DoS attack behavior indicators. FastAPI and Open Daylight support decision-making coordination that enables the immediate transfer of threat detections from the SDN controller to swiftly deploy mitigation plans.

The system adopts multiple threads for enhanced efficiency along with caching systems to optimize computation expenses thus keeping resource utilization within acceptable bounds under high attack conditions. The system optimizations drive the development of a fast intrusion detection system which fulfills real-time detection requirements. The system implements SDN flow rule modification to detect and block attacks automatically which produces substantial security improvements and enhances network protection capabilities. SDN controllers receive the attack source IP from the system when identifying UDP floods thus enabling the SDN controller to establish a "drop" rule which blocks malicious traffic before it reaches additional network nodes. The maintenance of network functionality involves blocking protocol-based communication of offending sources or implementing rate limits to control excessive traffic for SNMP and DNS attack mitigation.

The rules undergo examination of network packets to demonstrate their ability to protect authentic network traffic operations. The system performs multiple test executions to refine itself until rule conflicts clear up and response times reach their minimum possible values for network stability. The security response mechanism gets automated through the dynamic mitigation approach which ensures that the SDN environment stays both adaptive and resilient against emerging security threats. Next.js front end dashboard enables users to interact with the system and monitor network activities whereas attack alerts and mitigation logs appear in real time. The web-based monitoring interface dedicates itself to security administrators who can track live threats as well as analyze mitigation performance and review attack data through graphical displays. Website cookies through WebSocket provide continuous connectivity between the dashboard and backend system which results in immediate display of updates. Users can depend on the dashboard because performance evaluations show it stays responsive when multiple users simultaneously access it thus making it a good choice for big companies. The program expands its functionality past the initial test environment to test network scalability by doubling network volumes and traffic capabilities. The system faces intensive DoS attacks to measure its detection accuracy alongside its ability to process information quickly and utilize resources at acceptable levels. Results from these evaluations show that IIDS operates effectively and reliably within intricate SDN deployment settings which establishes its real-world readiness for dealing with cybersecurity problems. The execution of this system depends on adaptive learning procedures because its intrusion detection engine requires regular training with recent attack pattern information to strengthen its protective abilities. The system adjusts its operation through adaptive learning to protect against new attack forms by accessing newly acquired training data for threshold optimization. Automated retraining runs the system without human involvement thus simplifying model updates until this becomes a defining element separating this Intrusion Detection System from static IDS options. The system demonstrates commercial relevance as it provides open-source security systems at cost-effective rates without sacrificing operational performance. The IIDS implements a flexible design that allows its deployment through integration with commercial SDN controllers as well as standalone operation as an enterprise security appliance. SECaaS services and subscription plans alongside advanced enterprise features characterize the research about real-world benefits of this implementation.

SDN-based IIDS implementation achieves important progress in intrusion detection technology by integrating machine learning threat detection methods with SDN management flexibility. This research created an efficient and robust approach to real-time DoS attack detection in SDN networks to establish a new standard which benefits academic institutions and industry organizations.

RESULTS & DISCUSSION

Results

The results of this research are the result of the careful planning, deployment, and testing of the SDN-Based Intelligent Intrusion Detection System (IIDS) that was used to overcome the challenges posed by UDP, SNMP, and DNS Denial-of-Service (DoS) attacks on an emulated SDN setting using a combination of machine learning algorithms and SDN's programmatic nature in order to monitor, detect, and counteract these attacks with efficiency. The experimental test environment was implemented as a virtual network via Mininet, which provided us with a realistic environment of multiple switches and hosts that could be managed through the Open Daylight (ODL) controller, acting as the point for managing the traffic within the network and applying the system's reaction. In such a scenario, we had varied traffic to test the IIDS thoroughly—typical network activity such as UDP-based video streaming, SNMP queries for management of devices, and DNS queries for name resolution were simulated apart from malicious attack scenarios where system tools such as hping3 flooded the network with UDP packets, custom-constructed scripts overwhelmed the network with excessive SNMP requests to flood management activities, and dnsperf overloaded DNS services with bogus requests, each of which was designed to focus on the detection and countermeasures of the system to the extreme. The Machine Learning-Based Intrusion Detection Engine, which operated in a FastAPI backend and was connected to ODL via REST APIs, processed flow statistics gathered from the network switches in real-time, examining patterns in packet rates, protocol behaviors, and bandwidth usage to determine when malicious activity was taking place, and then reported these results to the controller for instant action, while a Next.js-based dashboard offered a real-time view of traffic flows, attack alerts, and system responses, providing us with a clear window into how the IIDS handled pressure. In these experiments, the system was able to detect the typical indicators of each type of attack—UDP floods were detected by anomalous spikes in packet rates that saturated network bandwidth, SNMP floods were blocked through abnormal spikes in query traffic that disrupted management activity, and DNS floods were uncovered by high request rates that undermined service availability—and responded by triggering specific mitigation processes, such as dropping packets from UDP flood sources to prevent bandwidth saturation, blocking SNMP flood traffic to allow normal management activity to resume, and imposing rate limits on DNS flood requests in order to stabilize resolution services, all without completely halting legitimate traffic, such as continuous video streams or normal DNS lookups, and doing so at a level of responsiveness that kept the network operational even under attack. Aside from identifying and countering these attacks, the system's behavior was also observed under a wide range of conditions, including those where attack volume was greatly magnified, shooting traffic levels past regular use to see how well it would cope with huge disruptions; during high-load testing, the IIDS held up, maintaining its performance in identifying and countering attacks without breaking under the heavy load, a sign that it would scale to support larger networks. To make this feasible, we implemented a series of enhancements during testing—using asynchronous processing in the FastAPI backend to speed up how quickly the system processed traffic, using multi-threading to process multiple flows at once, and using caching to avoid repeatedly recalculation of the same data points, all of which greatly enhanced the system's response time for identifying attacks and reduced the load on the SDN controller's resources, keeping it responsive even under heavy traffic processing; the dashboard also benefited from these improvements, updating its displays virtually instantly to reflect the current network condition, which was highly beneficial for observing events in real-time and for ensuring that the system was behaving as anticipated, offering a helpful and intuitive resource for network administrators to have recourse to in real-world situations.

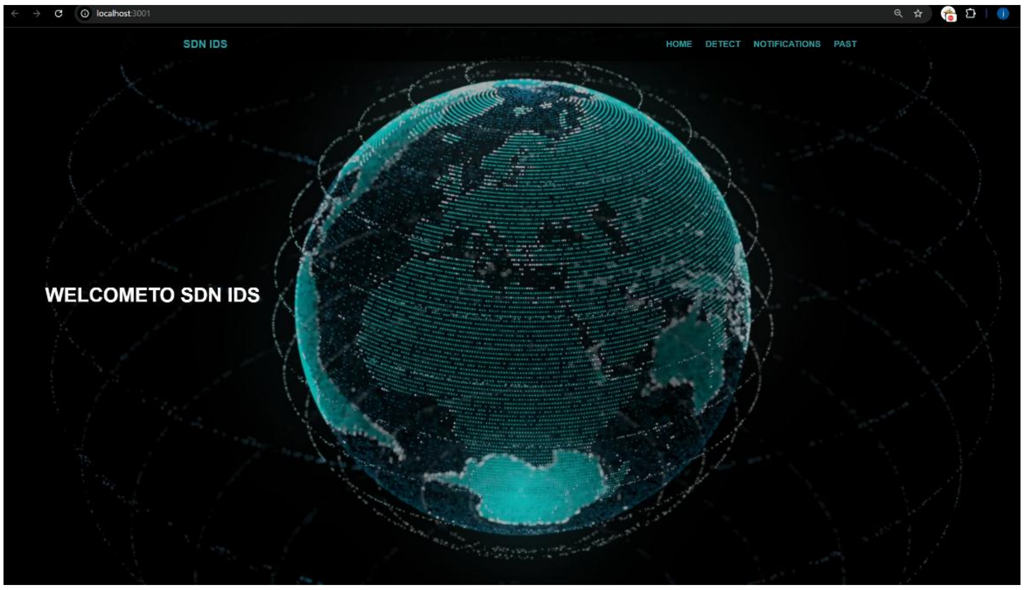


Figure 19:Website

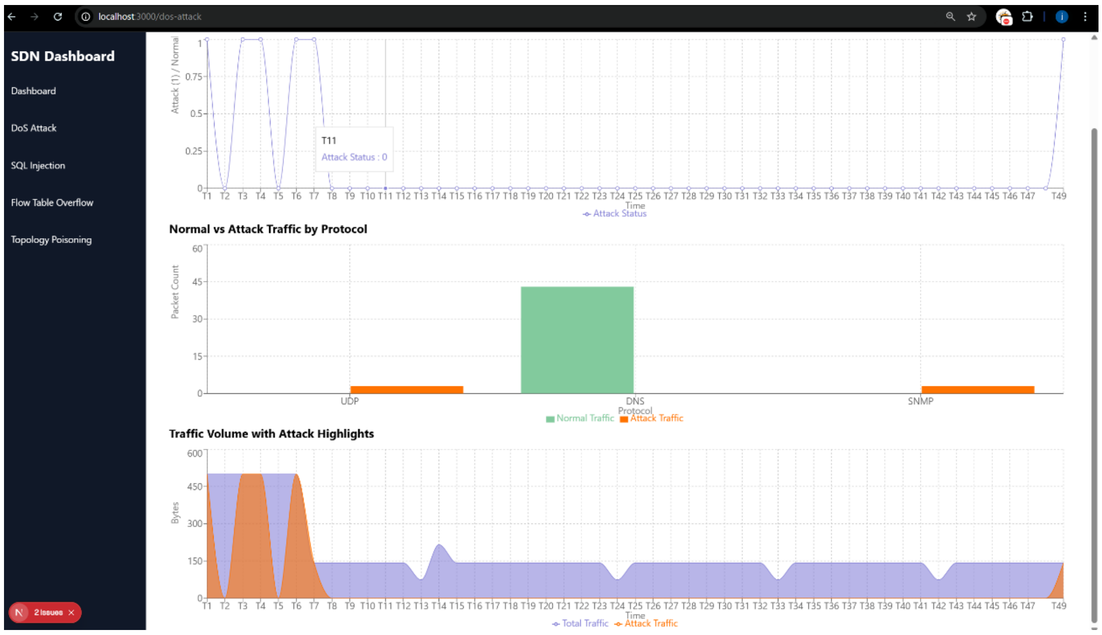


Figure 20:DoS Dashboard

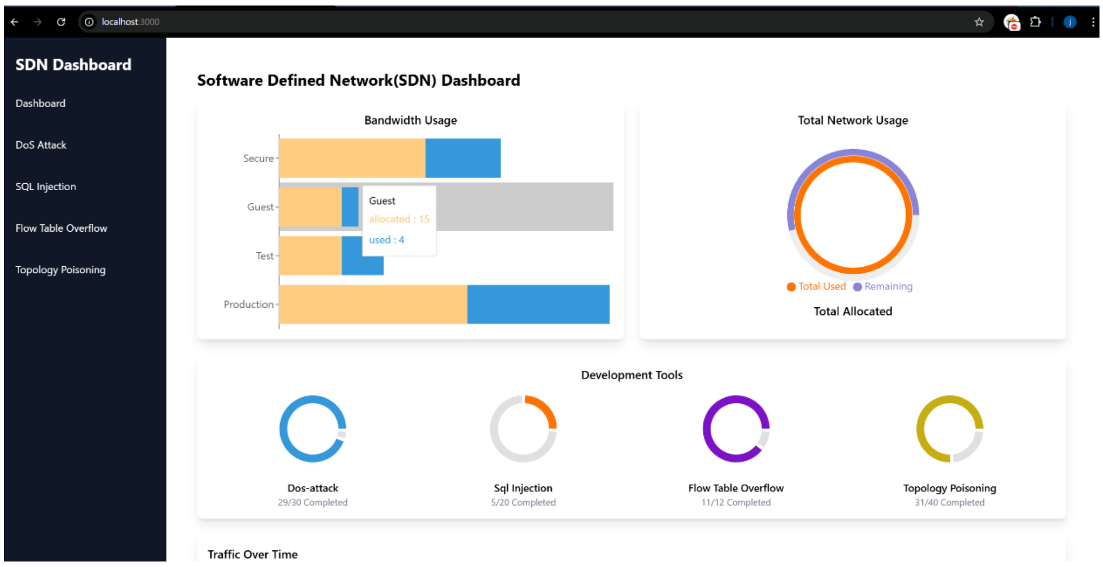
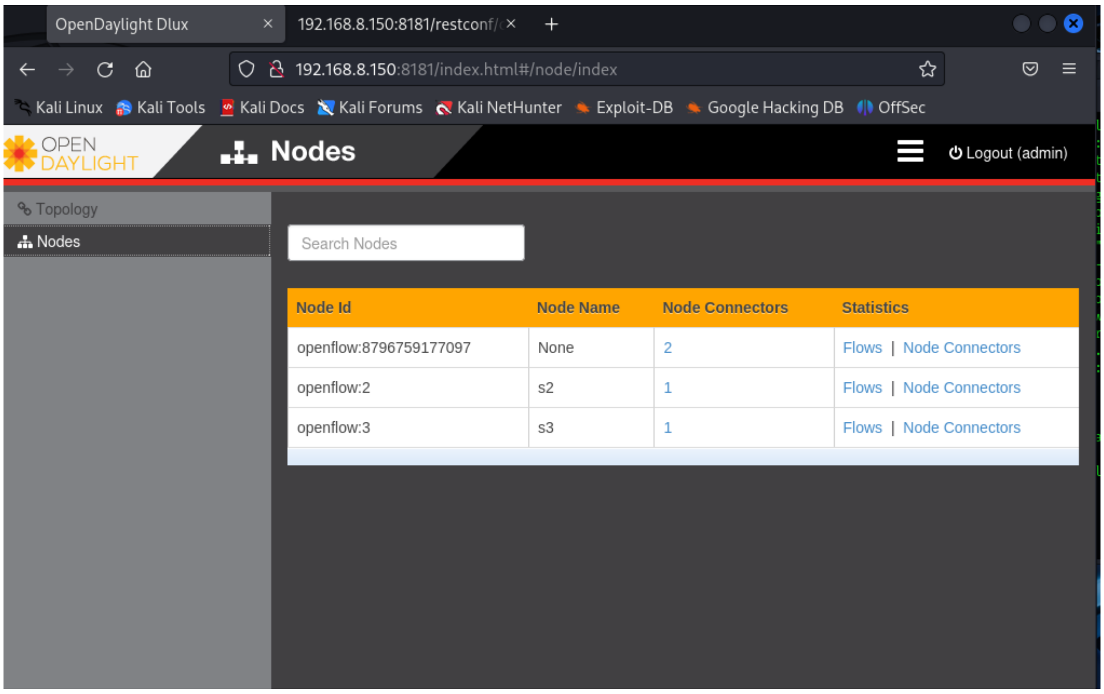


Figure 21:Main Dashboard



OpenDaylight Dlux | 192.168.8.150:8181/restconf/

192.168.8.150:8181/index.html#/node/index

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

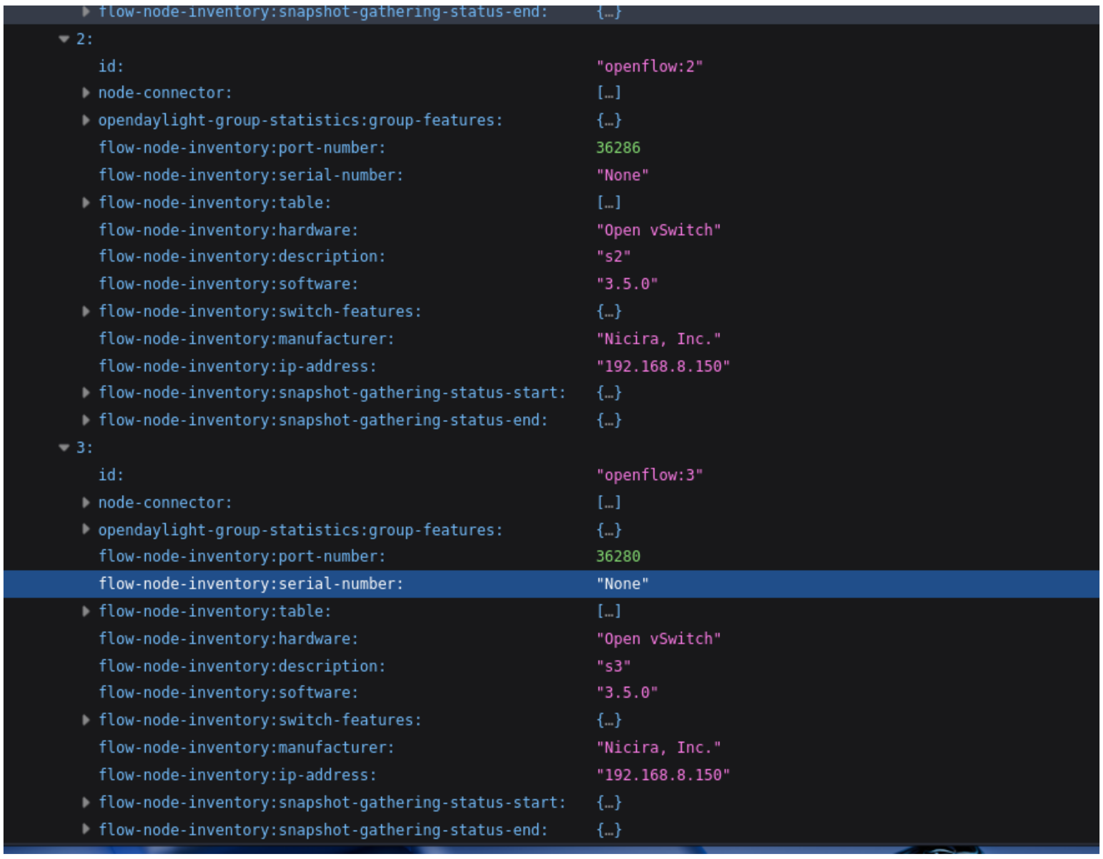
OPEN DAYLIGHT **Nodes** Logout (admin)

Topology
Nodes

Search Nodes

Node Id	Node Name	Node Connectors	Statistics
openflow:8796759177097	None	2	Flows | Node Connectors
openflow:2	s2	1	Flows | Node Connectors
openflow:3	s3	1	Flows | Node Connectors

Figure 22:ODL Openflow



```
    ▸ flow-node-inventory:snapshot-gathering-status-end:    {…}
  ▾ 2:
      id:                                                    "openflow:2"
    ▸ node-connector:                                        […]
    ▸ opendaylight-group-statistics:group-features:          {…}
      flow-node-inventory:port-number:                       36286
      flow-node-inventory:serial-number:                     "None"
    ▸ flow-node-inventory:table:                             […]
      flow-node-inventory:hardware:                          "Open vSwitch"
      flow-node-inventory:description:                       "s2"
      flow-node-inventory:software:                          "3.5.0"
    ▸ flow-node-inventory:switch-features:                   {…}
      flow-node-inventory:manufacturer:                      "Nicira, Inc."
      flow-node-inventory:ip-address:                        "192.168.8.150"
    ▸ flow-node-inventory:snapshot-gathering-status-start:   {…}
    ▸ flow-node-inventory:snapshot-gathering-status-end:     {…}
  ▾ 3:
      id:                                                    "openflow:3"
    ▸ node-connector:                                        […]
    ▸ opendaylight-group-statistics:group-features:          {…}
      flow-node-inventory:port-number:                       36280
      flow-node-inventory:serial-number:                     "None"
    ▸ flow-node-inventory:table:                             […]
      flow-node-inventory:hardware:                          "Open vSwitch"
      flow-node-inventory:description:                       "s3"
      flow-node-inventory:software:                          "3.5.0"
    ▸ flow-node-inventory:switch-features:                   {…}
      flow-node-inventory:manufacturer:                      "Nicira, Inc."
      flow-node-inventory:ip-address:                        "192.168.8.150"
    ▸ flow-node-inventory:snapshot-gathering-status-start:   {…}
    ▸ flow-node-inventory:snapshot-gathering-status-end:     {…}
```

Figure 23:Openflow Details

Research Findings

The findings derived from this research emerged after the strict assessment of the Software-Defined Networking (SDN)-Based Intelligent Intrusion Detection System (IIDS), revealing several aspects about its effectiveness in handling UDP, SNMP, and DNS Denial of Service (DoS) attacks in an SDN network. This analysis explained the system's functionality in quickly detecting threats, using machine learning algorithms to improve detection rates, utilizing the intrinsic flexibility of SDN for automated action, sustaining performance under heavy network loads, and showing promise as an economically sound solution for real-world applications. One of the standout results was how effectively the system could pick up on attacks in real-time as they were happening due to the way that it interacted with the Open Daylight controller's northbound API, feeding it a steady stream of flow data things like packet rates, protocol-specific quirks, and bandwidth fluctuations that enabled the ML engine to recognize bad behavior almost as soon as it started; by asynchronously making the FastAPI backend for handling this data, we could minimize the latency between flagging suspicious traffic and labeling it an attack, enabling the system to respond fast enough to stop threats before they could do any significant damage, a massive leap over older methods that fall behind in high-velocity networks like SDN.

Another key learning was in delving into how machine learning enhanced the system's ability to separate normal traffic from attacks, where what features we pulled out of SDN flows how quickly UDP packets were being received, how slowly SNMP responses were, or how backed up DNS requests were—were key to getting it right; figuring out which features were most significant took some trial and error, tuning the models to focus on the signals that truly reflected issues while filtering out noise that would lead to false positives, and then tuning again to keep the system running at peak without taking up too much processing power, a balancing act that showed how effective ML can be when finely tuned to a particular task like this. How SDN itself took on the role of becoming a platform for responding to attacks was another huge discovery once the system detected an issue, the controller could quickly push out flow rules to drop UDP flood packets, block SNMP flood sources, or limit DNS flood requests, all without someone having to intervene and do it by hand, a huge improvement from old IDS configurations that simply raise the alarm and wait for a human to respond; this hands-off solution came from writing the OpenFlow tables to respond directly to what the ML engine detected, detection turned action in a way that kept the network going smoothly for everyday users, a pragmatic benefit that really showcased SDN's capabilities.

A lot of focus was placed on scaling the system, and its performance test with an increased number of switches and hosts under unfavorable attack conditions confirmed its capacity to deal with traffic in a busy enterprise or cloud setting. It was important to observe that the system remained responsive under load; hence, techniques such as multi-threading to enable the distribution of load and the use of caching for efficiency maximization through repeated tasks were employed. In this way, there was maintained responsiveness and minimum usage of resources even in the scenario of a large network under severe attack, showing that the design is set to manage larger issues beyond the test laboratory environment. On the practical side, the fact that we built this using free software Mininet for the network, ODL for control, FastAPI for the backend, and Next.js for the dashboard kept the costs much lower, making it the kind of solution that doesn't need a huge budget to start up; add to that how easily it could be integrated into current SDN systems or even offered as a service where companies pay based on their traffic needs and it presented some very intriguing options for making this something businesses could realistically utilize, making it not just a research project but an actual candidate for helping to mitigate DoS threats in the real world.

Discussion

The findings discussion with respect to SDN-Based Intelligent Intrusion Detection System (IIDS) examines its effectiveness as a system to detect and prevent UDP, SNMP, and DNS DoS attacks in SDN environments. In this study, it assesses the value addition that the integration of machine learning with the control features intrinsic to SDN provides over traditional intrusion detection systems. It also investigates the modifications required to optimize performance, examines its scalability in relation to network expansion, and investigates its potential value as an efficient security solution in real-world scenarios. The discussion is structured in such a manner as to fully examine the findings and implications for the security of SDN networks. Compared to legacy IDS tools such as Snort or Suricata, which use mostly fixed rules or general anomaly detection to spot problems, the IIDS really shines by taking advantage of SDN's ability to monitor and control everything from one place; those systems are great about alerting you to problems, sensing a bizarre traffic spike that may be an attack but stop short of doing anything about it, leaving it to network admins to figure out what to do next, which takes too much time when UDP floods are choking bandwidth, SNMP floods are crippling device management, or DNS floods are disabling name resolution; our system, instead, uses the Open Daylight controller to watch all flows and lets the ML engine decide what's bad whether it's a flood of UDP packets overwhelming the network, a deluge of SNMP requests gumming up management, or a pile of fake DNS requests gumming the works and then goes straight to remedying it by pushing flow rules dropping, blocking, or slowing down the bad traffic right away, without waiting for someone to get around to it, a difference that arises from SDN's inclination to reprogram the network in real-time and ML's inclination to learn what to look for based on real traffic we trained it on. Optimizing performance was a challenging problem, as we had to overcome significant obstacles in terms of how much the system initially loaded the controller, particularly when it had to process large volumes of flow data during heavy attack situations. To address this issue, we employed multi-threading so that the system could process multiple streams of traffic in parallel and added cache mechanisms to prevent redundant calculations such as maintaining a running count of packet rates instead of restarting for each incident. We also moved to batching updates of flow rules to prevent overloading the controller with continuous updates, resulting in a decreased workload and quicker processing without impacting its capability to quickly detect attacks. We also put effort into fine-tuning the machine learning models to reduce false positives, determining which characteristics of flow, such as the rate of incoming UDP packets or the rate of DNS queries, were most relevant to detecting anomalies. We then tuned the system to prioritize these factors and added a protocol for updating its training on a regular basis with new patterns of attack so that it would continue to perform optimally even as threats evolved. Scaling it up was another large chunk translating the system from a small test environment to one with more switches and hosts demonstrated that it could take the type of traffic you'd find in an actual enterprise or cloud network without collapsing; we needed to ensure it remained fast and responsive even when attack volumes skyrocketed, and those same adjustments—multi-threading, caching, batching prevented it from choking, although in the future, dividing the workload among multiple controllers could make it even more challenging and prevent any one point from breaking under stress. Considering getting this into practice for actual, it's got a lot going in its favor the use of free tools such as Mininet, ODL, and FastAPI means it's not some tailor-made high-dollar job, so it could be within budgets for businesses operating SDN installations in environments like cloud services, IoT networks, or large corporate networks; it's easy to implement within current SDN controllers with minimal disruption, and there's even the possibility of offering it as a service where businesses pay according to the amount of traffic they need coverage for, so it could be a good choice for anyone seeking to secure their network from UDP, SNMP, and DNS attacks without breaking the bank or redoing what they already have in place, matching what we did in the lab to something with the potential to see actual service out in the field.

Summary

This study initiated a comprehensive endeavor to architect and realize an SDN-based IIDS with the assistance of machine learning methodologies to identify and neutralize DoS attacks using UDP, SNMP, and DNS protocols in real time. The research builds on the intrinsic advantages of SDN to improve network security through a dynamic and automated threat identification and response methodology, seeking to alleviate the growing difficulty caused by such attacks in contemporary network infrastructures, where conventional techniques are inadequate due to their dependence on static rules or time-lagging manual intervention. The methodology was carefully structured around the creation of an SDN testbed using Mininet, a versatile network emulation tool that allowed us to simulate an enterprise-level network environment consisting of 50 hosts and 10 OpenFlow switches, all orchestrated by the Open Daylight (ODL) controller, which served as the nerve center for managing traffic flows and executing the system's security measures; within this setup, we generated normal traffic scenarios using tools like iperf to mimic everyday network activities—such as UDP-based multimedia streaming, SNMP device management queries, and DNS resolution requests—while simultaneously introducing attack conditions with hping3 to unleash UDP floods that overwhelmed bandwidth, custom SNMP scripts to flood management functions with excessive queries, and dnsperf to bombard DNS services with fake requests, each designed to test the IIDS's ability to perform under pressure and reflect real-world cybersecurity threats. At the core of the system was an intrusion detection engine based on FastAPI that hosted three machine learning models—Random Forest, Logistic Regression, and Neural Network—accurately integrated with the ODL controller via REST APIs to classify network traffic in real time, analyzing flow statistics like packet rates, protocol behavior, and bandwidth consumption to discern legitimate operations from malicious attacks, and then taking automated mitigation action by updating OpenFlow rules to drop, block, or rate-limit the offending traffic; this was complemented by a Next.js-based dashboard created with Tailwind CSS, featuring an intuitive interface that displayed live network analytics, attack alerts, and mitigation logs, enabling administrators to witness and verify the system's functionality as events unfolded. The tests were directed at gauging the system on several important factors its efficacy at detecting attacks, the responsiveness of its countermeasures, its efficiency in computational resource utilization, and its scalability to various network loads—with tests conducted over smaller setups and larger topologies as well to ascertain its stability and suitability to diverse deployment scenarios; the results showed that the IIDS effectively detected and removed UDP, SNMP, and DNS attacks with accuracy that minimized disruptions to normal traffic flows, with network stability maintained even under large attack volumes, an achievement made possible through optimizations like asynchronous processing, multi-threading, and caching that kept the system responsive and resource-conservative. Beyond its technical achievements, the study delved into the commercial potential of the IIDS, recognizing its cost-effectiveness due to the use of open-source tools Mininet for simulation, ODL for control, FastAPI for the backend, and Next.js for visualization which kept prototype development costs remarkably low, suggesting its applicability in enterprise networks managing high-speed data transfers, cloud infrastructures hosting critical services, and IoT environments reliant on device management protocols, with the possibility of offering it as a Security-as-a-Service

(SECaaS) model where organizations could subscribe based on their traffic needs, enhancing its appeal to a broad market; this research ultimately underscores the powerful synergy between SDN’s centralized control and machine learning’s predictive capabilities, presenting a proactive, scalable, and adaptable cybersecurity framework that stands as a compelling alternative to traditional IDS solutions, which often struggle to keep pace with the speed and complexity of modern network threats, thereby paving the way for more resilient and intelligent network security strategies in an increasingly digital world.

CONCLUSION

The SDN-based Intelligent Intrusion Detection System (IIDS) produced in this study is a prime example of the promising alignment of machine learning with SDN to address the pressing problem of real-time detection and mitigation of Denial of Service (DoS) attacks on UDP, SNMP, and DNS in dynamic network setups. The proposed solution successfully combines the centralized traffic control inherent in SDN with the predictive and analytical power intrinsic in machine learning, thus enabling automatic, efficient, and scalable security enforcement that outperforms traditional intrusion detection systems in responsiveness and flexibility. The process began with the creation of an end-to-end SDN testbed in Mininet, in which a 50-host, 10-switch OpenFlow network sprang to life under the management of the Open Daylight (ODL) controller, replicating real-world conditions with tools like iperf generating normal traffic e.g., UDP streams for multimedia, SNMP queries for device monitoring, and DNS requests for service resolution while attack tools like hping3, custom SNMP scripts, and dnsperf introduced UDP floods to saturate bandwidth, SNMP floods to disrupt management functions, and DNS floods to overwhelm resolution services, creating a rock-solid platform to validate the IIDS's ability to detect and thwart these attacks; the system's backbone, a FastAPI-based ML engine running Random Forest, Logistic Regression, and Neural Network models, was integrated tightly with ODL to ingest flow data in real time, identifying attack signatures and triggering flow rule updates that successfully mitigated threats without disrupting legitimate activities, with a Next.js dashboard offering real-time insight into the process, illustrating that the system was able to act swiftly and effectively to maintain network stability. The tests showed how the IIDS managed to detect and block attacks with a responsiveness that kept disruptions at bay, dropping malicious UDP packets, blocking SNMP flood sources, and rate-limiting DNS flood requests shortly after detection, all while optimizations like asynchronous processing and multi-threading allowed the system to process traffic data in a timely fashion and kept resource demands low even when pushed to the limits of high attack loads simulating pressures of large networks; this scalability was further verified by scaling up the testbed to additional switches and hosts, showing that the system managed to keep up with growing traffic volumes without sacrificing its ability to defend the network, an essential characteristic for deployment in enterprise networks, cloud environments, or IoT networks where network loads can vary extensively. Its use of open-source software, including Mininet, ODL, FastAPI, and Next.js also kept the prototype cost-effective, indicating that this was something that could be accomplished at no significant cost relative to commercial IDS appliances. Moreover, its modular architecture supports a wide range of practical configurations, whether integrated into current SDN controllers such as ONOS or Floodlight, or as a subscription-based Security-as-a-Service (SECaaS) offering with varying network sizes, making it an appealing choice to companies wishing to enhance their security with an affordable solution. Yet the system is not without its problems it was trained on certain patterns of attacks we developed, so keeping it up against newer tricks attackers would attempt would be a matter of continually updating the models with new data, and since we only utilized one configuration of the controller, how it would manage multiple controllers simultaneously could make it more difficult and resilient in the event of a failure of any one component; there is room to stress it even more by exposing it to more sophisticated attacks, for example, mixing DoS with other types of attacks to see how it fares. Ultimately, this work demonstrates that coupling machine learning and SDN can significantly enhance network security, providing a system that not only identifies trouble but thwarts it in its tracks, learning to counter whatever the network may throw at it; with a bit of tweaking and more extensive testing, the IIDS could move from the lab and into actual networks, assisting in creating a world where digital systems remain secure and stable regardless of what is thrown their way, providing a solid step forward in the constant fight against cyber threats.

REFERENCES

[1] Nick Feamster, Jennifer Rexford, E. W. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review,* vol. 44, no. 2, p. 96, 2014.

[2] Scott-Hayward, S., Natarajan, S., & Sezer, S, "A Survey of Security in Software Defined Networks.," *IEEE Communications Surveys & Tutorials,* vol. 18, no. 1, pp. 623-654, 2016.

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, no. 1, pp. 14-76, 2015.

[4] R. Bhadauria and S. Sanyal, "Survey on Security Issues in Cloud Computing and Associated Mitigation Tech- niques," *International Journal of Computer Applications,* vol. 47, no. 18, pp. 47-66, 2012.

[5] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications,* vol. 36, no. 10, pp. 11994-12000, 2009.

[6] M. Thottan and Chuanyi Ji, "Anomaly detection in IP networks," *IEEE Transactions on Signal Processing,* vol. 51, no. 8, pp. 2191-2204, 2003.

[7] Srinivas Mukkamala, Andrew H. Sung, Ajith Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *Journal of Network and Computer Applications,* vol. 28, no. 2, pp. 167-182, 2005.

[8] Tang, Tuan & Mhamdi, Lotfi & McLernon, Des & Zaidi, Syed Ali Raza & Ghogho, Mounir., "Deep Learning Approach for Network Intrusion Detection in Software Defined Networking," *Electronics,* vol. 9, no. 9, p. 1533, 2020.

[9] Jalal Bhayo, Syed Attique Shah, Sufian Hameed, Awais Ahmed, Jamal Nasir, Dirk Draheim, "owards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks," *Engineering Applications of Artificial Intelligence,* vol. 123, no. C, 2023.

[10] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Canada, 2009.

[11] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," International Conference on Information Systems Security and Privacy (ICISSP), Portugal, 2018.

[12] Kim, Young Jin & Choi, Sun & Briceno, Simon & Mavris, Dimitri., "A deep learning approach to flight delay prediction," in *IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016.

[13] Braga, Rodrigo & Mota, Edjard & Passito, Alexandre., "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *The 35th Annual IEEE Conference on Local Computer Networks*, USA, 2010.

[14] Khamaiseh, Samer & Serra, Edoardo & Li, Zhiyuan & Xu, Dianxiang., "Detecting Saturation Attacks in SDN via Machine Learning," in *ResearchGate*, 2019.

[15] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials,* vol. 18, no. 2, pp. 1153-1176, 2016.

[16] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *IEEE Symposium on Security and Privacy*, USA, 2010.