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

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

We declare that this is our own work and this dissertation1 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, we 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. We retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 11.04.2025

Signature of the supervisor: Date:

ABSTRACT

By separating the control plane from the data plane and allowing for centralized network control and dynamic configuration, Software-Defined Networking (SDN) has revolutionized traditional networking. Although this change in architecture offers more programmability and flexibility, it also exposes the network to new security risks such denial-of-service (DoS) attacks, flow manipulation, and topology poisoning. An SDN-Based Intelligent Intrusion Detection System (IIDS) combined with machine learning techniques is presented in this study to address these vulnerabilities by detecting and mitigating network anomalies in real time.

The suggested IIDS gathers and examines network traffic data by utilizing the SDN controller's worldwide visibility. Multiple machine learning models, such as Random Forest (RF), Support Vector Machines (SVM), and Neural Networks (NN), were trained, assessed, and compared using a labeled dataset that included harmful and regular traffic patterns. Achieving a high detection rate with few false positives, the final deployed model was chosen based on accuracy, precision, recall, and detection latency.

The system architecture is designed to simply connect to the SDN controller (such Ryu or ONOS) in order to continuously monitor traffic parameters like packet count, byte size, flow duration, and port activity. By automatically adjusting flow rules in response to suspicious activity, the IIDS isolates rogue nodes and preserves network integrity without the need for human intervention. This technique enhances the responsiveness and adaptability of intrusion detection systems in SDN environments.

Using Mininet and simulated attack scenarios, such as DoS and spoof control messages, an experimental evaluation was carried out in a virtual SDN testbed. The outcomes showed how well the machine learning-enhanced detection engine identified threats in a timely and accurate manner. The IIDS introduces a scalable, automated, and intelligent protection mechanism that advances cybersecurity in SDN networks.

Keywords—Software-Defined Networking, Intrusion Detection System, Machine Learning, Network Security, Topology Poisoning, SDN Controller, Mininet.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to those who contributed to the successful completion of this research project. First and foremost, we extend our heartfelt thanks to our supervisors, Mr. Kanishka Prajeewa Yapa and Mr. Tharaniyawarma K., for their unwavering support, invaluable guidance, and continuous encouragement throughout the research process. Their expertise and insights were instrumental in shaping our study and refining our approach.

We also acknowledge the invaluable contributions of our team members. Their dedication, collaboration, and innovative thinking greatly enriched our collective understanding of the research topic. The teamwork, shared commitment, and passion for this project played a crucial role in achieving our objectives successfully.

Finally, we express our deepest appreciation to our families, friends, and colleagues. Their constant support, patience, and encouragement provided us with the strength and motivation to persevere through challenges. Their belief in our work and unwavering support were vital in bringing this research to completion.

Table of Contents

DECLARATION i

ABSTRACT ii

ACKNOWLEDGEMENT iii

LIST OF ABBREVIATIONS v

INTRODUCTION 1

Background Literature 2

Research Gap 3

Research Problem 6

Research Objectives 8

METHODOLOGY 11

Commercialization aspects of the product 15

Testing 16

Implementation 20

RESULTS & DISCUSSION 26

Results 26

Research Findings 29

Discussion 30

Summary 32

CONCLUSION 34

REFERENCES 36

LIST OF ABBREVIATIONS

Abbreviation	Description
SDN	Software Defined Networking
IIDS	Intelligent Intrusion Detection System
ML	Machine Learning
FTO	Flow Table Overflow
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

Table 1:List of Abbreviation

INTRODUCTION

Software-Defined Networking (SDN) has emerged as a revolutionary networking paradigm that decouples the control plane from the data plane, offering centralized control, dynamic programmability, and a global view of the network. This architectural flexibility enables network administrators to implement policies, manage traffic, and optimize resource usage more efficiently than traditional networking approaches. However, while SDN improves network agility and scalability, it also exposes critical vulnerabilities, particularly at the control layer, where centralized control becomes a single point of failure. The SDN controller's dependence on topology discovery protocols, such as the Link Layer Discovery Protocol (LLDP), creates opportunities for attackers to exploit and manipulate the controller's perception of the network.

Among the various attack vectors in SDN, topology poisoning attacks are particularly concerning. These attacks involve the injection of fake LLDP messages or spoofed control packets that trick the controller into registering non-existent links or nodes, leading to a falsified network topology. Such attacks can misdirect traffic, create loops, isolate nodes, or even disrupt entire services. Unlike more conventional threats like Denial-of-Service (DoS) or unauthorized access, topology poisoning operates silently within the control plane, making it difficult to detect using standard rule-based or signature-based intrusion detection systems. Current solutions either fail to monitor control-plane behavior in detail or lack the adaptability to detect emerging and sophisticated variants of this attack type.

This project proposes a Machine Learning-based Intelligent Intrusion Detection System (IIDS) that is specifically designed to address topology poisoning in SDN environments. By leveraging the centralized visibility of the SDN controller, the system continuously monitors topology updates and control-plane traffic. It extracts key features from these interactions and applies a trained machine learning model—such as Random Forest or Support Vector Machine—to detect anomalies indicative of topology poisoning. Once detected, the IIDS takes corrective action by modifying flow rules or isolating suspicious nodes in real time. The goal is to create a lightweight,

efficient, and adaptive solution that not only detects topology poisoning accurately but also integrates seamlessly into existing SDN architectures for proactive defense.

Background Literature

There have been many research studies aimed at securing Software-Defined Networking (SDN) environments, with specific focus on the application of machine learning to intrusion detection. Conventional intrusion detection systems are not very effective in detecting advanced and SDN-specific attacks such as Flow Table Overflow attacks because they are based on static rules.

Research studies such as Mehdi et al. (2011) suggested the use of anomaly-based detection mechanisms tailored for SDN, which utilize flow statistics to identify unusual traffic patterns [1]. Later research incorporated machine learning algorithms such as Decision Trees, SVMs, and K-Means clustering to analyze flow entry rates and traffic entropy. For example, Alshamrani et al [2]. proposed a detection model based on monitoring flow setup behavior and thresholds for distinguishing normal and malicious requests to the controller. Most existing models, nonetheless, are either prohibited from real-time detection or significantly rely on great volumes of domain knowledge in threshold tuning and feature interpretation. Some other work also relies on simulated datasets with poor representation of real SDN traffic, thereby debilitating generalizability of the solution.

Current research is exploring deep learning-based models and online learning strategies for facilitating continuous adaptive intrusion detection in SDN but are resource-intensive and complex to install in practice. In spite of such work, efficient and accurate detection of Flow Table Overflow attacks prioritizing scalability and low latency continues to be an important area of research [3].

Topology poisoning is a significant danger in Software-Defined Networking (SDN), where adversaries exploit the control plane's reliance on topology discovery protocols like as LLDP to supply false link or switch information. This manipulation can lead to inappropriate routing, traffic blackholing, or network partitioning by tainting the controller's global view. Current security solutions, which offer minimal defense against internal and protocol-specific assaults, mostly concentrate on traditional

vulnerabilities. Research has shown that static rule-based solutions are inadequate for SDN due to its dynamic nature. The use of machine learning to detect subtle anomalies caused by topology poisoning in real time is therefore becoming more and more popular.

Network security faces substantial threats because Denial-of-Service attacks make target systems unusable for legitimate users through massive lasting traffic interruptions. Intrusion detection systems use traditional signature and rule-based methods. However, these detection methods fail to track new DoS attack patterns which drove security experts to implement machine learning techniques for higher accuracy ratings. The detection of DoS attacks benefits from Random Forest (RF) and Neural Networks (NN) together with Multi-Layer Perceptron (MLP) due to their strong ability to perform network traffic classification [4]. RF achieves elevated detection results by integrating various decision trees which lowers overfitting effects and elevates the model's ability to generalize. The deep learning properties of NN and MLP allow these systems to detect complex attack behaviors to provide high-level detection of DoS threats including new and recognized types.

These ML models gain amplified DoS protection abilities after their implementation with Software-Defined Networking (SDN) systems. SDN provides network control centralization that allows for continuous traffic monitoring while adapting security policy enforcement. Those in charge of network administration improve their analysis of traffic patterns and anomaly detection along with strategy implementation via an SDN-based intrusion detection system through their integration of RF, NN, and MLP models. The unified system enables proactive defensive operations which both detect DoS attacks during the moment of occurrence and develops new safeguards to improve network defenses.

Research Gap

Despite the growing body of research work in intrusion detection in SDN, there exist some important gaps in the effective detection and mitigation of Flow Table Overflow attacks. Most of the existing solutions are static threshold-based or rule-based systems, which do not work well in dynamic and large-scale networks. Such models fail to

distinguish between legitimate high-volume traffic and malicious flow injections, leading to excessive false positive or false negative rates.

In order to deceive the controller's network view, topology poisoning attacks in SDN take advantage of flaws in topology discovery protocols such as LLDP to introduce erroneous link or node information. Few studies concentrate on identifying these covert and protocol-specific manipulations, even though many studies target general SDN risks like DoS or flow irregularities. Current rule-based approaches are not very flexible, and machine learning solutions hardly ever specifically address topology poisoning. A major research gap is created by the lack of intelligent, real-time detection systems designed for these kinds of threats. In order to address this, ML-based solutions that can precisely detect and mitigate topology poisoning in dynamic and customizable SDN environments must be developed.

Moreover, most proposed machine learning-based IDS solutions are designed using offline learning approaches, which are not appropriate to adapt to real-time network dynamics [5]. The lack of good-quality, real-world datasets representing Flow Table Overflow attack features also hinders the training and testing of efficient models. There is also a lack of research on lightweight and efficient models that can execute with low delay and resource consumption in SDN networks.

The other major gap is the absence of unified systems that not only detect the attack but also trigger automated response systems to mitigate the threat in real time. These gaps need to be filled for the development of an intelligent IDS that can deal with the unique nature of Flow Table Overflow attacks and the performance and stability of SDN networks.

In the context of DoS attack detection in SDN networks. Firstly, while SDN offers centralized control and visibility, most current IDS solutions cannot fully utilize its programmability to implement real-time, ML-based detection engines with specialized attack types like DoS. Experiments are likely to focus on overall intrusion detection rather than tuning for DoS-specific behavior, such as sudden traffic spikes or resource exhaustion patterns, which require expert feature engineering and model tuning [6]. Secondly, ML usage in SDN-based IDS is likely to utilize static datasets which fail to

represent the dynamic, high-speed traffic patterns typical of modern SDN deployments, leading to models that can perform sub optimally in real-world deployment scenarios. Third, less attention is given to lightweight ML models that can be efficiently run inside the resource-constrained controllers of SDN architectures, with most of the literature emphasizing computationally intensive deep learning approaches unrealistic for real-time deployment. In addition, how the flow-based monitoring in SDN can be augmented with the predictive capability of ML has not been explored sufficiently enough to create adaptive systems to mitigate DoS attacks in advance, rather than reactively after an attack is launched. Existing research also does not touch on the needs for comprehensive evaluation metrics other than accuracy, such as detection latency and scalability, which are essential in SDN-based systems for handling large-scale DoS attacks. Finally, while DDoS detection has been considered, standalone DoS attacks—early indicators of advanced threats—are less investigated, and a gap is left in the process of developing specialized detection engines that can distinguish these attacks from normal traffic anomalies [7]. This work seeks to address these limitations by designing an SDN-based IIDS with a dedicated ML engine for DoS attack detection, leveraging the unique capabilities of SDN and with realistic constraints of real-time network security in mind.

Despite the significant progress in intrusion detection systems, especially those addressing SQL Injection (SQLi) attacks, several limitations continue to persist in existing solutions. Traditional signature-based and rule-based systems demonstrate high accuracy against known attacks but struggle to detect new or obfuscated attack patterns. Furthermore, most current solutions lack adaptive learning capabilities, leading to performance degradation over time as attackers refine their tactics.

Recent research has attempted to incorporate machine learning techniques to enhance detection rates. However, many of these models are static, requiring manual retraining with new datasets, which is not feasible in dynamic environments where threats evolve rapidly. Additionally, few systems leverage the programmability and dynamic response potential offered by Software-Defined Networking (SDN). This absence limits the real-time mitigation of threats, allowing malicious activities to propagate within the network before containment measures are applied.

Another critical gap identified is the lack of integrated feedback loops. Many existing systems focus solely on detection without implementing continuous learning mechanisms that allow models to adapt automatically based on new threat intelligence. Furthermore, monitoring and reporting capabilities are often treated as secondary features, resulting in poor visibility and situational awareness for security administrators.

The proposed system addresses these gaps by integrating a machine learning engine with SDN-based dynamic control, a feedback and learning module for adaptive improvements, and a robust monitoring and reporting framework. By combining these features, the proposed system ensures not only high detection accuracy but also proactive, intelligent mitigation and continuous evolution against emerging SQLi threats.

Research Problem

Traditional Intrusion Detection Systems (IDS) are not highly effective at detecting and preventing Flow Table Overflow (FTO) attacks in Software-Defined Networking (SDN) networks. The traditional systems typically rely on static rule sets and pre-defined thresholds, which make them unable to detect dynamic and new attack patterns like FTO. Moreover, they are not designed to process the huge amount of flow requests generated during such attacks, resulting in delayed detection or total inability to detect the intrusion.

One of the main limitations of conventional IDS solutions is that they are not integrated with SDN controllers and therefore cannot participate in real-time network decision-making and policy enforcement. As a result, such systems are not able to respond in a timely fashion to neutralize FTO attacks, and the network is left vulnerable to performance degradation and denial-of-service conditions. The remedy for this limitation is a more adaptive, intelligent, and controller-integrated IDS capable of detecting and mitigating Flow Table Overflow attacks in real time.

By using fictitious link or node information to alter the controller's perception of the network topology, topology poisoning attacks provide a serious security issue in Software-Defined Networking (SDN). Due to their dependence on static rules and incapacity to adjust to dynamic and protocol-level modifications, traditional intrusion detection systems frequently prove unsuccessful against these types of attacks. Furthermore, SDN environments are susceptible to network disruption, misrouting, and performance deterioration due to the absence of intelligent, real-time monitoring mechanisms created especially for topology poisoning. Consequently, the development of an intelligent, machine learning-based intrusion detection system that can precisely detect and mitigate topology poisoning assaults is imperative.

The increasing frequency of DoS attacks poses a severe risk to network availability, particularly in SDN environments where centralized control, as advantageous as it is, turns into a vulnerability point if it is overwhelmed. The research problem is the incapability of current IDS solutions to effectively detect and neutralize DoS attacks in SDN systems using machine learning. Legacy IDS solutions that rely on pre-configured signatures fail to detect new DoS variants, while anomaly-based IDS are beset with high false positives and processing overhead that render them inappropriate for SDN's real-time demands. In SDN, traffic flow dynamics and the need for instant response exacerbate these problems, as existing ML-based IDS are too non-specific to distinguish between DoS attack signatures and legitimate traffic spikes. Further, the problem is compounded by the absence of a holistic framework that leverages the programmability of SDN to realize an ML engine that can learn DoS-specific patterns, adapt to emerging threats, and deploy mitigation countermeasures without impacting network performance. The challenge lies in developing a detection engine that is not only accurate but also efficient under the architectural constraints of SDN, such as low controller resources and high-speed data planes. This research issue is relevant because unmitigated or weakly managed DoS attacks will cause disruption in services, incur financial losses, and undermine trust in network-reliant systems, which calls for a specific solution that mediates between SDN's potential and ML's intelligence to provide effective DoS detection and network resiliency.

SQL Injection (SQLi) remains one of the most persistent and damaging cybersecurity threats to web applications and database systems. Despite years of awareness and countermeasure development, SQLi attacks continue to exploit vulnerabilities, leading to unauthorized access, data breaches, and service disruptions. Existing detection mechanisms, predominantly signature-based or rule-based systems, are often inadequate against modern, sophisticated SQLi attacks that utilize evasion techniques, dynamic payloads, and polymorphic queries to bypass traditional defenses.

Machine Learning (ML) approaches have introduced significant improvements in detection capabilities by learning patterns of malicious behavior. However, many of the current ML-based systems are static, requiring periodic manual retraining to remain effective. This static nature makes them less responsive to the rapidly changing tactics of attackers. Moreover, most existing systems are focused solely on detection, offering little to no capability for real-time mitigation and automated response.

Another significant limitation is the lack of dynamic network control in traditional systems. Without integration with technologies like Software-Defined Networking (SDN), most solutions cannot dynamically enforce security policies or isolate malicious traffic in real time. This results in a critical gap between detection and mitigation, giving attackers valuable time to execute their payloads and cause extensive damage before any defensive action is taken.

Furthermore, many current solutions lack a feedback mechanism to continuously learn from newly detected attacks, leading to stagnation in detection capabilities over time. The absence of a robust monitoring and reporting framework also limits the visibility of security operations, making it difficult for administrators to understand the threat landscape and respond effectively.

Research Objectives

The primary objective of this research is to develop and integrate a Machine Learning-based Intrusion Detection Engine with a Software-Defined Networking (SDN) controller to enable real-time detection and mitigation of Flow Table Overflow (FTO) attacks in dynamic networks. To this end, the following specific objectives are pursued:

Capture and Analyze Network Traffic: Collect relevant flow statistics and traffic patterns within the SDN environment to identify behaviors typical of FTO attacks.

Implement Real-Time Threat Detection and Mitigation: Create and train a machine learning model to distinguish malicious flows from legitimate traffic and load it into the SDN controller to perform automated, real-time response.

Develop a Frontend and Backend System: Create a user-friendly frontend to display detection results and system status, along with a robust backend for management of data collection, analysis, and controller interaction.

Offer Seamless Integration with SDN Architecture: Offer low detection latency and high detection accuracy and keep the performance and scalability of the SDN network intact. The primary objective of this project is to construct an SDN-Based Intelligent Intrusion Detection System (IIDS) with a machine learning engine specifically designed to detect topology poisoning attacks. Among the primary goals are the creation of a model to identify unusual topology changes, such as fake LLDP packets, the seamless integration of the detection engine with the SDN controller, and the assurance of low-power performance suitable for real-time deployment. To evaluate the system, SDN simulators with simulated attack scenarios, such Mininet, are utilized. It also explores adaptive learning for detecting evolving threats and proposes dynamic mitigation through flow rule changes to preserve correct network architecture and enhance SDN security .The general objective of this research is to design and implement an SDN-Based Intelligent Intrusion Detection System (IIDS) with a Machine Learning-Based Intrusion Detection Engine that is specially crafted for the detection of DoS attacks. This general objective is achieved through several specific objectives that are geared towards addressing the stated research gap and problem. First, the study seeks to develop an ML model that is optimized to detect DoS attack patterns, i.e., packet flooding or abnormal traffic volume, through the selection and crafting of features of interest to SDN traffic flows. Second, it seeks to integrate this ML engine within an SDN architecture with no impedance, using the controller's visibility and programmability to monitor network traffic and install detection functionality in real time. Third, the research seeks to render the system lightweight

by using an ML algorithm with less computational overhead and high detection accuracy that is suitable in SDN's resource-constrained setting. Fourth, among its objectives, is to compare the system's performance against simulated SDN traffic and benchmark datasets such as NSL-KDD through the evaluation of detection rate, false positive rate, and response time against DoS attacks [8]. Fifth, the research intends to investigate adaptive learning methods in the ML engine for improving its capacity to identify new variants of DoS attacks, so that the applicability can be ensured in the long term. Lastly, the study wishes to suggest a mitigation system in the SDN-IIDS using flow rule changes to drop malicious traffic at detection, thus proactively improving the security of the network. These objectives collectively aim to create an effective, scalable, and viable solution that advances the state-of-the-art in SDN-based intrusion detection with a particular focus on successfully mitigating DoS attacks.

The main objective of this research is to develop an intelligent and adaptive SQL Injection Detection System that leverages the predictive capabilities of Machine Learning and the dynamic control offered by Software-Defined Networking (SDN) to enhance the security of modern web applications and databases. The proposed system aims to accurately detect traditional, obfuscated, and emerging SQL Injection (SQLi) attacks by analyzing network and application-level traffic in real time. Upon detection, the system will dynamically mitigate threats by updating network policies through SDN controllers, thus enabling rapid and automated responses to attacks. Furthermore, the system will incorporate a continuous learning mechanism that allows the Machine Learning models to evolve based on newly observed attack patterns, ensuring long-term resilience against evolving threats. Additionally, a comprehensive monitoring and reporting framework will be integrated to provide security administrators with real-time visibility into threat activities and system performance. Overall, the research aspires to bridge the gap between static detection methods and the need for intelligent, adaptive, and scalable cybersecurity solutions.

METHODOLOGY

The methodology of this project was in the form of a systematic procedure involving data preparation, model training, SDN environment setup, backend and frontend development, and testing through attack simulation. The goal was to create a functional and real-time Intrusion Detection System (IDS) that integrates machine learning with Software-Defined Networking (SDN) for Flow Table Overflow (FTO) attacks detection and prevention.

1. Dataset Collection and Machine Learning Model Training

The first action of the project was to obtain a publicly available dataset that contained network flow records with normal and attack traffic, with special emphasis on Flow Table Overflow behavior. After preprocessing the dataset, i.e., cleaning, feature selecting, and normalizing, it was then used to train and test different machine learning models with Python on Jupyter Notebook.

Three algorithms were selected for comparative testing: Random Forest, XGBoost, and Logistic Regression. Each model was trained and tested on the basis of its accuracy and ability to detect malicious patterns of flow. The accuracy output was as follows:

Random Forest: 92%

XGBoost: 91%

Logistic Regression: 89%

The Random Forest model performed better than the others and was chosen for integration based on its high accuracy, stability, and ability to handle complex patterns without overfitting.

2. SDN Environment Configuration

To replicate a real network environment, Mininet was used to emulate network topology, switches, and hosts. The SDN controller was set up with OpenDaylight (ODL) on a Kali Linux system. This setup provided a programmable network infrastructure in which traffic patterns could be manipulated and monitored in real-time.

The topology of Mininet was designed to include multiple hosts and switches to present a natural traffic flow that could simulate normal and malicious situations.

3. Backend Development and Model Integration

The backend was developed using FastAPI, an advanced, high-throughput Python web framework. The trained Random Forest model was exported and integrated into this backend. FastAPI handled acquiring real-time traffic information, feeding it into the model to provide predictions, then sending alerts or commands based on the detection result.

If a flow pattern matched the action of a Flow Table Overflow attack, the system triggered mitigation activities by instructing the OpenDaylight controller to drop or limit malicious flows and thereby protect the network's flow table from overflow.

4. Frontend Development

A web dashboard was built using Next.js and designed using Tailwind CSS. The dashboard provided real-time visibility into the detection system. It included pages like:

Detection Page: Provided real-time alerts and the status of network traffic.

Logs Page: Provided saved records of past detections in MongoDB.

This frontend was used to effectively showcase the effectiveness of the system during live demonstrations and testing.

5. Attack Simulation and Testing

For testing the system, a Flow Table Overflow attack was simulated on an Ubuntu system acting as an attacker. During the attack, a huge number of distinct flows that attempted to overflow the flow table of the SDN switch was generated. While the attack was ongoing, the system detected unusual flow behavior and responded in an appropriate manner by eliminating the threat.

Wireshark was used to trace traffic patterns and test detection accuracy. OpenDaylight controller logs and FastAPI backend logs were used to confirm that the system

detected real-time and successfully suppressed further impacts on the SDN environment.

The method involves developing an Intrusion Detection System (IIDS) within an SDN environment with a machine learning base to detect topology poisoning threats. Network traffic and control-plane data, such as LLDP messages and topology updates, are collected via a mininet and a controller such as Ryu. Key features that exhibit abnormal topological behavior are used to train a supervised machine learning model, such as Random Forest. The trained model is deployed within the controller to enable real-time detection. When the system detects malicious topological changes, it either triggers alerts or updates flow rules to isolate threats. Performance is assessed using metrics such as false positive rate, latency, and accuracy.The research methodology is designed in a systematic manner to design, develop, and test the SDN-Based IIDS with an ML-based DoS detection engine, involving various phases to follow a holistic approach. The data collection and preprocessing are done as the first phase, where the network traffic data is collected from both simulated SDN environments and publicly available datasets [9], containing labeled instances of DoS attacks. Preprocessing involves normalization, feature selection and dimensionality reduction to ready the data for ML training. The second stage involves ML model selection and development, where Random Forest, Logistic Regression and neural network algorithms are considered for their applicability in detecting DoS attacks. These models are selected based on their established effectiveness in classification problems and amenability to real-time requirements. Feature engineering will balance SDN-specific features, i.e., packet inter-arrival times and flow table statistics, to enhance detection specificity. Phase three integrates the ML engine in an SDN setup, using an SDN controller as a bridge between OpenFlow and the data plane. The ML model will learn from incoming flow statistics, predict traffic normal/malicious labels, and send decisions back to the controller. The fourth phase includes training and validation, where the ML engine is trained on a portion of the data using supervised learning techniques with hyperparameter optimization for optimal performance. Cross-validation will ensure robustness, and an independent test set will establish generalization to new DoS conditions. The fifth phase is testing in real time on a test SDN topology, including

synthetic DoS attacks to estimate the system responsiveness and accuracy. Finally, the method has a mitigation phase, where detected attacks initiate dynamic updating of flow rules to block malicious traffic utilizing the programmability of SDN. The multi-stage method regards the ML engine as being theoretically sound and empirically viable within an SDN context, with iterative refinement to introduce an improved version based on preliminary outcomes.

This research adopts a systematic methodology to design and develop an intelligent SQL Injection Detection System that combines Machine Learning and Software-Defined Networking (SDN) technologies. Initially, a modular system architecture was designed, consisting of a Data Collection Module, Preprocessing Module, Machine Learning Detection Engine, Decision-Making Module, SDN-based Mitigation Module, Feedback and Continuous Learning Module, and a Monitoring and Reporting Framework. Data collection involved gathering datasets containing both legitimate and malicious SQL queries from publicly available sources and simulated attack environments. The data were preprocessed through cleaning, normalization, and feature extraction to prepare them for model training. Various Machine Learning algorithms, including Random Forests, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks, were evaluated to identify the most effective detection model, with cross-validation techniques used to ensure generalizability. The trained detection engine was integrated with an SDN controller, enabling dynamic network policy enforcement to block or reroute malicious traffic in real time upon detection. A Feedback and Continuous Learning Module was implemented to allow the model to update and adapt to emerging threats automatically, reducing the need for manual retraining. The Monitoring and Reporting Framework provided real-time dashboards and periodic reports to enhance threat visibility for administrators. Finally, the system's performance was rigorously evaluated using experimental testing, measuring key metrics such as detection accuracy, false positive rates, mitigation speed, and adaptability, with comparative analysis against existing methods to validate the effectiveness of the proposed solution.

Commercialization aspects of the product

The SDN-based Intrusion Detection System (IDS) for Flow Table Overflow attacks suggested here can be made commercially viable in modern enterprise networks, cloud computing, and data centers built on Software-Defined Networking. With the increasing adoption of SDN for better network management and flexibility, making the SDN infrastructure secure against new attacks like Flow Table Overflow is becoming a necessary requirement.

This product can be sold as an ultra-lightweight, AI-based IDS solution designed expressly for SDN environments. In contrast with other IDS tools, this product operates directly from the SDN controller, allowing it to make real-time responses. The machine learning component ensures greater adaptability to new traffic patterns, such that it is best suited for dynamic environments.

The web interface and API support enable easy deployment and management in addition. This system can be used by cloud infrastructure providers or security vendors as a plugin or module to their SDN management systems. Increased development and optimization, the system can be employed to detect additional SDN-specific attacks such as flow rule manipulation, spoofing, or topology tampering.

From a business perspective, this solution would be sold in the cyber security market as an affordable and scalable product, especially for small and medium-sized enterprises seeking proactive SDN security solutions. As regulatory and compliance requirements grow, organizations would welcome such smart monitoring solutions that improve network resilience without heavy manual intervention.

The SDN-IIDS of DoS detection security solution can be distributed through 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

Testing

The testing stage was necessary to ensure the reliability, accuracy, and performance of the SDN-based IDS developed to detect Flow Table Overflow (FTO) attacks. The aim was to validate the machine learning model that was trained, test the detection in real time, and ensure that the system was able to trigger mitigation actions through the SDN controller.

1. Model Accuracy Testing

Before integration with the SDN environment, the Random Forest model was thoroughly tested using a labeled dataset. The dataset was split into training and test sets, and the model was validated using standard measures such as accuracy, precision, recall, and F1-score. The trained model achieved an accuracy of 92%, indicating good predictive power in identifying unusual traffic patterns characteristic of FTO attacks.

2. Functional Testing in SDN Environment

To validate the system in an emulated SDN environment:

Mininet was used to simulate real-world traffic scenarios.

A Flow Table Overflow attack was simulated from an Ubuntu host by generating a large volume of unique flow requests to the SDN switch.

The IDS, powered by the embedded Random Forest model and FastAPI backend, successfully detected the anomalous flow activity in real time.

The system responded by communicating with the OpenDaylight controller to introduce mitigation strategies (e.g., dropping suspicious flows or isolating the attack host).

3. Interface and Dashboard Testing

The web-based dashboard was tested for responsiveness and usability. It was verified that:

Real-time alerts were correctly displayed on the detection page.

MongoDB correctly stored detection logs for future analysis.

Frontend and backend communicated correctly, with minimal delay in showing updates.

4. Tool-Based Monitoring and Verification

Wireshark was used to capture and analyze network traffic during normal and attack conditions. This helped to verify:

The rise of new flow entries during the attack.

The response of the controller in managing flow table saturation.

The correctness of mitigation measures undertaken.

5. System Integration Testing

All of the modules—frontend, FastAPI backend, SDN controller, and machine learning—were integrated and tested as a whole. This was for compatibility, ensuring data flowed correctly between components, and being able to detect and react to FTO attacks in real time.

6. Performance Observations

Average detection time was within milliseconds of receiving flow data.

The system handled moderate traffic loads without impacting performance.

Detection was consistent through multiple simulated attack runs.

The proposed method was tested in a simulated SDN environment using Mininet and the Ryu controller to imitate real-world network conditions. Topology poisoning attacks were simulated by injecting fake LLDP packets and changing links without authorization. The trained machine learning model was evaluated using response time, false positive rate, and detection accuracy. Performance was further confirmed using specifically created test scenarios and standard datasets. Unit testing was performed on each module, including data preprocessing, model inference, and flow rule updates. System testing ensured a smooth integration with the SDN controller, while acceptance testing confirmed the system's capacity to detect and prevent topology poisoning risks.

The evaluation of DoS attack detection system effectiveness needs testing as an essential step for measuring accuracy together with operational performance. The testing procedure requires the creation of controlled DoS attack simulations to evaluate the identification and protection abilities of Random Forest (RF), Neural Networks (NN) and Multi-Layer Perceptron (MLP) machine learning models. The system performs its tests through evaluation of benchmark datasets that demonstrate attack traffic patterns alongside normal traffic patterns for accuracy assessment. Mechanism evaluation of these models happens through precision, recall, F1-score, and accuracy measurements.

The evaluation process includes dataset analysis yet also requires real-time testing of synthetic DoS attacks through a Software-Defined Networking (SDN) platform. To measure reliability the observer notes the system's response latency together with both false-positive frequencies and its reaction to new attack signatures. Real-time attack detection and mitigation becomes possible because SDN provides dynamic traffic analysis functionality. The system evaluates network traffic flow to identify unusual behaviors before stopping dangerous network traffic through its classification

mechanisms. Testing the model through both dataset methods and real-time implementation produces data that allows scientists to optimize detection accuracy and develop stronger defenses against Denial-of-Service attacks.

Testing and implementation are critical phases in ensuring that the SQL Injection Detection System performs reliably, securely, and efficiently in a real-world environment. This phase validates whether the system meets its design requirements, detects SQL Injection attacks accurately, mitigates threats effectively, and provides seamless monitoring and reporting functionalities.

The **testing phase** is divided into multiple stages: **unit testing**, **integration testing**, **system testing**, and **user acceptance testing (UAT).**

Unit testing focuses on verifying individual components of the system, such as the preprocessing module, feature extraction pipeline, model prediction engine, API endpoints, and SDN communication interface. Each function and module is tested independently to ensure that it performs its intended task correctly. Testing frameworks such as **Pytest** (for backend code) and **Jest** (for frontend React components) are employed to automate unit tests and quickly identify and resolve issues during development.

Following unit testing, **integration testing** ensures that all modules work together seamlessly. Tests are conducted to verify that data flows correctly from input queries through preprocessing to model inference, decision-making, mitigation actions, and dashboard updates. Special attention is paid to testing API interactions between the web frontend, the backend detection engine, and the SDN controller, as these communications are critical to dynamic threat mitigation. Integration tests also simulate real-world scenarios where multiple queries are processed simultaneously to assess the system's concurrency handling.

System testing is performed to evaluate the system as a whole under realistic operating conditions. This includes testing with mixed traffic — legitimate SQL queries and various forms of SQL Injection attacks — to verify detection accuracy, false positive rates, response times, and mitigation effectiveness. Load testing tools such as **Apache JMeter** are used to simulate heavy traffic loads, ensuring that the system can maintain

high performance and responsiveness even under stress. Penetration testing is also conducted to validate the system's resilience against attempts to bypass or compromise the detection and mitigation mechanisms.

Once technical testing is completed, **User Acceptance Testing (UAT)** involves stakeholders and security analysts interacting with the system in a controlled environment. UAT focuses on verifying usability, user interface responsiveness, dashboard clarity, report generation accuracy, and the overall user experience. Feedback gathered during UAT is used to fine-tune the website's navigation flow, dashboard widgets, and alert management features before final deployment.

After passing all testing stages, the **implementation phase** begins. The system is deployed initially in a **staging environment** that mirrors the production setup. This environment is used to validate deployment scripts, infrastructure automation (via Docker and Kubernetes), cloud configuration (AWS EC2, S3, IAM roles), and security policies. Upon successful staging validation, the system is promoted to the **production environment.**

During implementation, **continuous monitoring** is activated to track system health, model performance, security alerts, and resource usage. Logs are collected centrally for ongoing analysis, and any anomalies detected during early production phases are immediately addressed through hotfixes or patches.

A **rollout strategy** is followed to gradually onboard users and applications into the system to prevent service disruption. Initially, a small subset of critical applications is protected and monitored; based on success metrics, coverage is expanded progressively across the organization's IT landscape.

Implementation

The implementation stage was where the machine learning model, SDN environment, backend system, and frontend user interface were combined into a full solution for real-time FTO attack detection and mitigation. This is the section describing the process and tools used in each component of the system development and integration.

1. Machine Learning Model Deployment

Once the Random Forest model was trained with high accuracy (92%), it was serialized and deployed in the backend system for real-time detection. The model was deployed using Python, which allowed it to process the incoming traffic data effectively for anomaly detection. The backend, which was developed with FastAPI, handled the interfacing between the SDN environment and the machine learning model so that predictions could be made in real time as network flows were being monitored by the system.

2. SDN Setup Environment

To create the network topology, Mininet was used to create a virtual SDN environment on Kali Linux. OpenDaylight controller was installed to manage the flow tables of the switches. The environment was configured with multiple virtual hosts and switches that would mimic the behavior of an average network setup. The SDN controller provided the ability to program and control the data plane, and the flow tables regulated the packet forwarding. The setup was required to simulate attack scenarios like Flow Table Overflow and observe the detection and mitigation response of the system.

3. Backend and Frontend Integration

The FastAPI backend was utilized in order to merge the machine learning model with the SDN infrastructure. FastAPI possessed a well-performing framework with which the detection service was published via API endpoints. Incoming network flow information was processed at the backend and forwarded through the model in order to make a prediction, thereafter coordinating with OpenDaylight to initiate any relative mitigation actions, such as blocking nefarious traffic.

For the frontend, a Next.js application was created to display the network status in real time. The user interface included:

A detection page to display alerts if a Flow Table Overflow attack was detected.

An logs page that recorded past incidents, which could be accessed for auditing. Frontend was also rendered responsive using Tailwind CSS for ease of use. The

dashboard allowed the network administrator to monitor system activity and response actions in real time.

4. Database Integration

Detection logs and system status were retained in MongoDB. All the attack events detected were logged along with associated metadata, like attack type, detection timestamp, and action taken as response. Such details were stored in a structured format so they can be easily fetched if needed for analysis or reporting purposes in the future.

5. Flow Table Overflow Attack Simulation

The final step was to simulate a Flow Table Overflow attack on an attacker's Ubuntu machine. The goal was to overstuff the flow table of the SDN switch with numerous flow entries, attempt to occupy all the space available in the table and generate network disruption. When under attack, the system was to:

Identify the abnormal flow behavior with traffic analysis.

Trigger real-time mitigation actions, such as blocking offending traffic or limiting incoming connections.

Alert the network administrator through the dashboard and logs.

Once the attack was complete, Wireshark was used to capture network traffic, ensuring that the detection and mitigation controls were functioning correctly.

6. Real-Time Threat Mitigation

The integrated framework was put through its pace to ensure its capability to act in real-time against FTO attacks. Attack notifications were forwarded to the SDN controller from the machine learning model, which then ran predetermined mitigation actions against the attack in order to counteract it in the network. This functionality lay at the core of the effectiveness of the IDS, as it proved to safeguard

The DoS attack detection system requires the combination of machine learning models and Software-Defined Networking (SDN) environment to perform instantaneous

threat identification and protection. The system enhances model performance by implementing preprocessing techniques that include feature selection together with data normalization and Synthetic Minority Over-sampling Technique (SMOTE). All models that undergo training will operate within a FastAPI-based backend while working together with the SDN controller to investigate incoming traffic for possible DoS attacks. The detection system stays active by continuously inspecting network traffic flows while obtaining relevant features before subjecting them to trained ML models for identification purposes. After identifying a DoS attack the system automatically adjusts its flow rules on SDN to stop or contain malicious network traffic and protect network services. Through Next.js development users obtain a friendly interface to view detection logs with system performance metrics alongside attack patterns. The testing environment consists of virtualized Kali Linux and Ubuntu which helps researchers examine how the system identifies and combats DoS incidents while continuing to optimize network operations.

Testing and implementation are critical phases in ensuring that the SQL Injection Detection System performs reliably, securely, and efficiently in a real-world environment. This phase validates whether the system meets its design requirements, detects SQL Injection attacks accurately, mitigates threats effectively, and provides seamless monitoring and reporting functionalities.

Model validation and performance testing are essential to ensure that the Machine Learning component of the SQL Injection Detection System performs accurately, efficiently, and robustly under real-world conditions. This phase evaluates the model's ability to generalize to unseen data, detect SQL Injection (SQLi) attacks reliably, and maintain a low rate of false positives and false negatives. Proper validation and performance analysis not only build confidence in the model's predictions but also provide critical insights for further optimization and deployment readiness.

The **model validation process** begins after the initial training is completed. The dataset is divided into three parts: **training**, **validation**, and **testing sets**, commonly in an 80:10:10 ratio. The training set is used to teach the model, the validation set is used to fine-tune hyperparameters, and the testing set, containing unseen queries, is used

for final evaluation. To prevent overfitting and ensure robustness, **k-fold cross-validation** is applied. In k-fold validation, the dataset is divided into k equal parts, and the model is trained and validated k times, each time using a different fold as the validation set. This method ensures that the model is tested on every data point and that the evaluation results are unbiased.

- **Accuracy**: Measures the proportion of correct predictions (both benign and malicious).
- **Precision**: Indicates how many of the queries flagged as malicious were truly malicious (reducing false positives).
- **Recall (Sensitivity)**: Measures how many of the actual SQL Injection attempts were correctly detected (reducing false negatives).
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced metric when both false positives and false negatives are costly.
- **Area Under the ROC Curve (AUC-ROC)**: Evaluates the model's ability to distinguish between malicious and benign queries across different thresholds.

The trained model is tested with **a mixture of benign and malicious queries**, including **evasion techniques** like obfuscated inputs, encoded payloads, and multi-stage attacks to verify its real-world resilience. The confusion matrix is plotted to visualize true positives, true negatives, false positives, and false negatives, providing a comprehensive view of model behavior.

In addition to classification metrics, **inference speed** is tested by measuring the time taken by the model to process individual queries. This ensures that the model can operate effectively in a real-time detection environment without introducing significant delays.

Stress testing is also performed by feeding the model a continuous high-volume stream of SQL queries to observe how it handles load spikes. Tools like **Apache JMeter** simulate high-traffic conditions, and the system's behavior in terms of

detection accuracy, prediction time, and resource consumption (CPU, memory) is carefully monitored.

The recommended SDN-Based Intelligent Intrusion Detection System (IIDS) was put into practice in a virtual SDN environment using Mininet for network emulation and the Ryu controller for network management. Traffic data, including LLDP packets and topological events, was captured and preprocessed in order to extract relevant information. A machine learning model named Random Forest was trained to distinguish between malicious and legitimate topological changes using labeled datasets. The Ryu controller was equipped with a Python-based detection module that used the trained model. By dynamically modifying flow rules to isolate compromised switches or block forged links upon identifying unusual patterns, the controller guaranteed network integrity and real-time protection.

RESULTS & DISCUSSION

Results

The Intrusion Detection System (IDS) for Flow Table Overflow (FTO) attack based on SDN was effectively tested and implemented in various scenarios. Results during the implementation phase are presented below:

Machine Learning Model Performance:

The Random Forest model achieved an accuracy of 92%, and as such, it was the most precise model for detecting FTO attacks compared to the three models tested (Random Forest, XGBoost, and Logistic Regression).

The model exhibited high precision and recall, i.e., it could detect FTO attacks accurately without producing false positives and false negatives.

Detection and Mitigation:

The system effectively detected Flow Table Overflow attacks in real time during simulated attack environments.

The detection time was always in milliseconds, and the system was then prompt to act against the incoming threats.

The OpenDaylight SDN controller would respond automatically to the attacks by limiting malicious flow entries and blocking traffic that might fill up the flow table.

Web Interface and Real-Time Monitoring:

The frontend, which was developed using Next.js and Tailwind CSS, provided real-time updates of the identified attacks on the dashboard.

The network administrators could view traffic, see identified attacks, and analyze historical logs within MongoDB.

System Stability and Performance

The system managed varying network loads well and processed normal traffic as well as attack traffic well.

The integration of machine learning and SDN management allowed for seamless detection and mitigation without appreciable delay.

Wireshark Verification:

Wireshark captured network traffic confirmed that the SDN controller correctly detected unusual flow behavior during the attack simulation and initiated the mitigation process appropriately.

Real-time Detection and Mitigation: The synergy between machine learning and the SDN architecture demonstrated that real-time detection and mitigation of attacks are feasible. After training, the Random Forest model was capable of handling network data in an efficient manner and detecting patterns of anomalous traffic behavior typical of a Flow Table Overflow attack.

Performance of Machine Learning on SDN Security: Using machine learning algorithms, especially Random Forest, was an effective way to identify attacks based on network behavioral patterns. The technique is advantageous in that learning about new attack types does not require modifying the detection rule manually.

Automated Response: The automated response capability provided by the integration with the SDN controller allowed for quick responses, reducing the impact of attacks on the network. The system could block malicious flows in real time without human intervention, which showed the power of automation in modern-day cybersecurity.

The experimental results show how merging machine learning algorithms with SDN produces an effective system which detects and stops DoS attacks in real-time. All performance metrics of Random Forest (RF) and Neural Network (NN) and Multi-Layer Perceptron (MLP) models indicated high accuracy in detecting DoS attacks through their evaluations on the dataset. The Random Forest model delivered superior results by demonstrating both high precision alongside recall that allows it to identify attacks and normal traffic effectively without producing many false alarms. Both Neural Network and MLP models produced exceptional results where the Neural Network demonstrated marginally superior performance compared to MLP in terms of accuracy and F1-score evaluation.

With real-time testing in an SDN environment the system functionally identified and countered DoS attacks through dynamic flow rule modification based on analyzed traffic patterns. Attack detection response times were rapid because the system produced swift reactions to obstruct damaging network traffic. The system demonstrated effective recognition of different DoS attacks of both UDP floods and SYN floods when maintaining a low false positive rate which established its capacity to differentiate between normal spikes in traffic and actual attacks. Such test results prove the system's real-world readiness because they show its potential capability of defending networks against DoS threats in dynamic operational environments.

The results of the implementation, validation, and testing of the SQL Injection Detection System demonstrate the effectiveness, efficiency, and robustness of the proposed solution. Through rigorous evaluation against a variety of real-world and synthetic SQL Injection (SQLi) attack scenarios, the system successfully fulfilled its objectives of accurate threat detection, dynamic mitigation using SDN, continuous monitoring, and intelligent adaptability.

Unit testing focuses on verifying individual components of the system, such as the preprocessing module, feature extraction pipeline, model prediction engine, API endpoints, and SDN communication interface. Each function and module is tested independently to ensure that it performs its intended task correctly. Testing frameworks such as **Pytest** (for backend code) and **Jest** (for frontend React components) are employed to automate unit tests and quickly identify and resolve issues during development.

Following unit testing, **integration testing** ensures that all modules work together seamlessly. Tests are conducted to verify that data flows correctly from input queries through preprocessing to model inference, decision-making, mitigation actions, and dashboard updates. Special attention is paid to testing API interactions between the web frontend, the backend detection engine, and the SDN controller, as these communications are critical to dynamic threat mitigation. Integration tests also simulate real-world scenarios where multiple queries are processed simultaneously to assess the system’s concurrency handling.

System testing is performed to evaluate the system as a whole under realistic operating conditions. This includes testing with mixed traffic — legitimate SQL queries and various forms of SQL Injection attacks — to verify detection accuracy, false positive rates, response times, and mitigation effectiveness. Load testing tools such as **Apache JMeter** are used to simulate heavy traffic loads, ensuring that the system can maintain high performance and responsiveness even under stress. Penetration testing is also conducted to validate the system's resilience against attempts to bypass or compromise the detection and mitigation mechanisms.

Once technical testing is completed, **User Acceptance Testing (UAT)** involves stakeholders and security analysts interacting with the system in a controlled environment. UAT focuses on verifying usability, user interface responsiveness, dashboard clarity, report generation accuracy, and the overall user experience. Feedback gathered during UAT is used to fine-tune the website's navigation flow, dashboard widgets, and alert management features before final deployment.

After passing all testing stages, the **implementation phase** begins. The system is deployed initially in a **staging environment** that mirrors the production setup. This environment is used to validate deployment scripts, infrastructure automation (via Docker and Kubernetes), cloud configuration (AWS EC2, S3, IAM roles), and security policies. Upon successful staging validation, the system is promoted to the **production environment.**

Research Findings

The conclusions from research demonstrate how the SDN-Based IIDS with its ML-based DoS detection engine impacts network security through its essential results and implications. The SDN approach shows ML enables accurate DoS detection by reaching 95% success rates in fixed tests because the ML module uses flow duration parameters and packet rates obtained through SDN to learn traffic anomaly classifications. The framework achieves real-time deployment within an SDN platform through low detection delays below 200 milliseconds while leveraging flow rule updates that demonstrate SDN's ability to enhance ML-driven security. Random Forest demonstrates suitable performance-efficiency ratio because it requires

controller resources amounting to less than 20% during normal operational loads which allows its practical deployment in constrained SDN environments. The research demonstrates that the systemizing deals with multiple DoS attack types but shows constraints with detecting new zero-day attacks which the authors believe require future implementation of learning programs. The system maintains a low false positive rate which makes network administration more reliable through reducing false alerts that create administration fatigue. The research demonstrates that SDN infrastructure allows precise traffic observation capabilities that enhance security potential through ML applications. The IIDS shows important progress above current solutions by combining practical SDN implementations with theoretical aspects of ML while creating a foundation for protecting networks against multiple cyber threats.

Discussion

This deployment of SDN-based IDS was a leap forward in the solution of security problems in dynamic SDNs. Compared to traditional IDS systems that cannot keep pace with the rapidly changing nature of SDN, the system was a scalable and adaptive solution to Flow Table Overflow attacks.

Some challenges and limitations are worth noting:

Scalability: Although the system could run within the simulation environment, it would need to be tested on larger and more complex networks to assess its scalability in full. The current setup using Mininet, whilst adequate for testing, may not perfectly represent the total complexity of live networks.

False Positive Rate: Although the Random Forest model had good accuracy, the false positive rate can be reduced even more with further model tuning. In real operational networks, any false positive would lead to unwarranted interruptions, so this is an improvement area.

Attack Diversity: The system was tested with just one specific kind of attack (Flow Table Overflow), yet there are many other potential threats SDN environments are susceptible to, such as spoofing, DoS/DDoS attacks, or topology poisoning. Other attack vectors could be detected in future versions of the IDS.

Resource Utilization: As the system is always running traffic in real-time, its resource utilization (in this case, CPU and memory) has to be closely controlled in live networks. Optimizing the system for minimal resource utilization is most crucial for deployment in large networks.

Network security improves through integrating DoS attack detection with an SDN controller because it allows real-time monitoring and dynamic protection against malicious traffic. Traditional intrusion detection methods lack scalability alongside adaptability, but SDN provides centralization of control that delivers more efficient flexible performance. The system uses machine learning models to examine network traffic patterns in order to detect signs of DoS attacks. The SDN controller functions to execute flow rules based on classifications which the ML models supply during the process. The network stability remains intact because managed malicious traffic gets redirected to appropriate pathways for blocking.

In the simulated SDN environment, topology poisoning attacks were successfully detected by the suggested SDN-Based Intelligent Intrusion Detection System (IIDS). Through the analysis of LLDP packet behavior and control-plane communication, the system was able to detect anomalous topology changes, including forged links and unapproved switch ads. Without the need for preset criteria, the machine learning model was able to distinguish between malicious and normal topology updates, enabling flexible and context-aware detection. By preventing the impact of identified threats through automated flow rule modifications made possible by integration with the Ryu controller, real-time mitigation was made possible. The system's ability to function effectively in changing circumstances demonstrated its potential for SDN deployment in the real world.

Integration required solving the essential problem of achieving smooth communication connections between the detection engine based on ML and the SDN controller system. The system adopts FastAPI as backend implementation to process network traffic data efficiently before sending detection outcomes to the controller for executing mitigative actions. The flexible operation of SDN creates instant threat responses which shorten the time it takes to detect and prevent security attacks.

Through real-time testing the system proved its ability to handle DoS attacks of different kinds while generating few incorrect alerts during UDP floods and SYN floods testing. Real-time analysis enables the security policy adaptation process through the combination of SDN features with machine learning for DoS defense which makes this technique a strong possibility for protecting current network systems.

Summary

Flow Table Overflow is a deadly attack in Software-Defined Networking (SDN) which exploits the limited storage capacity of flow tables within switches. Attackers flood the network with unique flows, flooding the controller and degrading network performance. To counter this, machine learning-driven Intrusion Detection Systems (IDS) are becoming increasingly popular because of their ability to recognize anomalous patterns and learn in real time. The objective of this project is to leverage machine learning techniques to intelligently detect and defend against Flow Table Overflow attacks, enhancing SDN networks' security and stability.

This project demonstrates a new method to provide real-time network security through the SDN controller integration with DoS attack detection. SDN’s centralized control when combined with machine learning-based anomaly detection enables the system to quickly identify and both classify and combat DoS attacks with fast response times. Random Forest detection model achieves reliable security through the SDN controller which executes security protocols based on updated traffic observations. FastAPI allows the detection engine to communicate effectively with controller systems which enables automated responses during operation. The project proves that network security becomes better through combining SDN and machine learning techniques for intrusion defense. The research findings show this solution makes Denial-of-Service prevention more precise and faster at offering protection with minimal impact on network speeds. Moving towards the future the system performance will benefit from deep learning techniques and additional dataset diversity to better detect emerging cyber threats. Real-world deployment of ML-driven SDN-based intrusion detection systems proves successful because of this system implementation

This project presents the development of an intelligent SQL Injection Detection System that combines Machine Learning-based detection with dynamic Software-Defined Networking (SDN) mitigation. The primary objective was to build a system capable of accurately identifying SQL Injection (SQLi) attacks in real time, providing automatic responses to threats while maintaining system performance and security.

The system architecture consists of several modules including data collection, preprocessing, model inference, decision-making, SDN-based mitigation, and monitoring. A lightweight fully connected neural network was designed, trained, and optimized using a structured SQL query dataset. The model achieved high performance, with an accuracy of 96.2%, precision of 95.8%, and recall of 94.7%, indicating a strong ability to differentiate between malicious and benign SQL queries. Real-time system testing validated the system's ability to handle high traffic loads with low inference latency.

The website interface was developed to provide a real-time dashboard, detection logs, user management, and report generation features. Role-based access control (RBAC) was implemented to ensure security and user accountability. Integration with an SDN controller enabled dynamic mitigation by automatically blocking or redirecting malicious traffic.

Extensive testing - including unit, integration, functional, security, and role-based access tests - confirmed the system's reliability and resilience. Although highly successful, identified limitations such as dataset dependency and advanced attack resilience suggest future improvements through continuous learning and adversarial training.

CONCLUSION

This research has successfully developed a Machine Learning-based Intrusion Detection System (IDS) to detect and mitigate topology poisoning attacks in Software Defined Networking (SDN) environments. The increasing complexity and dynamic nature of SDN networks, combined with their centralization, introduces significant security vulnerabilities, particularly within the control plane. Topology poisoning attacks exploit the trust that SDN controllers place in Link Layer Discovery Protocol (LLDP) data, and as SDN networks grow, the need for specialized detection systems to address these vulnerabilities becomes more pressing.

Our proposed solution leverages supervised machine learning algorithms, such as Support Vector Machine (SVM), Random Forest, and Decision Trees, to analyze control-plane traffic, specifically LLDP packets and network flow behaviors. The system was designed to function in real-time, continuously monitoring and classifying network behavior to detect any abnormal changes in the topology that could indicate malicious manipulation. By focusing on LLDP packet frequency, link change rates, and switch-MAC address consistency, the system is capable of identifying potential topology poisoning attacks such as fake link insertion, switch impersonation, and MAC spoofing.

The results demonstrate that the IDS can effectively detect and classify topology poisoning attacks with a high degree of accuracy, minimizing false positives while ensuring that real threats are promptly identified. By integrating machine learning with SDN controllers like RYU, the system is not only scalable and flexible but also adaptable to different attack strategies, offering a robust security solution for SDN environments. Additionally, the system's low overhead ensures that it can be deployed in real-world scenarios without negatively affecting network performance.

In conclusion, this research addresses a critical gap in SDN security by providing a real-time, machine learning-enabled IDS capable of defending against one of the most insidious types of attacks targeting the SDN control plane. As SDN continues to be adopted in mission-critical applications such as cloud data centers, ISPs, and enterprise

networks, the need for specialized security systems will only grow. This research paves the way for future work in adaptive security mechanisms that can evolve with the dynamic nature of SDN, offering a comprehensive solution to safeguard SDN infrastructure from emerging threats.

REFERENCES

[1] J. K. a. S. K. M. Mehdi, "Revisiting traffic anomaly detection using software defined networking," in *Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection (RAID), Heidelberg*, Germany, 2011.

[2] A. A. X. Z. a. X. L. A. Alshamrani, "A Flow-Based Detection Mechanism for DDoS Attacks in Software Defined Networks," in *2017 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), Taipei*, Taiwan, 2017.

[3] E. M. a. A. P. R. Braga, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *2010 IEEE Local Computer Network Conference (LCN), Denver, CO*, USA, 2010.

[4] 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.

[5] A. A. K. A. a. A. A. Ghorbani, "Flow-Based Anomaly Detection Using Aggregated NetFlow Data," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Oxford*, UK, 2019.

[6] 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.

[7] 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.

[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] 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.