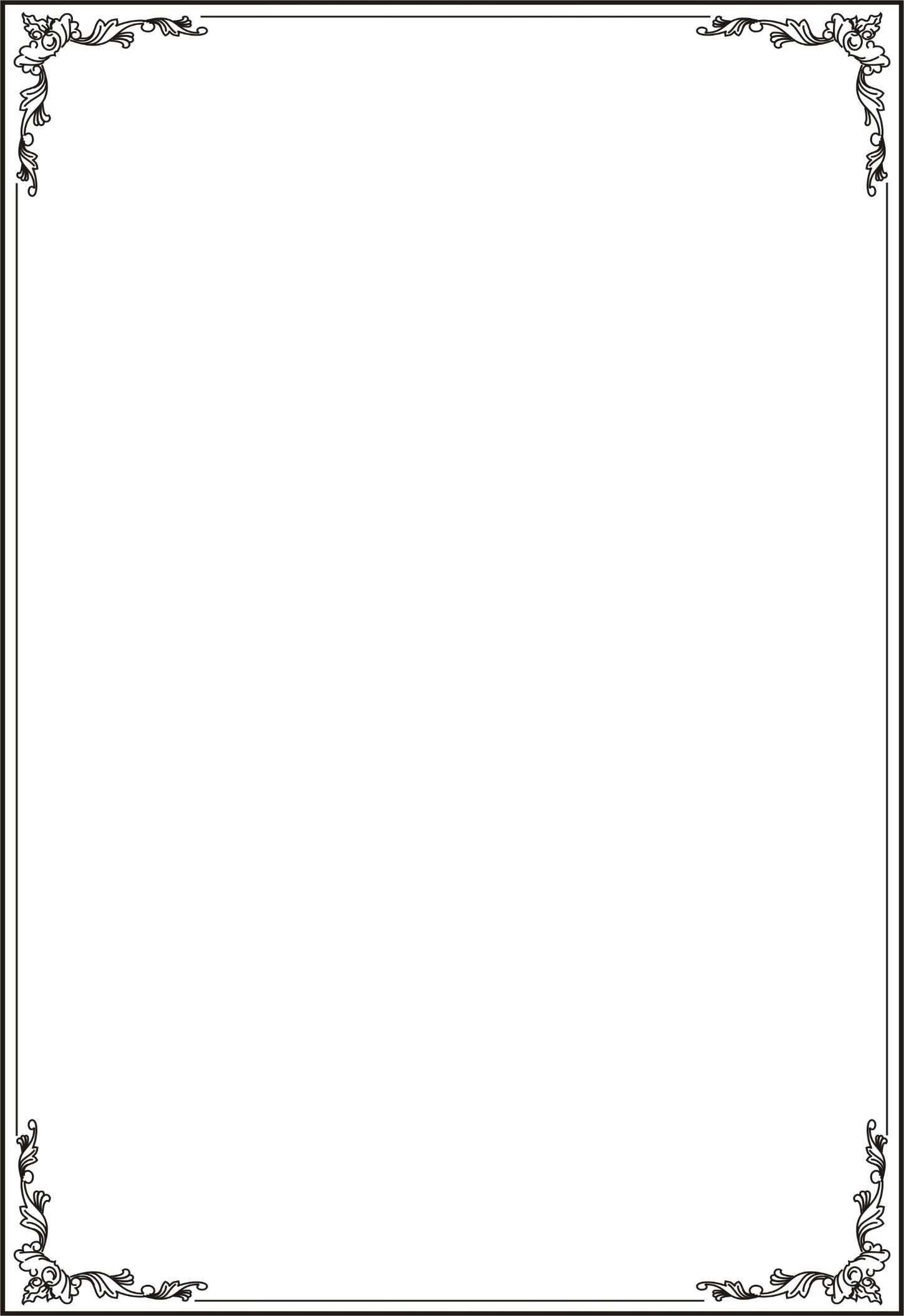
|  |
| --- |

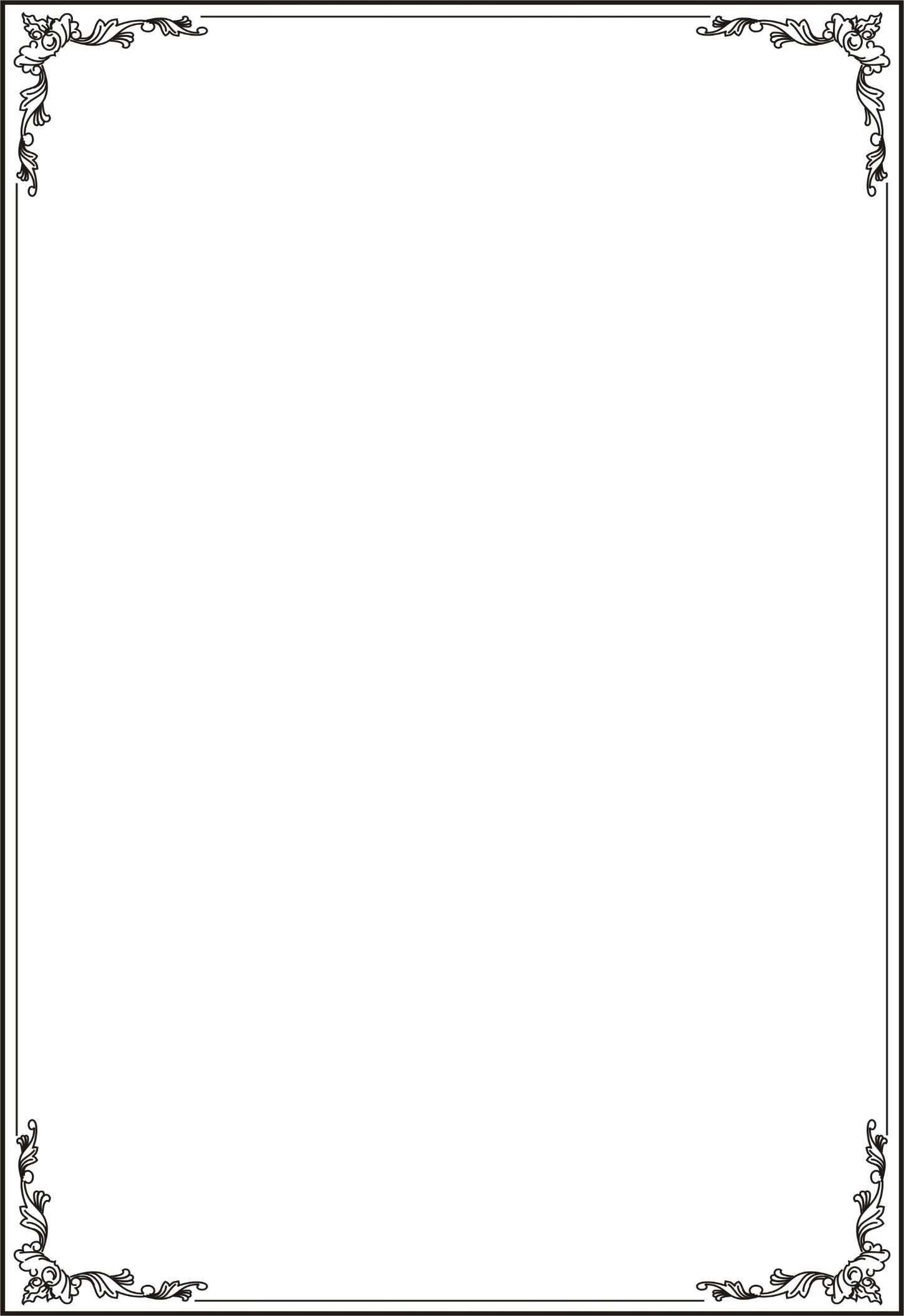
ĐẠI HỌC QUỐC GIA HÀ NỘI

**TRƯỜNG ĐẠI CÔNG NGHỆ**

**-----**🙞🙜🕮🙞🙜**-----**

**Logo

Description automatically generated**



Abstract

This research paper presents a multi-attack detection approach for Software-Defined Networking (SDN) networks. The proposed approach combines entropy-based anomaly detection with machine learning techniques to detect various types of attacks, including Distributed Denial-of-Service (DDoS), Man-in-the-Middle (MitM) and Slow-rate DoS attacks. The method uses the entropy method with the IP analysis to classify attacks. Entropy-based anomaly detection method measures the randomness of traffic flows and identifies abnormal traffic patterns, while machine learning algorithms, specifically the Random Forest classifier, are used to classify the traffic flows as normal or attack traffic. The proposed approach was implemented and evaluated on both SDN simulation and practical environment using the POX controller and OpenFlow protocol. The results show that the proposed approach can effectively detect DDoS, MitM, and Slow-rate DoS with high accuracy and low false positive rates. The proposed approach provides a practical and efficient solution for improving the security of SDN networks against various types of attacks.

**Keywords: SDN, DDoS attacks, MitM attacks, Slow-rate DoS, network security, statistical analysis, entropy, IP analysis.**

Table of Contents

[**Abstract ii**](#_heading=h.30j0zll)

[**Table of Contents iii**](#_heading=h.1fob9te)

[**List of Figures v**](#_heading=h.3znysh7)

[**1. Introduction 1**](#_heading=h.2et92p0)

[1.1. Research objectives 1](#_heading=h.tyjcwt)

[1.2. Research methodology 1](#_heading=h.3dy6vkm)

[1.3. Contribution of the research 1](#_heading=h.1t3h5sf)

[**2. Details of research report 1**](#_heading=h.4d34og8)

[2.1. Related work 1](#_heading=h.2s8eyo1)

[2.2. SDN Overview 1](#_heading=h.17dp8vu)

[2.2.1. SDN 1](#_heading=h.3rdcrjn)

[2.2.2. OpenFlow Protocol 2](#_heading=h.26in1rg)

[2.2.3. Pox Controller 3](#_heading=h.lnxbz9)

[2.3. Attacks in SDN Environments 4](#_heading=h.35nkun2)

[2.3.1. Distributed Denial of Services (DDoS) 4](#_heading=h.1ksv4uv)

[2.3.2. Man-in-the-Middle (MitM) 5](#_heading=h.44sinio)

[2.4. System model 5](#_heading=h.2jxsxqh)

[2.4.1. System model 5](#_heading=h.z337ya)

[2.4.2. Entropy 6](#_heading=h.3j2qqm3)

[2.5. Model Evaluation 7](#_heading=h.1y810tw)

[2.5.1. Simulation Environment 7](#_heading=h.4i7ojhp)

[2.5.2. Practical Environment 9](#_heading=h.2xcytpi)

[2.5.3. Evaluation 10](#_heading=h.1ci93xb)

[**3. Conclusion 10**](#_heading=h.3whwml4)

[**4. References 10**](#_heading=h.2bn6wsx)

[**5. Appendix 10**](#_heading=h.qsh70q)

List of Figures

[**Bảng 4.1. Đây là ví dụ của bảng 3**](#_heading=h.3as4poj)

# Introduction

## Research objectives

The research objectives for multi-attack detection in SDN networks are:

1. To propose a multi-attack detection approach for SDN networks that can detect different DDoS, MitM, and Slow DoS
2. To compare the proposed approach with existing techniques for attack detection in SDN networks.
3. To evaluate the performance of the method in a real SDN model

Overall, the research objectives aim to provide a practical and efficient solution for improving the security of SDN networks against multiple types of attacks and to contribute to the growing body of research on attack detection in SDN networks.

## Research methodology

To the main method to detect the multi-attack in our research is the combination of the IP analysis method and the Hybrid (SVM + Entropy) method. The following steps have been taken:

1. Data collection and preprocessing: Collect network traffic data from an SDN controller and preprocess the data by filtering out traffic and converting the data into suitable formats for analysis.

2. Feature extraction and selection: Extract relevant features from the network traffic data, such as packet size, packet arrival time, and source and destination IP addresses, and select the most relevant features using techniques such as principal component analysis and mutual information.

3. Model training and evaluation: Train machine learning models using the selected features and evaluate their effectiveness in detecting various types of attacks, including DDoS, MitM, and Slow DoS attacks, then use performance metrics to evaluate.

## Contribution of the research

On multi-attack detection in SDN networks contributes to the field of network security in several ways. Firstly, this is the first research to detect DDoS, MitM, and Slow DDoS in one model, furthermore, we propose a practical and efficient solution for detecting these attacks. Secondly, it addresses the growing concern about the vulnerability of SDN networks to various types of attacks. Next, the research provides recommendation for the practical deployment of the proposed approach in real-world SDN networks. Overall, the research contributes to the advancement of knowledge in the field of SDN security and provides a practical and effective solution for improving the security of SDN networks against multiple types of attacks.

# Details of research report

## Related Work

Related works for multi-attack detection in SDN networks include various techniques that utilize machine learning, rule-based or hybrid approaches to detect different types of attacks.

One commonly used technique is flow-based anomaly detection, which uses statistical methods to analyze network traffic and identify abnormal patterns. Some research has explored the use of entropy-based anomaly detection for detecting DDoS attacks in SDN networks. For example, the work by Rahman et al. (2019) proposed an entropy-based anomaly detection method for detecting DDoS attacks in SDN networks using the SDN controller.

Many methods have been proposed to detect DDoS in SDN []. In the same topic, the authors of [18] developed a technique for identifying DDoS attacks based on fixed entropy values of 50 packets []. After calculating, the entropy value will be compared with a pre-set threshold value. If it is below the threshold, then the possibility of an attack is indicated. Otherwise, the entropy value will be set to the current entropy calculation to avoid further inaccurate analysis. This enables the detection algorithm to adjust in adapt response to the characteristics of traffic flow.

In paper [], H. Lotfalizadeh et al suggested the use of real-time entropy to differentiate between normal and attack traffic. Each flow statistic is only applied to the associated time window. In other words, statistics of flows are retrieved for each time window without any data from earlier time windows. Consequently, the threshold will change over time and help the system detect the new flow of attack more accurately. The suggested approach was tested on 3-time windows of 10, 30, and 60 seconds in which the 10 seconds window provides the best result.

In addition, some research has proposed hybrid approaches that combine multiple techniques to improve the accuracy and effectiveness of attack detection. For example, the work by Duan et al. (2019) proposed a hybrid approach that combines entropy-based anomaly detection and machine learning for detecting DDoS attacks in SDN networks.

Overall, the related works demonstrate the potential of using different techniques to detect various types of attacks in SDN networks. However, there is still a need for more research to improve the accuracy and efficiency of attack detection in SDN networks, especially for multi-attack scenarios.

## SDN Overview

### SDN

Software-Defined Networking (SDN) is a networking paradigm that separates the control plane from the data plane in network devices. This separation allows network administrators to centrally manage and control the network, rather than having to configure each individual device separately. In SDN, the control plane is moved from network devices, such as switches and routers, to a centralized controller. The controller communicates with the devices using a protocol such as OpenFlow, allowing it to program the forwarding rules and policies for each device. The architecture of SDN is visualized in Figure. 1

Diagram

Description automatically generated

SDN provides several benefits over traditional networking approaches, including increased flexibility, scalability, and agility. With SDN, network administrators can easily change network policies and configurations, allowing them to respond quickly to changing network conditions or business requirements. SDN also provides a more granular control over network traffic, allowing administrators to apply policies to individual flows or packets.

SDN has been applied in a variety of use cases, such as network virtualization, traffic engineering, and security. In network virtualization, SDN is used to create multiple logical networks on top of a physical network infrastructure, allowing multiple tenants to share the same physical resources. In traffic engineering, SDN is used to optimize network performance by dynamically adjusting network paths and traffic flows based on network conditions. In security, SDN can be used to detect and mitigate cyber-attacks by monitoring network traffic and enforcing security policies.

Despite its many advantages, SDN also faces several challenges, such as security, interoperability, and scalability. SDN controllers can be a single point of failure and are vulnerable to cyber attacks, making it crucial to secure them against potential threats. Additionally, interoperability can be an issue when implementing SDN, as different vendors may use different implementations of the OpenFlow protocol. Finally, as networks become larger and more complex, scaling SDN can become a significant challenge.

Overall, SDN represents a significant shift in networking architecture, allowing for greater flexibility and control over network resources. As the demand for more agile and dynamic networks continues to grow, SDN is likely to play an increasingly important role in the future of networking.

### OpenFlow Protocol

The OpenFlow protocol is a key technology in the field of software-defined networking (SDN), which has emerged as a promising approach to network management and optimization. The protocol defines a standardized way for a network controller to communicate with the forwarding elements (i.e., switches) in a network, allowing for centralized control and management of network resources. By separating the control plane from the data plane, OpenFlow enables greater flexibility and programmability in network management, as administrators can dynamically configure forwarding rules and traffic flows from a single controller.

One of the key benefits of the OpenFlow protocol is its ability to provide granular control over network traffic. This is achieved through the use of flow tables, which store information about the various flows of traffic in the network. The controller can dynamically modify these flow tables in response to changing network conditions, allowing it to optimize traffic routing and manage congestion. OpenFlow also enables the creation of customized forwarding rules, which can be based on various factors such as source and destination addresses, port numbers, or even the contents of specific packets.

Another important aspect of the OpenFlow protocol is its support for network virtualization. By abstracting the physical network infrastructure and providing a virtual network layer, OpenFlow enables the creation of multiple logical networks that can be dynamically configured and managed. This is particularly useful in cloud computing environments, where multiple tenants may share the same physical infrastructure but require separate and secure virtual networks.

OpenFlow has been widely adopted in both academic and industrial settings, and has contributed to the development of new approaches to network management and optimization. The protocol has been used in a variety of applications, including data center networking, mobile networks, and even industrial control systems. As the demand for more flexible, efficient, and scalable network management solutions continues to grow, it is likely that OpenFlow and other SDN technologies will play an increasingly important role in the future of network architecture and design.

### Pox Controller

The POX controller is a popular open-source controller for Software-Defined Networking (SDN) that is written in Python. It provides a flexible and easy-to-use framework for developing custom SDN applications and has gained widespread adoption in both industry and academia. The POX controller supports multiple OpenFlow versions, making it compatible with a wide range of network devices, and provides a rich set of APIs for interacting with the network topology, forwarding rules, and network events.

One of the key features of the POX controller is its modular architecture, which makes it easy to extend and customize. Developers can create new modules to implement additional features and services, such as network virtualization, traffic engineering, and security. The POX controller also includes a range of built-in modules for common tasks, such as topology discovery, network monitoring, and traffic management.

The POX controller has been used in a variety of research projects, including network function virtualization (NFV), network slicing, and traffic engineering. In NFV, the POX controller is used to create virtual network functions (VNFs) that can be dynamically deployed and managed in response to changing network conditions. In network slicing, the POX controller is used to create multiple logical networks on top of a physical network infrastructure, allowing multiple tenants to share the same physical resources. In traffic engineering, the POX controller is used to optimize network performance by dynamically adjusting network paths and traffic flows based on network conditions.

One of the benefits of the POX controller is its ease of use, especially for developers familiar with Python. The POX controller provides a simple and intuitive programming model, allowing developers to quickly prototype and test new applications. It also includes a range of built-in tools and utilities for debugging and testing, making it easier to troubleshoot and optimize applications.

While the POX controller has gained widespread adoption, there are also other open-source controllers available for SDN, such as Ryu and OpenDaylight, each with their own strengths and weaknesses. The choice of the controller will depend on the specific requirements of the network and the applications running on it. However, the flexibility, ease of use, more straightforward and effective detection algorithm, and extensibility of the POX controller make it a popular choice for researchers and developers looking to build custom SDN applications.

## Attacks in SDN Environments

### Distributed Denial of Services (DDoS)

Distributed Denial of Service (DDoS) attacks have been a major threat to network security for many years. However, with the emergence of software-defined networking (SDN), the nature of these attacks has evolved. DDoS attacks in SDN networks are particularly concerning due to the centralized nature of SDN controllers, which provide a single point of failure and control for the entire network. These attacks aim to overwhelm the network with a flood of traffic, causing legitimate traffic to be dropped or delayed, and ultimately leading to service disruption.

One of the main challenges in facing DDoS attacks in SDN networks is the need to identify and distinguish legitimate traffic from attack traffic. Traditional detection techniques, such as signature-based detection, are not always effective in these scenarios, as attackers can use various evasion techniques to bypass these methods. Therefore, new techniques are required that can accurately and efficiently identify and mitigate DDoS attacks in SDN networks.

Therefore, new techniques are required that can accurately and efficiently identify DDoS attacks in SDN networks. Machine learning algorithms, such as Support Vector Machines (SVMs) and Random Forests, have been shown to be effective in detecting DDoS attacks based on features extracted from packet headers, flow characteristics, and behavioral patterns. These algorithms can be trained using supervised or unsupervised learning techniques and can be implemented in the SDN controller to provide real-time detection and mitigation of DDoS attacks. Another effective method that can be used to detect DDoS attacks is the statistical method with the use of Entropy. The Entropy will determine the randomness of the IP address and therefore identify the IP with the abnormally large amount of present to confirm an attack.

### Man-in-the-Middle (MitM)

A Man-in-the-Middle (MitM) attack is a type of cyber attack in which an attacker intercepts communication between two parties and can potentially modify or manipulate the communication. In a MitM attack, the attacker positions themselves between the two parties, acting as a relay or proxy for communication. The attacker can then intercept, read, and potentially modify the communication before forwarding it to the intended recipient. MitM attacks can occur in a variety of scenarios, including over wireless networks, through public Wi-Fi hotspots, or even via compromised network devices.

MitM attacks can be carried out using a variety of techniques, such as ARP spoofing, DNS spoofing, or SSL stripping. In ARP spoofing, the attacker sends false Address Resolution Protocol (ARP) messages to redirect traffic to their own device. This allows the attacker to intercept and manipulate traffic between the two parties. In DNS spoofing, the attacker intercepts and modifies Domain Name System (DNS) requests to redirect traffic to their own server. This can allow the attacker to intercept and modify communication between the two parties.

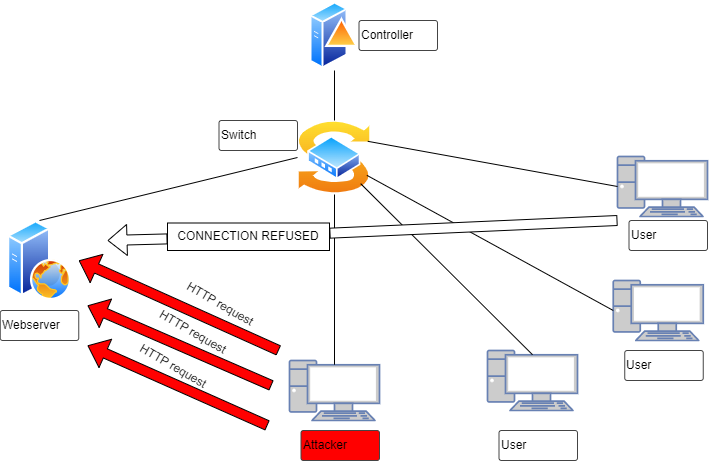
MitM attacks can be used for a variety of malicious purposes, such as stealing sensitive information, injecting malware, or impersonating the victim. MitM attacks can be difficult to detect because the attacker is positioned between the two parties and can potentially modify the communication without either party realizing it.

To prevent MitM attacks, several techniques can be used, such as implementing secure communication protocols, using digital certificates to verify the identity of the communication partner, and monitoring network traffic for suspicious activity. However, MitM attacks remain a persistent threat in the cyber security landscape, and attackers continue to develop new and sophisticated techniques to carry out these attacks.

### Slow-rate DoS attacks

A Slow Denial of Service (Slow DoS) attack is a type of cyber attack that aims to consume the resources of a target system or network by exploiting vulnerabilities in its infrastructure or applications. Unlike traditional DoS attacks that overwhelm a target with a large volume of traffic in a short period of time, Slow DoS attacks use low-rate traffic or exploits that slowly drain the target's resources over time, causing the system to become unavailable or significantly slower to respond.

Slow DoS attacks can take many forms, including TCP SYN floods, HTTP GET/POST floods, and low-and-slow attacks, among others. TCP SYN floods are designed to flood the target system with a large number of connection requests, causing it to allocate resources to incomplete connection requests, which eventually leads to a denial of service. HTTP floods are similar but are targeted at web applications, using GET or POST requests to exhaust the target's resources. Low-and-slow attacks, on the other hand, are designed to send a small number of requests that are spaced out over time, making them harder to detect and mitigate.



Slow DoS attacks can be difficult to detect and mitigate, as the traffic can be indistinguishable from legitimate traffic. Furthermore, they can be launched from a single machine or from a botnet, making it harder to block the source of the attack. However, there are several techniques that can be used to detect and mitigate Slow DoS attacks.

1. Baseline Traffic Analysis: Monitoring the network traffic and identifying normal patterns of traffic can help detect Slow DoS attacks. By establishing a baseline of normal traffic patterns, abnormal traffic that is characteristic of a Slow DoS attack can be detected.
2. Anomaly Detection: Using machine learning algorithms and other statistical techniques, anomaly detection can help detect Slow DoS attacks. The algorithm can analyze traffic patterns, traffic volume, and resource utilization to identify deviations from normal behavior that may indicate an attack.
3. Packet Inspection: Deep packet inspection (DPI) is a technique that involves analyzing the contents of network packets to identify patterns that may indicate a Slow DoS attack. DPI can identify patterns that are not visible at the network layer, such as slow HTTP requests or TCP SYN packets that are sent at irregular intervals.
4. Behavior-based Detection: Behavior-based detection involves monitoring the behavior of network connections and identifying connections that are exhibiting suspicious behavior. For example, connections that are sending a large number of requests or using unusual protocols may be indicative of a Slow DoS attack.
5. Resource Monitoring: Monitoring the utilization of network resources such as CPU, memory, and bandwidth can help detect Slow DoS attacks. Slow DoS attacks consume resources over time, and monitoring the utilization of these resources can help identify abnormal patterns that may indicate an attack
6. Traffic Shaping: Traffic shaping involves regulating the flow of traffic to prevent network congestion and can help detect Slow DoS attacks. By monitoring traffic patterns and limiting the amount of traffic that can be sent, traffic shaping can help prevent an attacker from consuming all available network resources.

In summary, detecting Slow DoS attacks requires a combination of techniques that involve monitoring network traffic, analyzing traffic patterns, and identifying deviations from normal behavior. By using these techniques, organizations can detect Slow DoS attacks and implement appropriate measures to mitigate the impact of the attack. For that, we will develop a module based on traffic analysis and machine learning algorithms, which can correctly detect Slow DoS attacks in SDN.

## System model

### System model

### Entropy

Entropy is a measure of randomness or uncertainty in a system, and it has been used as a feature for detecting Distributed Denial of Service (DDoS) attacks in network traffic. Entropy-based detection methods rely on the observation that DDoS attacks often generate traffic with higher entropy than normal traffic. This is because DDoS attacks often involve a large number of compromised devices generating traffic with similar characteristics, leading to a less predictable pattern of traffic. By measuring the entropy of packet payloads, packet headers, or other features of network traffic, it is possible to identify patterns of traffic that are characteristic of DDoS attacks.

Consider a collection with items ( ≤ ) that represent a window of IP addresses and represents the number of distinct destination IP addresses in the incoming packet headers:

(1)

Then, the entropy value is determined using to the following formula:

(2)

The probability of an IP address in W is:

(3)

Where represents the number of IP addresses in while is the size of the (the total IP address). stands for the window's size.

In (2), if decreases and approaches zero, it means that there is an anomalous event is occurring throughout the system. Whereas, in normal event, packets are sent to different destinations with almost the same speed, no destinations receive a disproportionately large number of packets compared to other destinations. As a result, will be in an optimal average approximated state.

In [27], a static test threshold is chosen based on the execution of many attacks in order to detect a DDoS attack.

(4)

In (4), stands for the sample mean while the remaining is called the margin of error: is a confidence coefficient, is the sample standard deviation and is the sample size. The chosen confidence level is 95% ( = 1.9599).

Firstly, we will find the difference in which is calculated as normal average traffic minus the reliability interval and is equivalent to the average entropy value in attack event plus a confidence interval. Finally, the static threshold is determined as . This static threshold is fixed and any entropy value below it will be regarded as an ongoing attack [7].

However, this static threshold value based on previous attack data. As a result, it limits the ability to adjust the threshold for identifying new attacks. In this study, the threshold that we utilized will not be fixed but it will fluctuate over time based on the changing of entropy value in incoming traffic. Once the entropy values have been calculated, it will be stored in the window. Based in these parameters, we will calculate the average entropy value and the standard deviation for each window.

(5)

(6)

In (5) and (6), is the entropy value over period which denotes the number of windows calculated using the previously described in (2). Depending on the parameters determined above, we consider a dynamic threshold value with the formula defined as follows:

(7)

In (7), and denote the average entropy value and standard deviation at the time of , respectively. The normal distribution indicates that 95% of entropy values will fall within the range . These values, which are smaller than , will not significantly affectthe result. Then, we can choose for this system based on this fact. In [8], is a constant value and equal to -2 according to experiment.

### Feature extraction

With DDoS attacks,

In a slow DoS attack, the hacker attempts to create as many connections as possible to the webserver. Each of these connections will try to maintain the minimum conditions to keep it from being deleted. Based on the advantages of OpenFlow Switch, we propose the following characteristics that can distinguish between when an attack is occurring and when it is not. First, we collect the necessary information about flow entries in the flow table. We then preprocess and extract four features as follows:

1. "avePackets" represents the average number of packets transmitted per flow. Hackers during slow DoS attacks attempt to establish as many connections as possible by sending minimum packets that can maintain the connection open without deletion. A decrease in the number of packets per flow over time can indicate a slow DoS attack.
2. "aveBytes" represents the average number of bytes transmitted per flow. In a slow DoS attack, the header is divided into several parts and sent to the web server at a very slow rate. The web server collects enough parts of the header to respond to the request. As a result, the average number of bytes per flow may decrease, indicating a slow DoS attack.
3. "flowPerIP" represents the number of flow entries per IP address. During a slow DoS attack, the number of flow entries per IP address may increase since the hacker will send and keep many connections alive that are displayed in a flow table. A significant increase in this metric may indicate a slow DoS attack.
4. "newFlowPerTime" represents the number of new flows created within a 10-second time window. Slow DoS attacks can be detected by an increase in the number of new flows created within the 10-second window, which suggests that an attacker is attempting to overwhelm the system with many new connections. The 10-second time window matches the idle timeout of flow entries, ensuring that flows are properly deleted and new ones are created.

By extracting these metrics, network administrators can detect and respond to slow DoS attacks based on a machine learning model.

## Model Evaluation

### Simulation Environment

Our simulation was carried out on a Lenovo computer with an Intel® Core™ i5 - 9300H processor operating at 1.2 GHz and 8 GB of DDR4 RAM 2666 MHz, along with Ubuntu 20.04 as the operating system. We chose Mininet as a network emulator with a POX controller for simulation purposes. With Mininet, we could create an attack on a virtual server and examine the outcomes of our DDoS attack detection model. Then, we apply our proposed method to detect DDoS in this model.

Diagram

Description automatically generated

In order to make the host communicate with each other through POX, we use the module in Pox. This module offers layer 3 learning capabilities by storing a list of IP address information between nodes. will analyse and extract the IP address from each new packet that comes in. This information will be compared with the list and if there is no similar path, the module will start the ARP protocol to start the request. In addition, we edited integrated algorithms that make it possible for the POX controller to calculate entropy values and parameters needed to detect attacks when there is an unusual change in incoming traffic.

Scapy handled packet initialization and transmission in the system. Scapy is used to generate UDP packets and spoof their source IP addresses to simulate an attack and normal traffic in the simulation system. The hosts in the model are given IP addresses that increase gradually, starting from 10.0.0.1.

1. *Phase 1: The system is in normal state:* In normal state, we use a host to initiate traffic and distribute packets to the whole system. The packet is sent every 0.1 second with a destination port of 80 and a source port of 2. 500 packets which equivalent to 10 windows will be delivered in all during a single run.

We use formula (2) and (3) to determine the current entropy in a window of 50 packets. Formulas (5) and (6) are used, respectively, to calculate Average Entropy and Standard Deviation. The dynamic entropy threshold is then calculated using the above value and formula (7).

For instance, the immediate entropy value is almost 0 with 50 identical destination IP addresses. In contrast, when there are 50 separate IP addresses in a window, this figure peaks at about 1.5.

In normal event, packets are transmitted to a wide range of network destination addresses. Therefore, the randomness will increase as well as the entropy value at that time. As the immediate entropy value exceeds the dynamic entropy threshold value, the system can conclude that the system is in normal state.

1. *Phase 2: The system is in a State of Attack:* There are 2 attack scenarios that we perform in Phase 2 which related to different attack intensities on the system, 50% and 75%, respectively. The rate of an attack is determined by:

(8)

In (8), and are the period of time where attack traffic and normal traffic occur, sequentially. In the system, normal traffic is randomly forwarded to all hosts, whereas attack traffic is only intended for one host.

First, we launch a 50% attack rate on a host on 10 times. It will allow the controller to detect any attack with packets accounting for 50% of incoming traffic or more. Then the higher-rate tests of 75% were performed on a host to examine a more focused attack. The changes in entropy between the two events can be seen more clearly in these simulations.

### Practical Environment

In the preceding session, we detect the DDoS attack in a simulation environment. In this section, the model will be put into practice on the Aruba Switch 2930F in which OpenFlow protocol is enabled.

We build a practical topology with 1 controller, 2 switch, and 8 hosts as shown in Figure 4. By using the IP address and listen port of the controller interface, we can acquire the flow status of the switch and calculate entropy. The normal and attack script is implemented as same as in the simulation environment. Host 10.10.0.6 produce samples of normal traffic and forward to the whole network. The attack scenario will be implemented using scapy. Host 10.10.0.3 is the attacker, from there we use scapy to flood UDP packets to the target, host 10.10.0.7. The network is affected by the DDoS and can not communicate as normal.

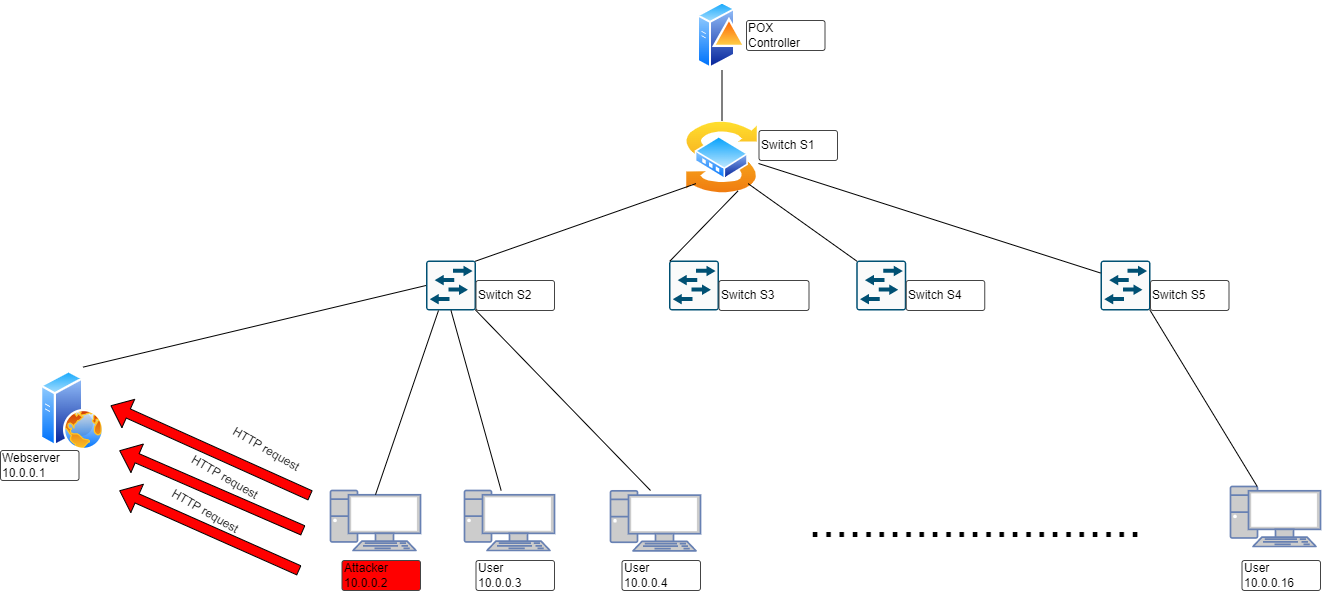
Diagram

Description automatically generated

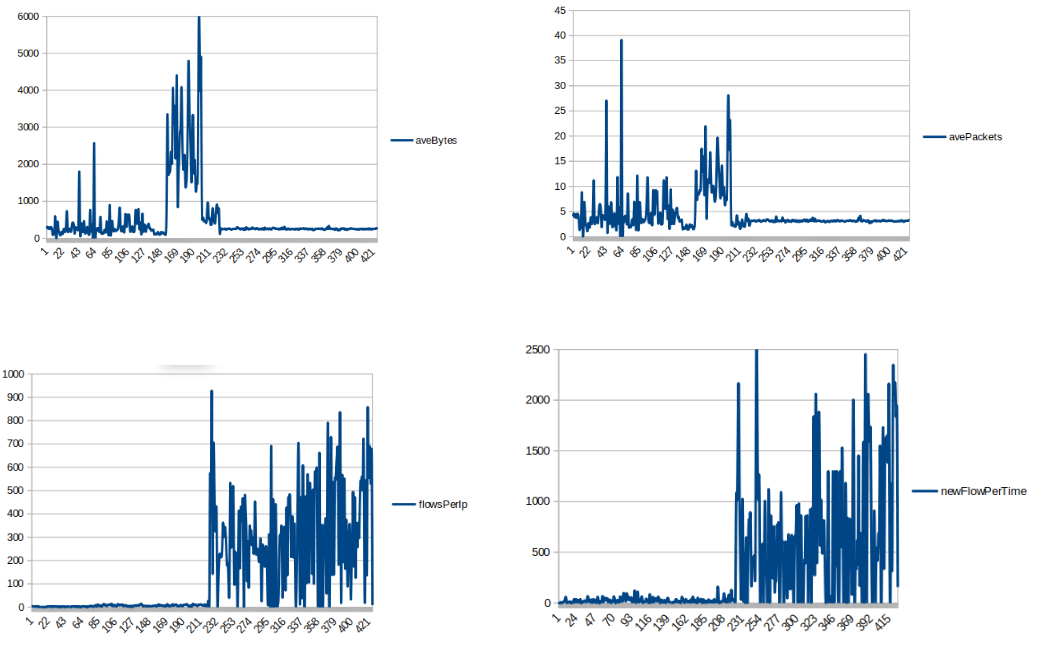
### Evaluation

1. DDoS attacks
2. MitM attacks
3. Slow DoS attacks

We performed our method in a topology as in the figure below



We installed a webserver on host h1 (10.0.0.1), which is a simple webserver that we developed ourselves. It contains a webpage and can handle a customizable number of simultaneous connections. In this study, we set a limit of 150 concurrent connections for the webserver (similar to the Apache webserver). If the number of connections exceeds this limit, it will be unable to process them and requests to the webpage will be denied. Host 2 (10.0.0.2) serves as the attacker, continuously sending HTTP requests to the web server every 15 seconds. Each time requests are created, we randomly generate between 200 and 1000 requests simultaneously, ensuring that they are within the web server's threshold. The remaining users send requests to display the pre-installed webpage on the webserver. We processed and modified the attack traffic and normal traffic generation software according to our requirements. After running and measuring the proposed features, we obtained the results as shown below.



Our team generated legitimate traffic during the period from cycle 1 to cycle 230, with each cycle being a sliding window of 10 consecutive captures. It is easy to see that the aveBytes value varies significantly and fluctuates between 700 and 6000 bytes on average per stream, which is higher than during an attack because data is continuously sent and received through the streams. On the other hand, when an attack occurs, the value fluctuates around 200 bytes per stream, which is the minimum value that a hacker can maintain to prevent the streams from being deleted and disconnected.

Moving on to the avePackets value, this represents the average number of packets per stream. When normal, it fluctuates between 5 and 30 packets per stream, and during an attack, it stabilizes at around 4 packets per stream. Hackers try to send the minimum number of packets possible to maintain the connection, which means that the packet size will be as small as possible.

The flowsPerIp value also reflects the difference when abnormalities occur in the network. During an attack, the number of flows generated per IP address is very high. We take an average value because in many cases, there may be multiple attackers attacking a web server. This value fluctuates greatly when an attack occurs, ranging from 100 to 700 connections per IP address.

The next value is newFlowPerTime, which represents the number of new flows created within a certain time period, specifically 10 seconds. When connections are deleted, hackers continuously create new connections in large numbers. It is clear that during an attack, the number of new connections created fluctuates between 400 and over 2000 connections.

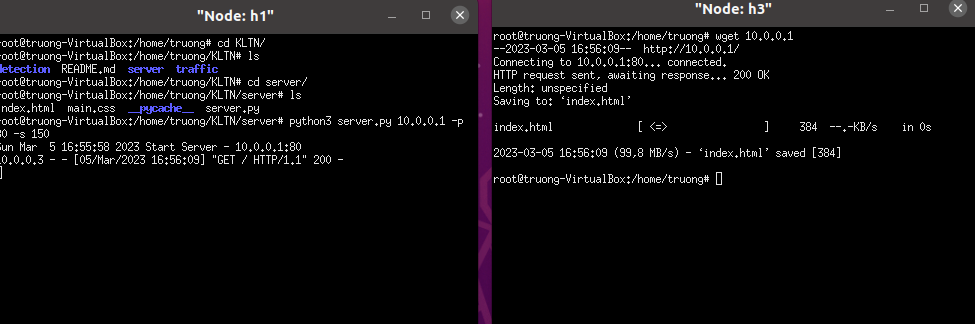
After surveying these four characteristics, we created a dataset with 1000 samples representing normal traffic and 1000 samples representing slow attacks. We then chose the SVM machine learning model for classification, based on the performance of the model as measured by the statistics we collected, as follows:

|  | Precision | Recall | F1-Score |
| --- | --- | --- | --- |
| Normal | 92% | 100% | 96% |
| Attack | 100% | 92% | 96% |
|  |  |  |  |
| Accuracy | 96% | | |

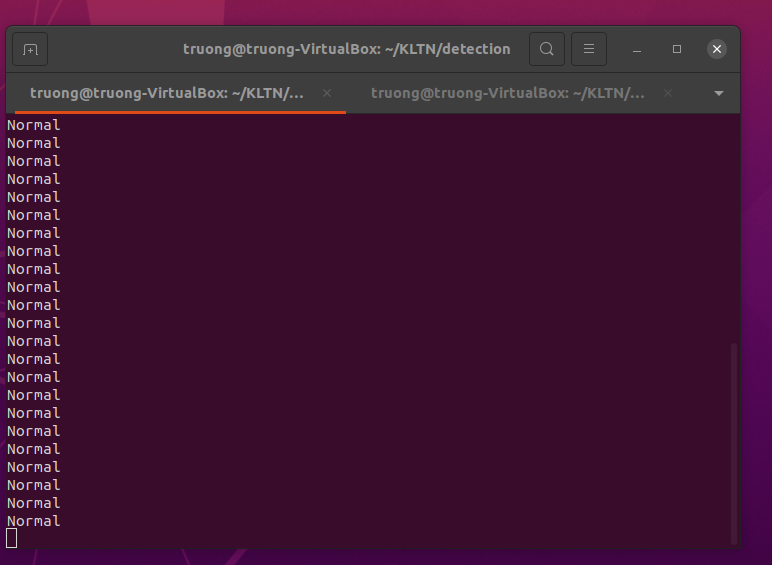
The presented results show that the SVM model achieved high accuracy and performance in detecting Slow DoS attacks. The model achieved an overall accuracy of 96%, indicating that it correctly classified 96% of the instances in the test dataset. The precision for the normal class was 92%, indicating that when the model predicted that an instance belonged to the normal class, it was correct 92% of the time. The recall for the normal class was 100%, indicating that the model correctly identified all instances that actually belonged to the normal class. The F1-score for the normal class was 96%, which is the harmonic mean of precision and recall and provides an overall measure of the model's ability to correctly classify the normal class.

For the attack class, the precision was 100%, indicating that when the model predicted that an instance belonged to the attack class, it was correct 100% of the time. The recall for the attack class was 92%, indicating that the model correctly identified 92% of the instances that actually belonged to the attack class. The F1-score for the attack class was also 96%, indicating that the model's performance in detecting attacks was similarly high as its performance in detecting normal instances.

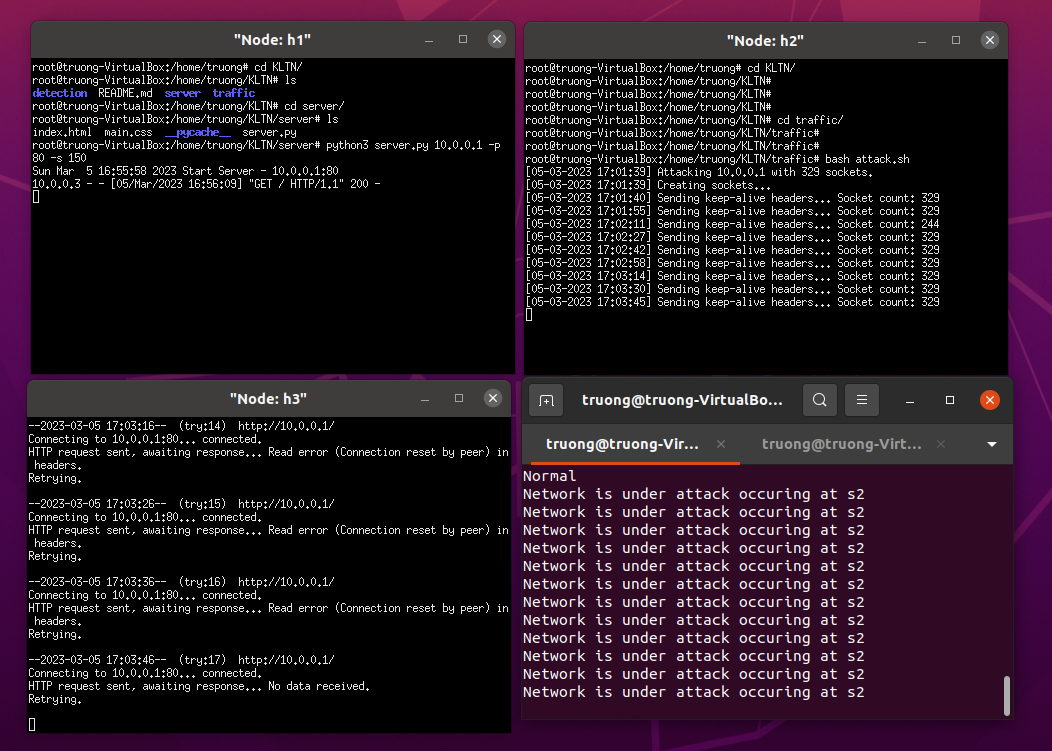
Overall, the model's high accuracy and performance in detecting both normal and attack instances suggest that it can be effectively integrated and deployed as a module within an SDN network to detect Slow DoS attacks in real time. The presented figure illustrates an example scenario where host 3 sends a GET request to retrieve an HTML file from host 1.



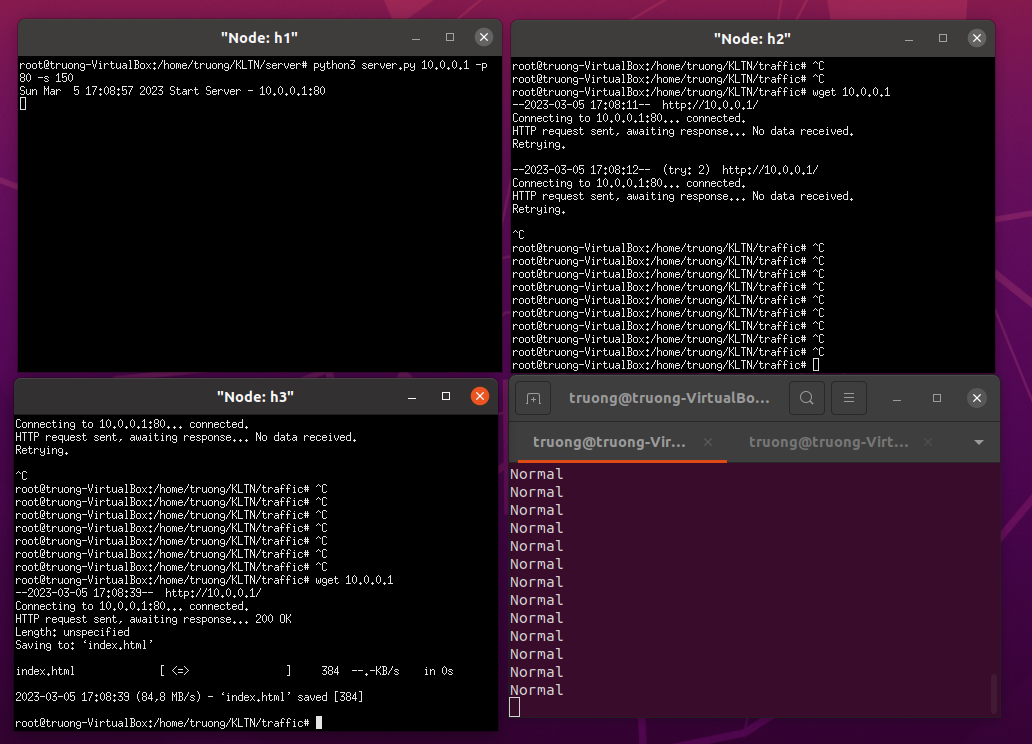
The request is received and returned with a successful response code of 200. At the same time, the detection module also reports that the network is in a normal state.



After that, we sent attack traffic from host 2 to the web server. At this time, host 3 was unable to access the web server and was continuously denied due to host 2 sending requests exceeding the web server's allowed threshold (up to 329 connections at the same time). Additionally, the detection module also issued a warning that the network was under attack at switch number 2 (s2), where host 2 was connected and sending data.



After host 2 stopped the attack, host 3 was able to send requests to access the web server normally.



The estimated response time for our module to detect the abnormality in the network is about 10 seconds after host 2 performs the attack. This time frame is fast enough to detect a slow DoS attack because of the nature of this type of attack, which is very similar to normal traffic, making it much harder to classify compared to traditional DDoS attacks.

The slow DoS attack can go unnoticed for a long time and has the potential to cause significant damage to a network by consuming its resources, making it inaccessible to legitimate users. Therefore, early detection is crucial to mitigate the damage caused by the attack. In our case, the SDN network with the integrated module is capable of detecting and preventing slow DoS attacks, providing a higher level of security to the network.

# Conclusion

# References

[1]

[2]

# Appendix