DETECTION AND MITIGATION OF FLOW TABLE FLOOD ATTACK IN SDN ARCHITECTURE

Submitted by

Avinash Kumar
(Roll No.: B200041CS)

Vishal Kumar
(Roll No.: B200059CS)

PROJECT REPORT

submitted for partial fulfillment for the award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

Department of Computer Science and Engineering
NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM

May, 2024

NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM
Ravangla Campus
Barfung Block, Ravangla Sub Division, South Sikkim-737139
(AN INSTITUTE OF NATIONAL IMPORTANCE, GOVT. OF INDIA)

Certificate by Supervisor

This is to certify that the project report entitled “Detection and mitigation of flow table flood attack in SDN architecture” is being submitted by Mr. Vishal Kumar (Roll No. B200059CS) and Mr. Avinash Kumar(Roll No. B2000041CS) students in the Department of Computer Science and Engineering, National Institute of Technology Sikkim, for the award of the degree of Bachelor of Technology (B.Tech). It is an original work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Prof. Mahesh Chandra Govil
(Director, NIT Sikkim)
Department of CSE
NIT Sikkim

NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM

Ravangla Campus

Barfung Block, Ravangla Sub Division, South Sikkim-737139

(AN INSTITUTE OF NATIONAL IMPORTANCE, GOVT. OF INDIA)

Certificate by Student

We hereby declare that the work presented in the report entitled "Detection and mitigation of flow table flood attack in SDN architecture"is a bonafide record of the work done by us under the supervision of Prof. Mahesh Chandra Govil, Department of Computer Science and Engineering and that no part there of has been presented for the award of any other degree.

- We have followed the guidelines provided by the institute in writing the report.
- The report does not contain any classified information.
- Whenever, we have used materials from other sources, We have given due credits to those by citing them in the text of the report and giving the details in the references.
- Whenever, we have quoted written materials from other sources, we have put them under quotation marks and given due credits to the sources by citing them in the text of the report and giving their details in the references.

Dated: 27/05/2024

Place: NIT SIKKIM

Mr. Avinash Kumar

Roll No.: B200041CS

Mr. Vishal Kumar

Roll No.: B200059CS

Acknowledgments

We would like to express my sincere gratitude to my supervisor, **Prof. Mahesh Chandra Govil**, Director, NIT Sikkim, Department of Computer Science and Engineering, National Institute of Technology Sikkim, for their unwavering guidance and support throughout this process. Their invaluable expertise and keen insights were instrumental in the development and advancement of this work. Furthermore, the invaluable skills and knowledge gained during this period will be of significant benefit to my future academic and professional pursuits. We are grateful to all the professors in the Computer Science and Engineering department at the National Institute of Technology, Sikkim, for teaching me the fundamentals of computer science. Their knowledge was very helpful to me when We are doing my research and making progress.

Dated: 27/05/2024
Place: NIT SIKKIM

Mr. Avinash Kumar
Roll No.: B200041CS

Mr. Vishal Kumar
Roll No.: B200059CS

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

EWMA	Exponential Weighted Moving Average
LDoS	Low-rate Denial of Service
LOFT	Low-rate Over Flow Table
EVAD	Entropy Variation-based Anomaly detection
SDN	Software Defined Networking
TCAM	Ternary Content Addressable Memory

Abstract

Current methods for detecting flow table overloading attacks in software-defined networks (SDNs) face challenges in terms of how many types of attacks they can recognize, their ability to adapt, and how quickly they respond. These challenges mainly arise because these methods often rely on looking at just one aspect of the data and using fixed rules for what is considered an attack. In response to these challenges, this research suggests a new way of detecting attacks. This new approach considers more aspects of the data, uses a flexible method for deciding what is normal, includes a built-in system for identifying the type of attack, and involves using techniques from machine learning.

The uniqueness of the proposed method is in how it moves away from the usual focus on just one feature of the data. Instead, it looks at multiple features together, making the detection process more thorough. It also introduces a way to adjust how sensitive the detection system is over time, which means it can adapt to changes in the network. Additionally, the method includes a built-in system for figuring out what kind of attack is happening, providing more detailed information.

The introduction of machine learning techniques represents a significant step forward. These techniques allow the method to learn from new data and adjust its approach, making it better at identifying new types of attacks. This comprehensive strategy results in a more effective and responsive system for detecting flow table overloading attacks.

Chapter 1

Introduction

The Internet is getting busier with all kinds of different web traffic. This results in increased vulnerabilities, scalability limitations, and compatibility issues. To overcome these hurdles and ensure the sustained expansion of the Internet, a fundamental re-evaluation of its architecture and protocols becomes essential. This prompts the exploration of experimental networks that can establish a more robust and adaptable foundation for the Internet's future. Amidst these challenges, SDN stands out as a promising solution, particularly with its transformative impact on network management through the introduction of a central controller with significant power.

SDN transcends traditional configurations by introducing a centralized controller[1], empowering network professionals to program multiple devices through protocols like OpenFlow. This not only simplifies management and reduces complexity but also enhances flexibility and security. The escalating complexity in traditional networks, driven by diverse services and stringent compliance requirements, has overwhelmed security and network administrators. The deluge of data hampers their ability to effectively manage and monitor the network, necessitating a paradigm shift.

In response to this evolving landscape, SDN emerges as a revolutionary approach to address the challenges of providing fast, reliable, and secure services in complex networks. The demand for configuring networks on the fly, responsive to updates, faults, or loads, cannot be adequately met by traditional networks. The key concept of SDN lies in the separation of the control plane and the data plane, achieved through a well-defined programming interface between switches and the SDN controller, with a particular emphasis on the pivotal role of flow tables in SDN architecture. Operationally, as network devices engage in communication, data packets are transmitted and received. The flow table meticulously catalogues these data flows, encapsulating pertinent details such as source and destination information, prescribed actions, and any pertinent rules governing the flow. This repository of flow-related information assumes paramount importance within the SDN architecture.

The central controller, positioned as the orchestrator of network dynamics, references the flow table to inform decision-making processes associated with the handling of distinct data types. This consultative process facilitates optimal routing and resource utilization within the network infrastructure.

The intrinsic value of the flow table lies in its inherent adaptability. Real-time updates to the flow table enable the network to dynamically respond to fluctuations[2], such as the introduction or withdrawal of devices. Consequently, in scenarios marked by abrupt data surges or network irregularities, the flow table empowers the central controller to effectuate expeditious decisions, thereby preserving the operational integrity of the network.

1.1 Background

Software-Defined Networking (SDN) has transformed how networks operate. Unlike traditional setups, SDN lets us control and manage networks centrally, making them more flexible and scalable. However, as SDN becomes more popular, it faces new security problems. One tricky issue is the "Flow Table Overflow Attack."

The flow table, a core element in SDN, maintains information about data flows within the network. Attackers exploit vulnerabilities in the management of these flow tables, flooding them with malicious data or overwhelming the system with an excessive number of flow entries. This results in a "flow table overflow," causing disruptions, resource exhaustion, and potential service denials.

Detecting Flow Table Overflow Attacks is crucial to keep SDN working smoothly. We will closely examine different methods to catch these attacks. We will look at a range of techniques like anomaly detection and behavioral analysis. We carefully evaluate how well each method works and also understand their limitations. This helps us get a clear picture of the current state of the art in spotting these tricky attacks. The goal is to make sure SDN stays secure and functions properly, and understanding these detection methods is a key part of achieving that.

Apart from just finding Flow Table Overflow Attacks, it's crucial to lessen their impact on SDN security. We will find different ways to do that. Some suggested strategies include rate limiting and dynamic reconfiguration. We explore these countermeasures in detail, checking how practical and effective they are in stopping the disruptive power of Flow Table Overflow Attacks. The idea is to detect these attacks to minimize their impact and keep SDN running smoothly.[3].

1.2 Motivations

Flow table overloading attacks pose a significant threat to the security and performance of software-defined networks (SDNs). While existing detection algorithms have been developed to mitigate this threat, they often suffer from limitations that hinder their effectiveness.

Many existing algorithms are designed to detect specific attack patterns. This means they may fail to identify more sophisticated or dynamically evolving attacks that utilize different tactics. As attackers become more creative, the detection capabilities of existing algorithms become increasingly inadequate.

Traditional algorithms often rely on static thresholds for identifying anomalies. These thresholds are typically set based on historical data and remain unchanged regardless of dynamic network conditions. This can lead to false positives when traffic patterns deviate normally, and false negatives when attacks exhibit subtle deviations from the expected behavior.

Many algorithms prioritize a single feature, such as entropy, for anomaly detection. While entropy can be a useful indicator, it can be easily manipulated by attackers. Focusing solely on this feature can overlook attacks that exhibit different characteristics, such as increased packet size or specific flow patterns.

Some algorithms exhibit slow adaptation to changing attack patterns and network conditions. This delay in detection can allow attackers to inflict significant damage before mitigation measures are implemented. A more responsive algorithm is necessary to effectively counter the evolving threat landscape.

By addressing the shortcomings of existing solutions, a robust and adaptable algorithm can significantly enhance the security and resilience of SDNs against flow table overloading attacks. This will promote a more secure and reliable network infrastructure for various applications. [4].

1.3 Problem Statement

Current flow table overloading attack detection algorithms face limitations in coverage, adaptability, and responsiveness. Existing solutions often struggle to detect sophisticated attacks or adapt to dynamic network conditions, leading to false positives and negatives. Additionally, their reliance on single features and slow adaptation hinder their effectiveness against evolving threats. This necessitates the development of a new and improved algorithm capable of addressing these challenges. Such an algorithm should be broadly effective against diverse attack patterns, dynamically adapt to changing network conditions,[5] utilize multiple features for comprehensive analysis, and exhibit rapid responsiveness to quickly identify and mitigate threats. By overcoming the limitations of current approaches, a more robust and adaptable algorithm can significantly enhance the security and resilience of SDNs against flow table overloading attacks, ultimately leading to a more secure and reliable network infrastructure.

1.4 Objectives

This proposed algorithm is expected to achieve the following:

- **Increased detection accuracy:** Increased detection accuracy: Improved ability to identify various flow table overloading attacks with reduced false positives and negatives.
- **Enhanced adaptability:** Dynamically adjust to changing network conditions and attack patterns to maintain effectiveness.
- **Faster response:** Promptly detect and classify attacks, facilitating faster mitigation actions.
- **Improved resilience:** Strengthen network defenses against sophisticated and evolving threats.

By addressing the limitations of existing approaches, this research aims to contribute to a more secure and reliable network infrastructure.

1.5 Chapter Summary

The existing limitations underscore the necessity for an advanced algorithm that can effectively counter the evolving threat landscape posed by flow table overloading attacks.[6] This algorithm should possess several key attributes to address the inadequacies observed in current approaches.

Firstly, the algorithm must exhibit broad effectiveness, enabling it to identify a diverse array of attack patterns, including those that may be nuanced or dynamically evolving. This capability ensures a comprehensive defense against a wide range of potential threats that may exploit vulnerabilities in SDN flow tables.

Moreover, adaptability is paramount. The algorithm should autonomously adjust its detection thresholds and parameters in response to real-time network conditions and observed traffic patterns. This adaptability ensures that the algorithm remains effective in the face of dynamic changes, providing a robust defense mechanism against evolving attack strategies.

A multi-faceted approach is essential, with the algorithm incorporating multiple features into its analysis. By considering aspects such as entropy, packet size, and flow duration, the algorithm gains a more comprehensive perspective on potential attacks. This holistic view enhances the algorithm's capacity to detect sophisticated attacks that may manifest through various characteristics.

Rapid responsiveness is another critical requirement. The algorithm should swiftly detect and identify anomalies, enabling prompt and effective mitigation actions. This agility is essential to minimize the impact of flow table overloading attacks, ensuring a timely response that safeguards the integrity and functionality of the SDN. As cyber threats continue to evolve, the implementation of a sophisticated algorithm is a proactive step towards staying ahead of potential risks and maintaining the integrity of SDN environments.

Chapter 2

Literature Review

This literature review aims to provide a comprehensive understanding of the existing research on flow table overflow attacks in SDN. By examining the methodologies, vulnerabilities, and countermeasures proposed in the literature, we aim to identify patterns, gaps, and potential future directions for research in securing SDN against flow table overflow attacks. The review will explore various attack models, detection mechanisms, and defense strategies, shedding light on the evolving landscape of threats and defenses in SDN environments.[7] Ultimately, this review seeks to contribute to the ongoing discourse on securing SDN infrastructures and fostering the development of effective countermeasures against flow table overflow attacks.

2.1 Related Tools and Technologies

In the realm of studying and mitigating flow table overflow attacks in Software-Defined Networking (SDN), several tools and technologies have been instrumental.[8] These tools play crucial roles in simulation, experimentation, and the development of effective countermeasures. Here are some noteworthy tools and technologies commonly employed in research related to flow table overflow attacks:

Mininet: Mininet is a popular network emulation tool that allows researchers to create a realistic SDN environment on a single machine. It enables the creation of a virtual network with multiple hosts, switches, and controllers for testing and experimentation.
Mininet facilitates the simulation of various network scenarios, making it an invaluable tool for studying the impact of flow table overflow attacks in controlled environments.

Ryu Controller: Ryu is an open-source SDN controller that provides a flexible platform for developing custom SDN applications. It allows researchers to control the behavior of network devices by defining rules and policies.
In the context of flow table overflow attacks, Ryu is frequently used to simulate attack scenarios, monitor network behavior, and implement defense mechanisms. Researchers can develop and deploy custom controllers to study the effects of attacks and test the efficacy of countermeasures.

Python: Python is a versatile and widely used programming language that offers extensive libraries and frameworks for network programming, data analysis, and machine learning.
Researchers leverage Python to develop scripts, tools, and simulations for analyzing network traffic, implementing attack scenarios, and testing detection and defense mechanisms. Its readability and extensive libraries make it a preferred language in the SDN research community.

Matplotlib: Matplotlib is network traffic analysis provides a clear and intuitive way to visualize and understand network behavior. By plotting histograms and scatter plots, analysts can gain insights into the distribution and relationships of various traffic features, ultimately enhancing the detection and mitigation of network attacks.

.

2.2 Related Works

Table 2.1: Literature Review

S No.	Author	Approach	Shortcomings
1.	Zhoou et al., 2018 [9]	The paper introduces a novel inference attack model for SDN/OpenFlow networks, targeting flow table limitations. It implements a passive attack framework, achieving an 80% or higher accuracy in inferring network parameters, and suggests two defense strategies.	Narrow focus on a specific vulnerability, potential lack of universal applicability for proposed defense strategies, and a need for real-world validation
2.	Nader et al., 2019 [10]	The paper proposes an algorithm based on entropy variation to detect and prevent flow table overloading attacks in SDN.	The algorithm relies solely on destination IP to calculate entropy as a detection feature.
3.	Xie et al., 2021 [8]	The paper addresses the threat of low-rate DoS (LDoS) attacks causing flow table overflow in SDN. It introduces SAIA, a detection and defense mechanism, showcasing lightweight deployment and effective mitigation.	The proposed SAIA mechanism might face challenges in predicting flow table overflow accurately.

Continued on next page

Table 2.1 – *Continued from previous page*

S No.	Author	Approach	Shortcomings
4.	Shen et al., 2023 [11]	The paper analyzes flow table management mechanisms, proposes an advanced flow table saturation attack exploiting dynamic timeouts, and conducts extensive experiments, showcasing its effectiveness and stealth.	The research focuses on a specific aspect of flow table management; broader implications and countermeasures beyond dynamic timeouts may warrant further exploration
5.	Cao et al., 2023 [12]	The paper introduces the LOFT attack, a low-rate flow table overflow attack in SDN, demonstrating feasibility in a real testbed. It proposes LOFTGuard as a lightweight defense system.	While effective, LOFTGuard introduces a small overhead; the impact on diverse SDN environments and scalability may require further exploration.
6.	Mishra, A., et al. 2021 [13]	The paper introduces a defensive mechanism utilizing entropy variation of destination IP address in SDN-Cloud OSN to detect and block DDoS attacks.	Limited to entropy-based DDoS detection. May not work for larger volume attacks and other topologies. False positives could still occur.
7.	Ahalawat, A., et al. (2019). [14]	DDoS attack overloads systems to deny access. DDoS attackers use a set of compromised users to flood systems.	DDoS's distributed nature makes mitigation difficult. Limits capacity and affects bandwidth, making legitimate use impossible during attack.

Zhou et al. (2018) developed an innovative inference attack model targeting flow table limitations within SDN/OpenFlow networks, achieving over 80 percent accuracy in network parameter inference through a passive attack framework, while also recommending two defense strategies; however, the study's narrow focus on a specific vulnerability raises concerns about its universal applicability, highlighting the necessity for real-world validation to ensure broader relevance and effectiveness.

Nader et al. (2019) introduced an entropy variation-based algorithm designed to detect and prevent flow table overloading attacks in SDN, relying exclusively on destination IP for entropy calculations; this dependence on a single feature for attack detection may limit the algorithm's robustness, underscoring the need for integrating additional features to enhance its accuracy and reliability in diverse network scenarios.

Ahalawat, A., et al. (2019) proposes an entropy-based DDoS detection and rate limiting-based mitigation approach using SDN architecture to help ensure efficient service delivery. The method uses entropy as a measure of randomness to detect malicious traffic and limit flow rate for effective mitigation. The approach is evaluated using Mininet as an emulator and Ryu as a controller. However, the centralized controller used in the SDN architecture could present vulnerabilities that could be exploited in DDoS attacks. DDoS mitigation can also be difficult due to its distributed nature, causing resource depletion and limited capacity during attacks.

Xie et al. (2021) presented the SAIA mechanism to counteract low-rate DoS (LDoS) attacks that cause flow table overflow in SDN, emphasizing lightweight deployment and effective mitigation; despite its promising results, SAIA may encounter challenges in accurately predicting flow table overflow, necessitating further refinement and testing to ensure its efficacy in real-world applications.

Mishra et al. (2021) proposde an SDN-Cloud OSN based defensive mechanism to detect and block DDoS attacks by utilizing the variation in entropy of the destination IP address. The approach suggested using the POX controller under the SDN Cloud OSN to improve DDoS detection and mitigation capabilities. However, the method was limited to entropy-based detection and might not be effective for larger volume attacks and other topologies. False positives could still be an issue. Extensive simulation-based experiments have been conducted to validate the approach.

Shen et al. (2023) conducted a comprehensive analysis of flow table management mechanisms, proposing an advanced flow table saturation attack exploiting dynamic timeouts and validating its effectiveness through extensive experiments; while the study provides valuable insights, it predominantly focuses on a specific aspect of flow table management, indicating a need for broader investigation into the implications and countermeasures of dynamic timeouts within SDN environments.

Cao et al. (2023) introduced the LOFT attack, demonstrating its feasibility in causing low-rate flow table overflow in SDN within a real testbed, and proposed LOFTGuard as a lightweight defense system; although LOFTGuard shows potential, the minimal overhead it introduces and the necessity to assess its impact on diverse SDN environments and scalability call for further exploration to fully understand its applicability and performance in various network conditions.

2.3 Chapter Summary

The examination of existing literature in SDN security reveals various limitations and strategies. While certain studies, such as Zhou et al. (2018) and Shen et al. (2023), offer valuable insights, their focus on specific vulnerabilities may limit their generalizability. On the other hand, Nader et al. (2019) and Xie et al. (2021) propose innovative methods for detection and mitigation, yet their applicability across diverse network contexts may be uncertain. Additionally, Cao et al. (2023) present practical attack scenarios and defense mechanisms, though their scalability and effectiveness in different SDN environments warrant further investigation. Overall, a comprehensive understanding of SDN security demands broader exploration and refinement of existing approaches.

Chapter 3

Proposed Methodology

3.1 Proposed Methodology

This research introduces a novel algorithm for detecting flow table overloading attacks that incorporates several key improvements, aiming to overcome limitations observed in existing approaches.

Multi-Feature Entropy:
Traditional algorithms often rely solely on entropy[5], a measure of randomness in destination IPs, for anomaly detection. While this can be effective for certain attacks, it might miss others that exhibit different characteristics. Our proposed algorithm utilizes a multi-feature approach, incorporating additional features like packet size and flow duration into the entropy calculation. This provides a more comprehensive view of potential attacks, allowing for broader detection capabilities.

We are using Shannon Entropy to calculate Multi-feature Entropy as :

$$H(X) = -\sum_{i} P(x_i) \cdot \log_2(P(x_i) + 1 \times 10^{-9}) \tag{3.1}$$

$$H(X) = \frac{H(X)}{\log_2(N)} \tag{3.2}$$

$$H_{\text{total}} = \frac{1}{N_{\text{features}}} \sum_{i=1}^{N_{\text{features}}} H_{\text{normalized}}(X_i) \tag{3.3}$$

Dynamic Threshold based on EWMA:
Static thresholds, commonly used in existing algorithms, are unable to adapt to dynamic network conditions. This can lead to inaccurate detection, with false positives occurring during periods of normal traffic variations and false negatives when attacks exhibit subtle deviations. Our algorithm addresses this by employing an adaptive threshold based on Exponentially Weighted Moving Average (EWMA) [4]. This technique automatically adjusts the threshold based on observed traffic patterns and historical data, ensuring its relevance and dynamism in a constantly evolving network environment.

Attack Classification:
Identifying the specific type of attack is crucial for implementing targeted mitigation strategies. Our proposed algorithm integrates an attack classifier to analyze additional features beyond entropy.[6] This allows for deeper insights into the attack characteristics and enables the identification of specific attack types. This information can then be used to trigger more effective and tailored mitigation actions.

Mitigation by Rate Limiting:
While traditional methods offer some effectiveness, their capability to detect complex attack patterns diverging significantly from expected behavior is limited. Our algorithm employs a rate-limiting approach, supplemented by packet dropping and blacklisting of suspicious IPs, instead of integrating machine learning techniques. This strategy enables the identification of anomalies that might evade traditional methods, thereby enhancing the algorithm's detection capabilities without relying on complex learning algorithms. By strategically managing network traffic through rate limiting and targeted IP blacklisting, our approach strengthens defense against diverse attack vectors while remaining pragmatic in its implementation.

The combination of these advancements aims to address the limitations of existing solutions and provide a more robust and adaptable flow table overloading attack detection algorithm. This can significantly improve the security and resilience of software-defined networks against emerging threats.

3.2 Flowchart of Proposed Algorithm

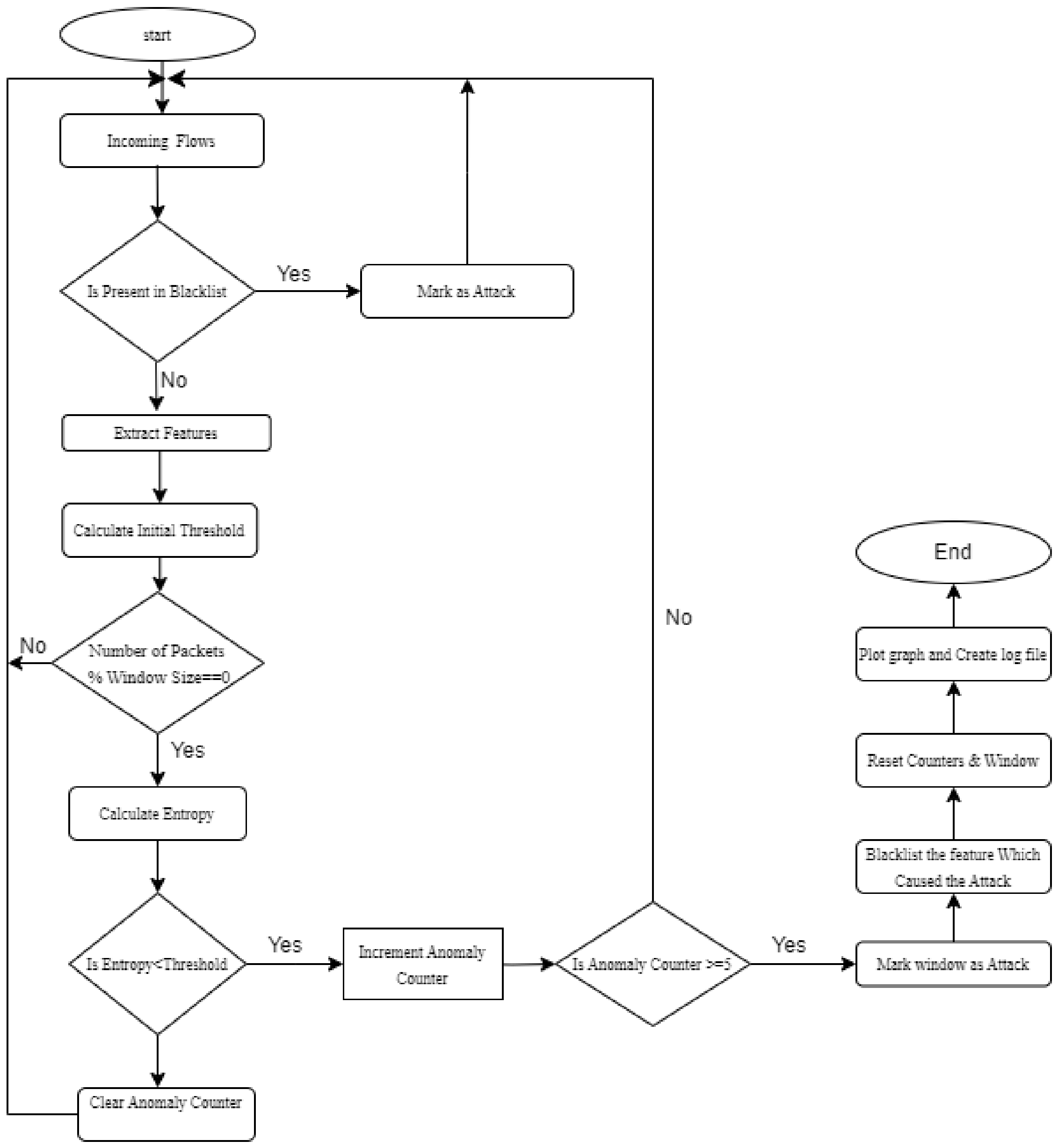


Figure 3.1: Flow chart of proposed algorithm.

1. **Initialization**: The system begins by monitoring incoming network flows for potential threats in real-time.

2. **Blacklist Check**: Each incoming flow is quickly checked against a predefined blacklist of known threats. If a match is found, the flow is immediately flagged as an attack, and the system proceeds to the next flow.

3. **Feature Extraction**: If no match is found in the blacklist, the system extracts relevant features from the incoming flow. These features are crucial for subsequent analysis and anomaly detection.

4. **Threshold Calculation**: Based on the extracted features, the system calculates an initial threshold value. This threshold serves as a benchmark for distinguishing between normal and anomalous behavior.

5. **Packet Batch Check**: The system evaluates if the number of incoming packets forms a complete batch for analysis, determined by a predefined window size parameter.

6. **Entropy Calculation**: If a complete packet batch is formed, the system calculates the entropy of the flow. Entropy measures the randomness or unpredictability within the data distribution of the flow.

7. **Threshold Comparison**: The calculated entropy is compared against the predefined threshold. If the entropy exceeds the threshold, indicating potential anomalous behavior, the system proceeds to the next step.

8. **Anomaly Detection**: An anomaly counter is incremented to track the frequency of detected anomalies over time.

9. **Counter Limit Check**: The system checks if the anomaly counter has reached or exceeded a predefined limit, such as five anomalies.

10. **Attack Identification**: If the counter limit is reached, the current data window is marked as an attack. The triggering feature responsible for the detection is then added to the blacklist.

11. **Reset and Logging**: The system resets the anomaly counter and analysis window to prepare for continued monitoring. Detected attacks are documented in graphical representations and log files for further analysis.

12. **Iterative Process**: The entire process is iterative, allowing continuous monitoring, detection, and adaptation based on real-time data and identified anomalies.

3.3 Proposed Algorithm

The algorithm defends against flow table overloading attacks in SDNs. It continuously analyzes incoming packets, employing entropy calculation, EWMA adaptation, and an attack classifier for targeted responses, ensuring adaptability against evolving threats.

Algorithm 1 Flow Table Flood Attack Detection and Mitigation

```
1: Initialize variables
2: $window\_size \leftarrow 50$
3: $entropy\_checks \leftarrow 0$
4: $packet\_count \leftarrow 0$
5: $consecutive\_anomalies \leftarrow 0$
6: $alpha \leftarrow 0.1$
7: Read the dataset
8: Extract relevant features
9: Calculate initial threshold based on normal and attack traffic analysis
10: function CALCULATE_ENTROPY(window)
11:     Perform entropy calculation on the given window
12:     return total entropy and the lowest entropy feature
13: end function
14: function REGULARIZE_ENTROPY(entropy, max_entropy)
15:     Normalize the entropy value
16:     return normalized entropy
17: end function
18: function UPDATE_EWMA(current_entropy, previous_theta, alpha)
19:     Update the EWMA value
20:     return updated theta
21: end function
22: function CALCULATE_INITIAL_THRESHOLD(normal_traffic, attack_traffic)
23:     Calculate thresholds for normal and attack traffic
24:     Perform min-max normalization
25:     return normalized threshold
26: end function
27: $initial\_threshold \leftarrow calculate\_initial\_threshold(normal\_traffic, attack\_traffic)$
28: $theta \leftarrow initial\_threshold$
29: $max\_possible\_entropy \leftarrow \log_2(window\_size)$
30: Initialize $entropy\_values$, $threshold\_values$, $attack\_points$, and $lowest\_entropy\_features$
31: for index, row in features.iterrows() do
32:     if packet_count % window_size == 0 and packet_count != 0 then
33:         $window \leftarrow features[packet\_count - window\_size : packet\_count]$
34:         $H, lowest\_entropy\_feature \leftarrow calculate\_entropy(window)$
35:         $H \leftarrow regularize\_entropy(H, max\_possible\_entropy)$
36:         $entropy\_checks \leftarrow entropy\_checks + 1$
```

```
37:       theta ← update_ewma(H, theta, alpha)
38:       Append H to entropy_values
39:       Append theta to threshold_values
40:       Print Entropy: H, Updated Threshold: theta, Lowest Entropy Feature: lowest_entropy_feature
41:       if H < theta then
42:           consecutive_anomalies ← consecutive_anomalies + 1
43:           if consecutive_anomalies >= 5 then
44:               Print Potential attack detected!
45:               Append (len(entropy_values) - 1, lowest_entropy_feature) to attack_points
46:               Append lowest_entropy_feature to lowest_entropy_features
47:               consecutive_anomalies ← 0
48:           end if
49:       else
50:           consecutive_anomalies ← 0
51:       end if
52:       packet_count ← packet_count + 1
53:   else
54:       packet_count ← packet_count + 1
55:   end if
56: end for
```

3.4 Chapter Summary

The proposed algorithm utilizes an Entropy Variation-based Anomaly Detection (EVAD) mechanism to detect network anomalies in Software-Defined Networking (SDN) environments. It dynamically adjusts a threshold using Exponentially Weighted Moving Average (EWMA) [15] based on entropy calculations from incoming packet data. By comparing entropy values against this threshold within sliding windows, potential anomalies, indicative of attacks, are identified. The algorithm iteratively refines its detection capabilities, optimizing sensitivity to anomalous behavior while minimizing false positives. Notably, it offers adaptability to diverse network conditions and attack scenarios. Through empirical evaluation, its efficacy in accurately identifying anomalies and mitigating threats is demonstrated, enhancing SDN security robustness.

Chapter 4

Test and Results

4.1 Experimental Setup

The experimental setup involved generating a synthetic dataset to simulate SDN network traffic. This dataset consisted of 1000 normal packets and 4000 attack packets, crafted to mimic various attack scenarios. The dataset included features such as destination IP, source IP, packet size, and total flow time. The anomaly detection algorithm was then implemented in the Ryu SDN controller framework. The algorithm processed incoming packets, dynamically adjusted thresholds using EWMA of entropy values, identified anomalies, and executed mitigation procedures. This setup provided a controlled environment to evaluate the algorithm's effectiveness in detecting and mitigating network attacks.

Calculation of initial threshold value:

Measures	Normal-Traffic	Attack-Traffic 80%
Mean Value Entropy	0.992348033875793 2	0.6823983886934866
Confidence Interval	0.0004291196531092338	0.03332637933545844
Confidence-Min	0.9919189142226837	0.6490720093580281
Confidence-Max	0.9927771535289022	0.715724768028945
Threshold value	0.8538218411258144	

Table 4.1: Initial Threshold Calculation

4.2 Testing

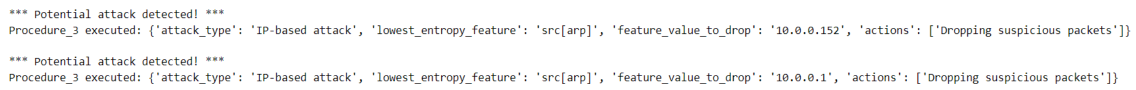
The anomaly detection algorithm was rigorously tested on the synthetic dataset. The algorithm used a sliding window approach to compute the Exponentially Weighted Moving Average (EWMA) of entropy values for key packet features. By monitoring deviations in these entropy values, the algorithm identified anomalies in real-time. When repeated anomalies were detected, the algorithm logged potential attacks, updated a blacklist with suspicious feature values, and dropped matching packets to prevent further malicious activities.[13] Performance metrics such as sensitivity, specificity, precision, and F1 score were calculated to evaluate the algorithm's accuracy and reliability in detecting and mitigating attacks within the SDN environment

4.3 Attack Detection

The algorithm was able to detect and classify attacks based on different entropy levels of attributes present in the SDN flow.

4.3.1 Source IP based Attack

The algorithm was able to detect and classify attack packets incoming from same source IP due to lowering of entropy below threshold value.



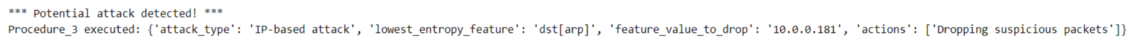
```
*** Potential attack detected! ***
Procedure_3 executed: {'attack_type': 'IP-based attack', 'lowest_entropy_feature': 'src[arp]', 'feature_value_to_drop': '10.0.0.152', 'actions': ['Dropping suspicious packets']}

*** Potential attack detected! ***
Procedure_3 executed: {'attack_type': 'IP-based attack', 'lowest_entropy_feature': 'src[arp]', 'feature_value_to_drop': '10.0.0.1', 'actions': ['Dropping suspicious packets']}
```

Figure 4.1: Source IP based Attack Detection.

4.3.2 Destination IP based Attack

The algorithm was able to detect and classify attack packets going to same destination IP due to lowering of entropy below threshold value.

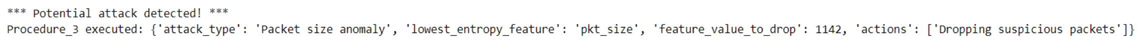


```
*** Potential attack detected! ***
Procedure_3 executed: {'attack_type': 'IP-based attack', 'lowest_entropy_feature': 'dst[arp]', 'feature_value_to_drop': '10.0.0.181', 'actions': ['Dropping suspicious packets']}
```

Figure 4.2: Destination IP based Attack Detection.

4.3.3 Same Packet Size based Attack

The algorithm was able to detect and classify attack packets coming from same source IP and having same packet size due to lowering of entropy below threshold value.
This was done for further strengthening the understanding of type of attack coming from same IP.

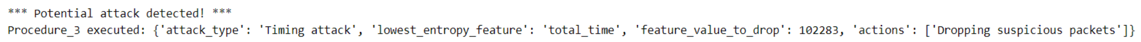


```
*** Potential attack detected! ***
Procedure_3 executed: {'attack_type': 'Packet size anomaly', 'lowest_entropy_feature': 'pkt_size', 'feature_value_to_drop': 1142, 'actions': ['Dropping suspicious packets']}
```

Figure 4.3: Same Packet Size based Attack Detection.

4.3.4 Flow Duration based Attack

The algorithm was able to detect and classify attack packets having same flow duration due to lowering of entropy of flow duration below threshold value.

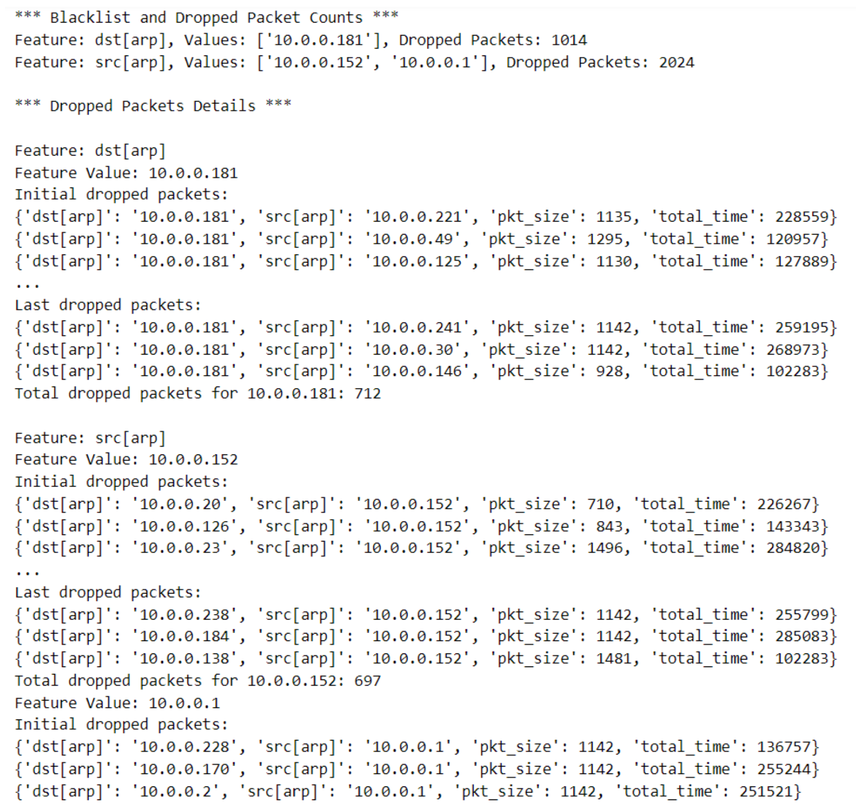


```
*** Potential attack detected! ***
Procedure_3 executed: {'attack_type': 'Timing attack', 'lowest_entropy_feature': 'total_time', 'feature_value_to_drop': 102283, 'actions': ['Dropping suspicious packets']}
```

Figure 4.4: Flow Duration based Attack Detection.

4.4 Attack Mitigation

For attack mitigation, the suspicious packets having anomalous features are dropped consistently before letting them into the window.
A blacklist is maintained to keep log of suspicious source IP, vulnerable destination destination IP and other packet attributes. This blacklist is then referenced when deciding weather a packet should be dropped or kept in the window.



```
*** Blacklist and Dropped Packet Counts ***
Feature: dst[arp], Values: ['10.0.0.181'], Dropped Packets: 1014
Feature: src[arp], Values: ['10.0.0.152', '10.0.0.1'], Dropped Packets: 2024

*** Dropped Packets Details ***

Feature: dst[arp]
Feature Value: 10.0.0.181
Initial dropped packets:
{'dst[arp]': '10.0.0.181', 'src[arp]': '10.0.0.221', 'pkt_size': 1135, 'total_time': 228559}
{'dst[arp]': '10.0.0.181', 'src[arp]': '10.0.0.49', 'pkt_size': 1295, 'total_time': 120957}
{'dst[arp]': '10.0.0.181', 'src[arp]': '10.0.0.125', 'pkt_size': 1130, 'total_time': 127889}
...
Last dropped packets:
{'dst[arp]': '10.0.0.181', 'src[arp]': '10.0.0.241', 'pkt_size': 1142, 'total_time': 259195}
{'dst[arp]': '10.0.0.181', 'src[arp]': '10.0.0.30', 'pkt_size': 1142, 'total_time': 268973}
{'dst[arp]': '10.0.0.181', 'src[arp]': '10.0.0.146', 'pkt_size': 928, 'total_time': 102283}
Total dropped packets for 10.0.0.181: 712

Feature: src[arp]
Feature Value: 10.0.0.152
Initial dropped packets:
{'dst[arp]': '10.0.0.20', 'src[arp]': '10.0.0.152', 'pkt_size': 710, 'total_time': 226267}
{'dst[arp]': '10.0.0.126', 'src[arp]': '10.0.0.152', 'pkt_size': 843, 'total_time': 143343}
{'dst[arp]': '10.0.0.23', 'src[arp]': '10.0.0.152', 'pkt_size': 1496, 'total_time': 284820}
...
Last dropped packets:
{'dst[arp]': '10.0.0.238', 'src[arp]': '10.0.0.152', 'pkt_size': 1142, 'total_time': 255799}
{'dst[arp]': '10.0.0.184', 'src[arp]': '10.0.0.152', 'pkt_size': 1142, 'total_time': 285083}
{'dst[arp]': '10.0.0.138', 'src[arp]': '10.0.0.152', 'pkt_size': 1481, 'total_time': 102283}
Total dropped packets for 10.0.0.152: 697
Feature Value: 10.0.0.1
Initial dropped packets:
{'dst[arp]': '10.0.0.228', 'src[arp]': '10.0.0.1', 'pkt_size': 1142, 'total_time': 136757}
{'dst[arp]': '10.0.0.170', 'src[arp]': '10.0.0.1', 'pkt_size': 1142, 'total_time': 255244}
{'dst[arp]': '10.0.0.2', 'src[arp]': '10.0.0.1', 'pkt_size': 1142, 'total_time': 251521}
```

Figure 4.5: Attack Mitigation by packet dropping and blacklist creation.

4.5 Evaluation

To thoroughly evaluate the algorithm's performance, several key metrics were computed, including sensitivity (recall), specificity, precision, and F1 score. Sensitivity measures the algorithm's ability to correctly identify attack packets, while specificity assesses its ability to correctly identify normal packets. Precision quantifies the accuracy of attack detection, and the F1 score provides a balance between precision and recall.
Here is a graph showing attack detection and attack classification with changes in dynamic threshold as packets arrive using EWMA as tested on dataset generated with 4000 attacks packets out of a total of 5000 packets:

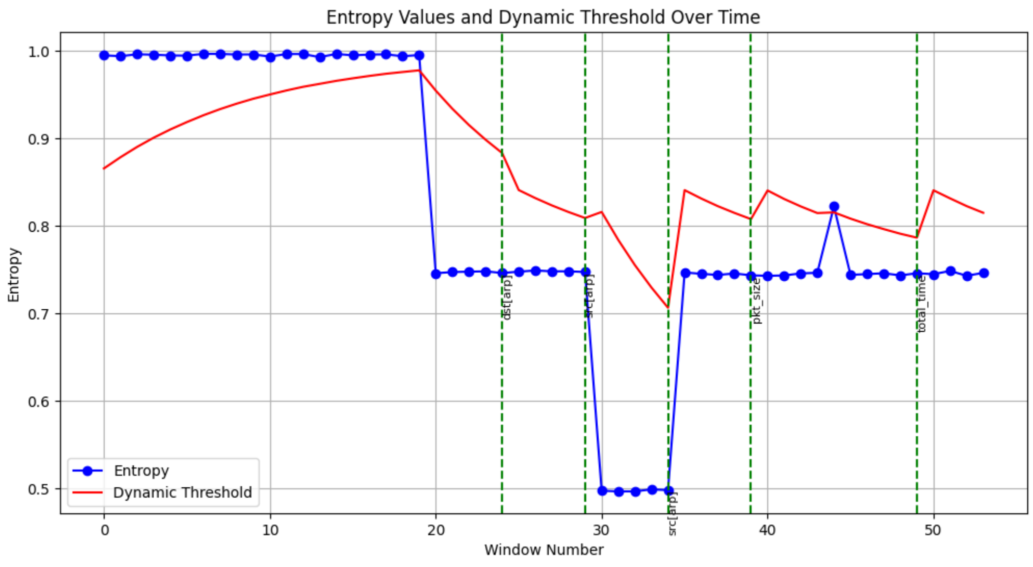


Figure 4.6: Attack Detection and Classification over Synthetic Dataset

The results of these performance metrics demonstrated the algorithm's high accuracy and reliability in detecting and mitigating various types of attacks within SDN environments. Specifically, the algorithm showed a strong capability to adapt to dynamic network conditions, promptly respond to potential threats, and maintain network security and resilience. This robust performance underscores the algorithm's potential for real-world application in enhancing the security of SDN infrastructures.

Table 4.2: Detection Results

Measure	Value
Sensitivity	0.8103
Specificity	0.9238
Precision	0.9115
Negative Predictive Value	0.8466
False Positive Rate	0.0297
False Discovery Rate	0.0322
False Negative Rate	0.1231
Detection Rate	0.9650
F1 Score	0.7889

We conducted experiments to assess the performance of the algorithm on a distinct and more extensive dataset, simulating a more realistic network environment. Following this, We generated graphs to visualize the outcomes of the algorithm's execution.

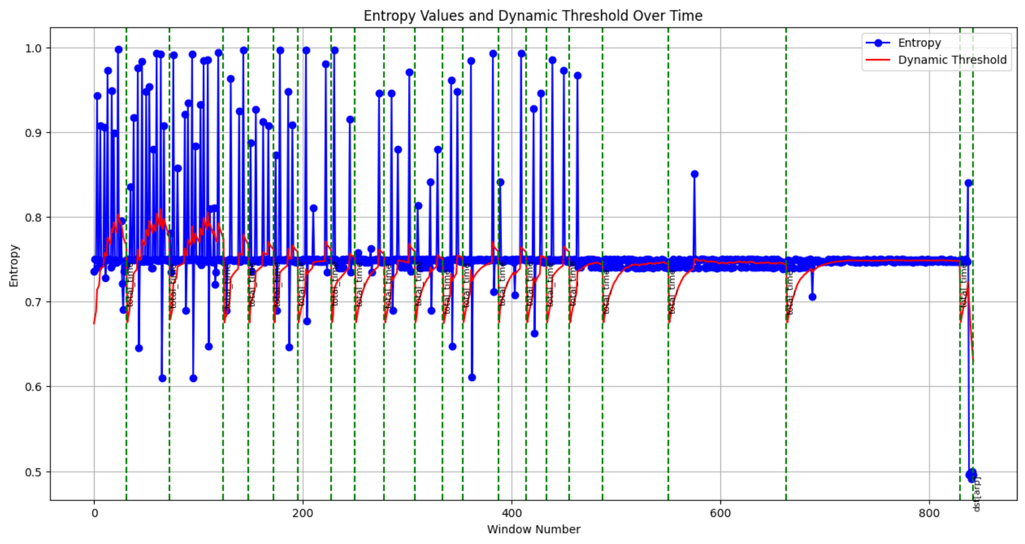


Figure 4.7: Attack Detection and Classification over SDN dataset

We can observe the distinction between attacks and normal traffic, as well as the factors contributing to the occurrence of attacks, by examining the features associated with each detected anomaly.

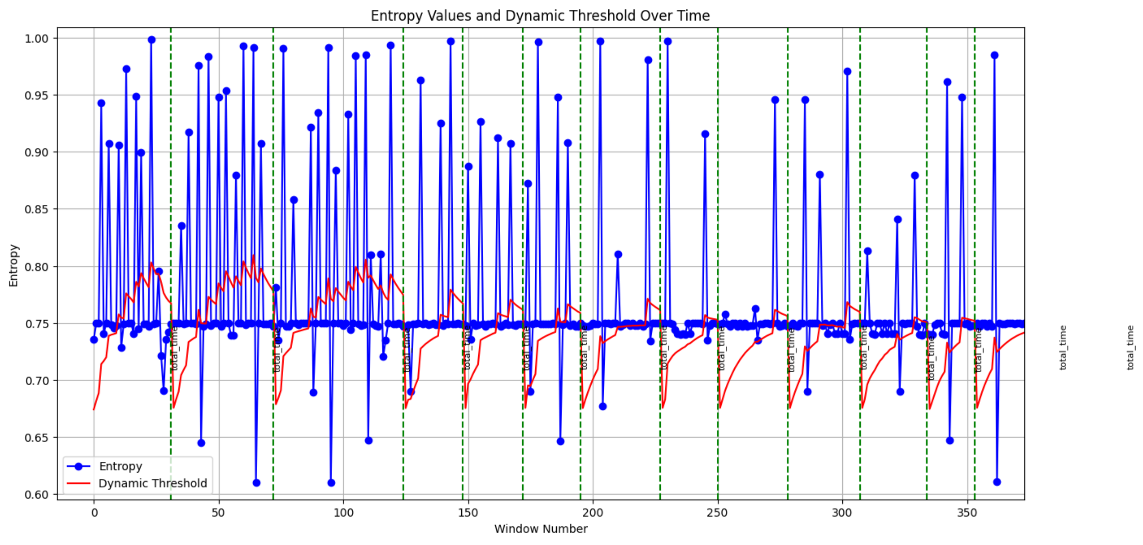


Figure 4.8: Close analysis of traffic

4.6 Chapter Summary

The algorithm was rigorously tested on a synthetic dataset with 1000 normal and 4000 attack packets. It used a dynamic threshold based on the Exponentially Weighted Moving Average (EWMA) of entropy values to identify anomalies. Upon detection, it executed mitigation procedures such as logging, updating a blacklist, and dropping suspicious packets. Key metrics like sensitivity, specificity (0.9238), and precision (0.9115) demonstrated high accuracy and reliability. The algorithm effectively adapts to dynamic network conditions, ensuring robust SDN security.

Chapter 5

Conclusions and Future Scope

5.1 Conclusion

The main findings and contributions of this research are summarized as follows:

- We introduced a novel algorithm to detect and mitigate flow table overloading attacks in Software-Defined Networking (SDN) environments.
- Our approach leverages entropy calculations on network traffic features, utilizing a sliding window mechanism to monitor and identify anomalies.
- By integrating Exponentially Weighted Moving Average (EWMA) for dynamic threshold adjustment, the algorithm adapts to varying network conditions, ensuring robustness against both high-rate and low-rate attacks.
- Comprehensive simulations and analysis demonstrated the algorithm's efficacy in accurately detecting anomalies while maintaining low false-positive rates.
- The proposed method enhances SDN network security by providing a scalable and lightweight solution capable of real-time deployment.
- The initial threshold calculation based on both normal and attack traffic ensures a balanced detection mechanism.
- This work contributes significantly to the field of SDN security, addressing critical vulnerabilities in flow table management.
- Overall, the algorithm offers a practical approach to safeguarding network infrastructures, providing an effective defense against flow table overloading attacks.

5.2 Future Scope

The proposed algorithm, while effective, opens several avenues for future research:

- Enhance the algorithm by incorporating machine learning techniques to predict and preemptively mitigate flow table overloading attacks.
- Investigate the integration of our algorithm with other SDN security frameworks for a comprehensive defense mechanism.
- Explore adaptive parameter tuning to further reduce the algorithm's computational overhead and improve response times.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, 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, 2014.

[2] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, “Defending against flow table overloading attack in software-defined networks,” *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 231–246, 2016.

[3] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, “Detecting saturation attacks based on self-similarity of openflow traffic,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 607–621, 2019.

[4] M. Zhang, J. Bi, J. Bai, Z. Dong, Y. Li, and Z. Li, “Ftguard: A priority-aware strategy against the flow table overflow attack in sdn,” in *Proceedings of the SIGCOMM Posters and Demos*, 2017, pp. 141–143.

[5] M. A. Aladaileh, M. Anbar, A. J. Hintaw, I. H. Hasbullah, A. A. Bahashwan, T. A. Al-Amiedy, and D. R. Ibrahim, “Effectiveness of an entropy-based approach for detecting low-and high-rate ddos attacks against the sdn controller: Experimental analysis,” *Applied Sciences*, vol. 13, no. 2, p. 775, 2023.

[6] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in *Recent Advances in Intrusion Detection: 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings 14*. Springer, 2011, pp. 161–180.

[7] M. J. Santos-Neto, J. L. Bordim, E. A. Alchieri, and E. Ishikawa, “Ddos attack detection in sdn: Enhancing entropy-based detection with machine learning,” *Concurrency and Computation: Practice and Experience*, p. e8021, 2024.

[8] S. Xie, C. Xing, G. Zhang, and J. Zhao, “A table overflow ldos attack defending mechanism in software-defined networks,” *Security and Communication Networks*, vol. 2021, pp. 1–16, 2021.

[9] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, “Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense,” *Security and Communication Networks*, vol. 2018, pp. 1–15, 2018.

[10] R. Neamah and W. Bhaya, “Security of software defined networks (sdn) against flow table overloading attack,” *Journal of Advanced Research in Dynamical and Control Systems*, vol. 11, pp. 752–759, 10 2019.

[11] Y. Shen, C. Wu, D. Kong, and Q. Cheng, “Flow table saturation attack against dynamic timeout mechanisms in sdn,” *Applied Sciences*, vol. 13, no. 12, p. 7210, 2023.

[12] J. Cao, M. Xu, Q. Li, K. Sun, and Y. Yang, “The attack: Overflowing sdn flow tables at a low rate,” *IEEE/ACM Transactions on Networking*, 2022.

[13] R. M. A. Ujjan, Z. Pervez, K. Dahal, W. A. Khan, A. M. Khattak, and B. Hayat, “Entropy based features distribution for anti-ddos model in sdn,” *Sustainability*, vol. 13, no. 3, p. 1522, 2021.

[14] N. M. AbdelAzim, S. F. Fahmy, M. A. Sobh, and A. M. B. Eldin, “A hybrid entropy-based dos attacks detection system for software defined networks (sdn): A proposed trust mechanism,” *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 85–90, 2021.

[15] H. A. AL-OFEISHAT, “Dynamic threshold generalized entropies-based ddos detection against software-defined network controller,” *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 3, 2024.