UNIVERSITY OF SURREY

SDN Security Using Machine Learning Algorithm

by

Bharadwaj Phani Datta Mahanthi

URN: 6671703

A dissertation submitted in partial fulfilment of the requirements for the award of

MSc Data Science – February 2021/23

Department of Computer Science

Faculty of Engineering and Physical Sciences

Supervisor: Dr Mohammad Shojafar

Declaration

I confirm that the submitted work is my work. No element has been previously submitted for assessment, or where it has, it has been correctly referenced. I have also clearly identified and fully acknowledged all material entitled to be attributed to others (whether published or unpublished) using the referencing system in the programme handbook. I agree that the University may submit my work to check this, such as the plagiarism detection service Turnitin® UK. I confirm that I understand that assessed work that has been shown to have been plagiarised will be penalised.

ABSTRACT

Software Defined Network is one of the most used technologies in the current world, it has so many uses, and at the same time, it is the most vulnerable to poisoning attacks. SDN is an approach in network management to bring all the control to a centralised plane rather than a distributed network. It has simplified networking over the past few years because, in traditional networking, you must buy many resources and program them individually. Still, in SDN, you can program as many times as you want, and the software decouples the network and data planes.

The main agenda of the SDN is to bring all the network components into one centralised network intelligence by disassociating the data plane from the control plane. The decoupling helps 5G networks to manage their infrastructure vastly because SDN separates the data plane and control plane, which allows operators to manage the features from a centralised location. The SDN capabilities support network slicing, which is one of the essential aspects that help 5G networks as it separates a single physical network into multiple virtual networks with the help of virtualisation, which share the same network infrastructure. Because of this feature, the network operator can provide different services based on the customer needs and helps to maintain Quality of Service (QoS).

Because of its features and importance, it is the primary target for cyber attackers, the main kind of attacks like Host Hijacking, ARP poisoning, and Link Latency Attacks. The attacks like DDoS, Network Manipulation, Traffic Diversion, Side Channel attack and App Manipulation are common attacks that will occur regularly and often. To provide security and measures for SDN, there is so much research going on to prevent attacks and provide better protection for the SDN and its users; some of them are TopoGuard and TopoGuard Plus, which are currently giving defence mechanisms for the SDN. In this research, I have used machine-learning techniques to identify the threat better, provide security, and prove that the attackers can override TopoGuard Plus at some point.

It will be a potential threat to SDN users and enterprises who depend on SDN technology, and it does not mean that SDN is weak; the proof-of-concept framework by network function virtualisation (NFV) for SDN proves that tenant isolation in SDN provides a high level of security in multi-tenant NFV cloud. It also proves that changing network configuration regularly and updating the resources through regular patches reduces the chances of an attacker identifying the exact location of the host in the cloud environment.

Acknowledgements

Firstly, I would like to acknowledge the efforts of my supervisor, Dr Shojafar Mohammad, who has supported me in undertaking this project through regular meetings, as well as teaching valuable skills, specifically through the SDN, Mininet, HTML, and Controllers; I am forever grateful for teaching me and spending his valuable time to prepare and guide through me the concepts. Secondly, I would like to acknowledge Soltani Sanaz, who helped me to learn all the concepts and supported my work and research, helping me from scratch to the final documentation and thanking her for spending her valuable time on me to teach and make me understand the concepts whenever the professor is not available.

Table of Contents

List of Figures

1. INTRODUCTION

The Internet is one of the oldest inventions of humans in the 20th century for military operations. Later, it was brought to the public for the development of technology and communication between computers. After the development of the internet and the increase in its demand in the 21st century, the World Wide Web has become the most popular thing, and people have started to use WWW services in computers to surf the internet. With the rise of popularity and benefits of the internet, companies began to use them in the office, which made the work very easy for the employees and applications for banking and other software were developed for the industrial and private sectors. Later, the internet was widely spread globally, reached the public very fast, became a powerful tool and made it accessible for the public to do private transactions in the banking sector. Applications for banking and others are rapidly developing, and the number of IT (Information Technology) jobs is improving at a high rise, but everything has a drawback and problems. The problems like theft and robbery have been there since the 18th century; at that time, it was all physical, like money stealing or items stealing. Security is a significant aspect for humans all the time, and it might be security from enemies, security for money or security for property. With the rise of digital banking in the 20th century, there has been a rise in problems with security attacks like ransomware and many malware attacks; people began to worry about the security of systems and data, which is more essential than money at the current time. Data is more crucial nowadays; it is everything for companies, it decides the future, and it helps people learn from their mistakes and progress towards their dreams.

1.1 Background

For companies, security became crucial, and they started investing in security measures like antivirus and other security tools that help them detect and prevent security breaches. The traditional network architectures are adopted from the OSI (Open System Interconnection) model, which helped to create perfect network models based on the size of the architecture. A peer-to-peer architecture is suitable if the architecture is like a small branch office with 20 – 200 systems; if the architecture is for a large company, then a client/server model is used. For better transfer of packets and to identify or validate them, Internet Protocols were introduced; the protocols like IPv4 and IPv6 were designed and assigned to the system as a unique addressing system. Scientists are predicted that there will be a massive increase in the usage of computers and internet services, so they have distinguished IP addresses into five different classes from Class A to Class E. Class A to Class C are for public use, and Class D is reserved for multicasting and Class E for Experimental or government purpose. With the architecture, we need protocols for managing the data transmission from the internet service provider to the company system; these protocols are often called firewalls, which are very important and the first line of defence for the system. Every company uses firewalls to protect the company from unwanted URLs and packets,

monitor the employees' activity, and manage special access within the company. However, due to the increase in the recent use of networking and its services in mobiles, computers, virtualisation and cloud services, there is an increase in traffic which has become a significant problem and questions the efficiency of traditional networks [5]. Due to the rise in the demand for the services provided by the internet and attempting to reach customers' needs, the hardware poses a considerable challenge for network operators to manage [18].

The dedicated services which the network providers offer is vendor locked, which implies that there is more requirement for nodes to support the customers; what means is that the vendor must buy additional equipment, space and power that are needed to meet up with the increased traffic and cope up with the customer need and demands. With this, the operator needs to reconfigure everything from the primary node to the new node, which will lead to an error-prone process, but this will lead to a security breach and data loss; therefore, networks should have a smart and easy way to manage themselves better [25]. With the rise of security problems, the industry and academic has collaborated on research that will help these human error problems to reduce and provide a more robust system/tool which allows the network to adapt to fluctuating network traffic dynamically and to keep up to its demands. The solutions like Network Function Virtualisation (NFV) and Software Defined Networks (SDN) are introduced as a product of the research. Both tools are cloud-based tools that use cloud computing technology. They promise flexibility, increased network, agility, and service-driven virtual networks that use the concepts of virtualisation and software [18].

The main feature of SDN is that it has a separate control plane and data plane, which gives network operators and system administrators an advantage. For network operators, the cost of having large data centres will be reduced and adding more nodes will also be reduced, bringing operational costs down and providing more service with less expense. As for the system administrators, the control over the network will be very high due to the centralised control plane, which is the controller. With this, the web became more stable and powerful enough to support as many extensive operations and services as the client and provider want. The centralised feature, which makes SDN more beautiful and powerful, also becomes one of the ideal targets for network attackers. Attacks like DDoS are significant; at the same time, host hijacking attacks using ARP poisoning become more prominent in the attacks by the attackers. Attackers can use port amnesia, which will force the port to reset, which the defences will not detect, and the attacker can use link fabrication to attack the SDN [1], [11], [12]. SDN is currently used in many sectors, mainly in the communication sector, to manage 5G telecommunication services to provide the best service to customers.

In the cloud, multi-tenancy is expected; the cloud operators will give their resources to the different clients and ensure that the AAA (Authentication, authorisation, and Accounting) are in place. Because when the resources are abundant, one will try to give them for lease and earn money from them, this is common in all big enterprises, and the main problem is resource pooling. The client might have enough resources as requested but might increase shortly, so he will use the entire resources to satisfy the customer's end. This might cause a dispute in the services of other clients, and if the other tenants are the client's competitors, he might try to use this chance to beat his market and company. So, the tenant should not be aware of the other competitors to minimise the risk of a security breach. The SDN will give the edge to the administrator to configure in a way that minimises the risk of identifying the other tenants in the cloud; the assignment of IP address should not be static; it should always be dynamic. Otherwise, it will give an advantage to attackers to identify the nodes in the network and identify the vulnerabilities and can choose the node and identify its place and origin.

1.1.1 Machine Learning

There is not a single person in the world who does not know the term artificial intelligence, and it is one of the advanced technologies of the 21st century which made human life easy and convenient in every household. As a part of Artificial Intelligence (AI), Machine Learning (ML) is evolved, which will help programmers and analysts to understand and predict what you want at what time; ML is a type of AI that gives computers some self-programming capabilities. Machine learning uses mathematical algorithms like naive Bayes, K -Means, Random Forest and more statistical algorithms based on their purpose and use. When a programmer/analyst builds his algorithm, he will take data as training data to predict what he wants or train the algorithm for further deep analysis. It has various applications like speech recognition, email filtering, computer vision, etc. We use supervised learning algorithms to explore the data we want for mining. One of the most powerful features of machine learning is neural networks representing the human biological brain.

Its usage has very high demand, and it is a very part of modern software all over the globe; even small industries use ML and AI to develop their product and predict the companies' profit and loss. The banking sector employs most of this analysis for business and estimation of stocks. Not only for this, but it is also used for fraud detection and preventing major security breaches; as the popularity says all about it, ML is the best and leading technology which will help the SDN to create a defence mechanism to protect the central plane from attackers. ML algorithms are based on real-world mathematical equations and solutions derived from naïve Bayes (probability using Gaussian, Bernoulli), Euclidean, Manhattan, etc. The ML algorithms are classified into four types supervised learning, unsupervised

learning, semi-supervised learning and reinforcement learning; supervised learning deals with labelled data and supervised learning must deal with unlabelled.

The supervised learning consists of algorithm models like :

1. K Nearest Neighbour
2. Decision Tree
3. Random Forest
4. Neural Networks
5. Support Vector Machine
6. Bayes theory
7. Hidden Markov model
8. Logistic Regression
9. Regression and classification

Unsupervised learning consists of algorithm models like :

1. K-means
2. Self-organising map
3. Hierarchal clustering
4. Principal component analysis
5. Independent component analysis
6. Clustering (Agglomerative, overlapping, probabilistic)

Further in this document, we will see some actual studies in SDN using machine learning and its purpose in the literature review part.

1.1.2 Mininet

Mininet is a tool for SDN to emulate networks in a virtual environment; it can emulate basic switches and routers to a complex network using an actual kernel running and application code [13]. Other tools like mininet are POX, Ryu, SDX Platform, GNS3 and NOX, which are accessible in the market. We use mininet because you can easily interact with the network using the Command Line Interface (CLI) of Mininet to customise or share it with others; we can also deploy it in real hardware [13]. One of the most incredible things about the mininet is that it will use python language to create networks, and we can edit it using the programming language as we want to add more switches or routers. It will support both python two and python 3, it uses OpenFlow protocols, and the switches are

also OpenFlow switches [26]. OpenFlow is a software developed by Open Networking Foundation (ONF) to support SDN in communicating between the control plane and the data plane; it tells the network switches where to send all the incoming and outgoing ports, and it is above the Transmission Control Protocol (TCP) and monitors the use of Transport Security Layer (TLS). We can use a mininet with a low-cost system. It does not use heavy resources, and the installation is straightforward and can quickly develop any network; it also provides a Graphical Interface to its users.

1.1.3 Floodlight Controller

It is an open-source controller developed by an open community of developers, many of whom are from Big Switch Networks. It uses OpenFlow protocols to manage the traffic flow of SDN, and the floodlight controller is responsible for maintaining all the network rules and providing the necessary instruction on handling the traffic. It is very advantageous for developers who want to rebuild or add new features by adding a few lines of code to the current code, and all the code is written in java language. It is tested in physical and virtual OpenFlow-compatible switches, works in various environments, and supports OpenStack. We will discuss how it will be used and how it supports our project in the literature review part.

1.1.4 The Architecture of SDN

SDN comprises three layers based on centralised control, which is different from the traditional network, which is a distributed control. The traditional network uses physical routers and switches, whereas, for SDN, there are Northbound and Southbound interfaces which will be used to communicate with applications and applications to controllers. The application layer is responsible for managing applications, and it can be communicated with the controller using a Northbound interface. The infrastructure layer manages the switches and nodes, which will communicate through the southbound interface, and the data will flow through the plane and not through the SDN controller.

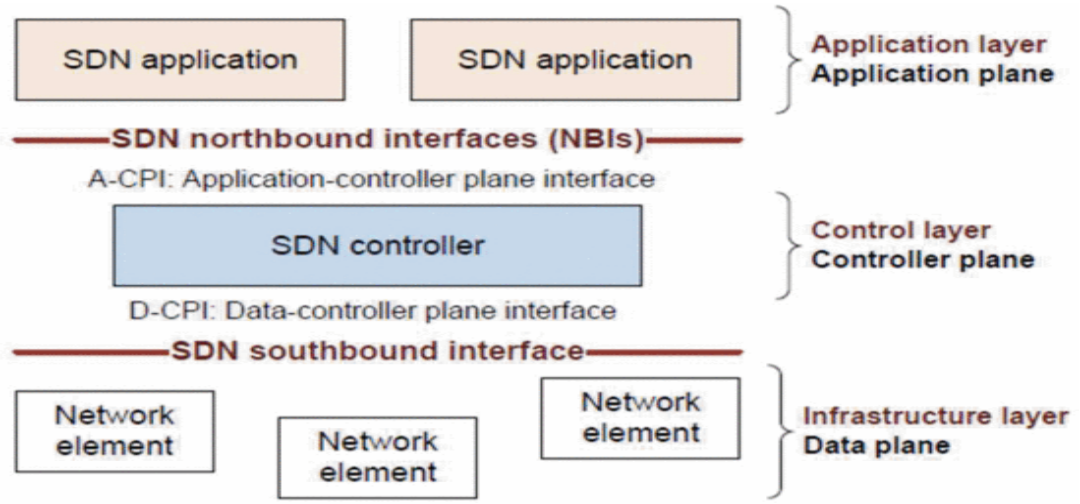


Figure-1 layer architecture of SDN (G. C. Kesavulu, 2021)

1.2 Objectives

The main objective of this dissertation is to identify the current security measures, use link latency attacks on the steps, and propose a security measure based on machine learning algorithms.

The objectives can be determined as follows:

- Use the current best security measures like TopoGuard and TopoGuard + and study them from scratch.
- Identify the difference between standard controller, TopoGuard and TopoGuard + and how it identifies the attacks.
- Identify different papers and methodologies other researchers propose and use the knowledge to identify the best solution.
- Use different types of attacks and identify if there is any security breach, then recreate the scenario and capture the packets using Wireshark.
- Use the machine learning algorithms on the captured data and train the models to identify the attack pattern to provide security for the SDN.

1.3 Summary

The chapter concludes with the introduction of the principal components of the experiment with the architecture of SDN and the main objectives of this dissertation. This chapter will not provide a deep understanding of the SDN and attacks though it will give the importance of the SDN and the security it needs.

2. Literature Review

This section of the paper will detail the technologies and studies that will support the project base and show how SDN, and Machine Learning will change the future of networking and how it provides better security to the SDN and the users. Existing solutions will protect SDN, and there are different security measures above the SDN control plane, which will manage and protect SDN from attackers. So many researchers are using honeypots as bait to attackers and gathering data from those systems to train their ML algorithms to identify the attack patterns and provide absolute security for the network [17]. The main difference between traditional networks and SDN is that the conventional network cannot handle a high volume of packet transfers/data transfers for the SDN due to decoupled control and data plane; SDN can handle a large volume of data transfer rates by blocking the packets at the switch level. The problem is not about how SDN handles the data but about security and data protection. As the world is growing and running towards the advancement of new technologies, SDN plays an essential role as a path between today and the future. We already know how advanced today's internet and mobile communications services are. Still, to withstand today's usage of internet services, we need heavy infrastructures which will support those higher data transfer rates and provide better service to the clients. To organise, manage, optimise, and maintain these traditional networks, we need ML algorithms. However, due to conventional networks' complex and distributed features, it is tough to apply and deploy on control and operator networks [17]. We use SDN for today's network to achieve those capabilities and provide better support and performance to the clients; today's new 5G networks use SDN as a central part of the infrastructure and offer higher transfer rates and can sustain a higher volume of packet transfer. If there is a security breach in SDN at the main level, all the information and data will be hijacked by the attacker, which is a significant problem for the service providers; to overcome it, we need security measures and resilient protocols which will help the SDN, ML will allow us to overcome this obstacle.

2.1 Related Work

Soltani (2021) proposed a new security measure for TopoGuard + using ML to provide better security measures; they have proved that the Link Latency Attack can bypass TopoGuard + security and attackers can use host hijacking on the hosts in the SDN to gain control on SDN [1]. There are three main parts to creating the system that detects incoming attacks. The first one is the creation of a dataset by collecting data from the previous attacks on SDN. The second step is to use ML classification algorithms like Logistic Regression, Support Vector Machine, K-Nearest Neighbours, Naïve Bayes, Random Forest and Multi-Layer Perceptron to classify the regular packets and packets sent by the attacker. The third step is the implantation of the MLLG algorithm on the controller, which works with

an accuracy of 96%, which is better than the other security measures like TopoGuard and TopoGuard + [1].

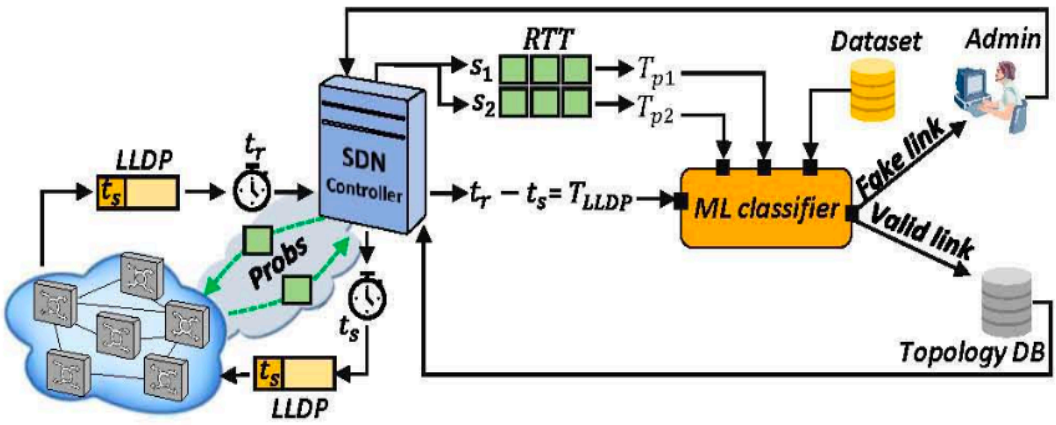


Figure -2 The architecture of the MLLG system (Soltani, 2021)

This system can support the controller in real time, and figure 1 is the architecture of the MLLG system to protect the system from LLA proposed by the author. The system's purpose is to identify the LLA attack due to the limitations and lack of authenticity of TopoGuard and TopoGuard +. The TopoGuard + will check for LLDP processing time (i.e., creation and transfer of LLDP), the attacker will use brute force to increase the load of the network using bots to launch a DDoS attack, and this will increase the default time in LLDP processing which will give advantage to the attacker. Then he will fabricate a new link between the switches in the payload time and stop the attack. The network will try to update the topology by sending LLDP packets to all the hosts and switches in the network. This process of checking the authenticity is called Link Latency Inspector (LLI). However, the LLI could not identify the difference between the LLDP because the Round-Trip Time has increased, and the formula to calculate the LLDP will show this is a valid time. So, the Machine Learning based authentication defence system was introduced here.

Karmakar (2019) has proposed a method for researchers to create their dataset and use ML to classify the threats and attacks that will happen on SDN by launching different types of attacks like a botnet, DoS, DDoS, Password Brute-Forcing, probe, exploitation SQL Injection and XSS on the Northbound API (which resides between the controller and Application Layer) which is the most vulnerable. Once the attack is made on the system, the packets are captured using Wireshark, exported as a CSV file, and used on ML algorithms for classification. The algorithms used here are single decision tree, Random Forest, Adaptive Boosting, K-nearest neighbour classifier, Naïve Bayes, Support vector machines and multi-layer perceptron are used here [3].

Krishnan, P (2019) has proposed a Network Security and Intrusion Detection System (NIDS) called VARMAN: adVanced multiplAne secuRity fraMework for softwAre defined Networks. This security scheme consists of coarse-grained flow monitoring algorithms on the data plane for instant anomaly detection and predicting DDoS/botnet attacks, along with the NIDS system hybrid deep learning-based classifier pipeline added on the control plane. Adding existing ML-based classifiers to the NIDS has cost a higher processing power and memory requirement, which is a setback. A hybrid model combining deep and shallow learning methods is implemented to address this problem. This system has an accuracy of 98% in detecting real-time attacks [7].

R. U. Rasool (2019) has worked on the Link Flooding Attack where the attacker employs bots to surreptitiously sends a low rate of legitimate traffic to the control channel, which will cause the control plane to disconnect from the data plane, to address this problem and provide better security CyberPulse was created which used ML classifiers to classify the network traffic using deep learning techniques and is implemented on Floodlight controller. The ML algorithms used here are Two class Decision Forest, Two-Class Boosted Decision Tree, Two-Class Decision Jungle, and Two-Class locally Deep SVM, which have the best accuracy and are very fast to train. The attacks are considered into two categories flooding flows, and another one is legitimate flows; the precision is 75%, the accuracy is more than 85%, and the overall accuracy is 76% [8].

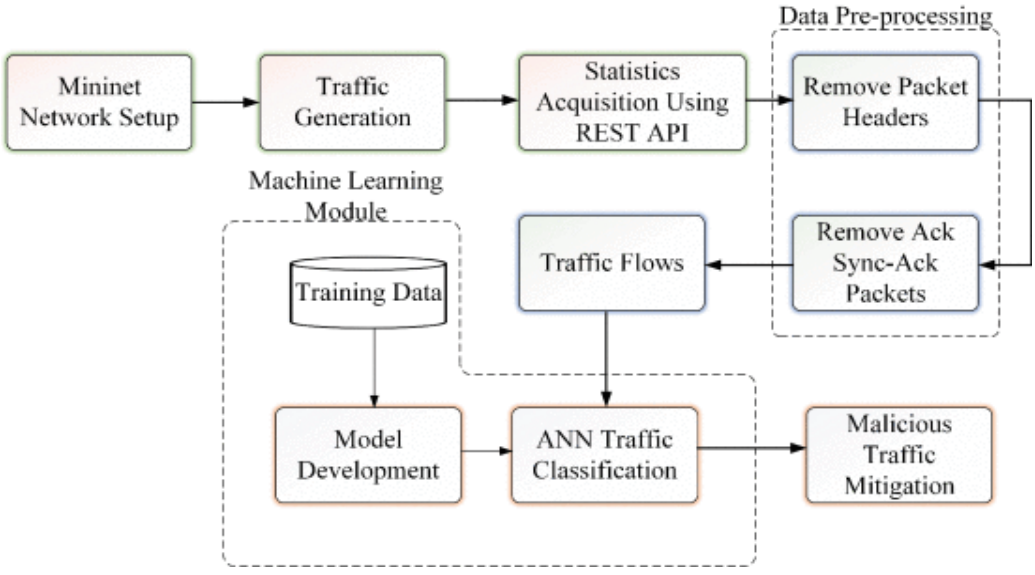


Figure-3 Flow diagram of CyberPulse operation (R. U. Rasool, 2019)

S. Nanda (2016) has used honeypots on SDN to train the SDN to identify attacks and attack patterns using ML algorithms by training the algorithm with the Historical Attack data and using real-

time network data. Historical data can be used to automatically identify and block malicious connections, which will help in the defence of SDN. Here the author used 32 honeypots with data from 17M login attempts originating from 112 different countries over 6000 distinct IP addresses. Here ML algorithms like C4.5, Bayesian Network, Decision Table, and Naïve Bayes are used to predict the host that will be attacked using historical data. An average of 91.68% was the accuracy achieved by the Bayesian Networks, which shows the prediction rate was high and better than the remaining algorithms [16].

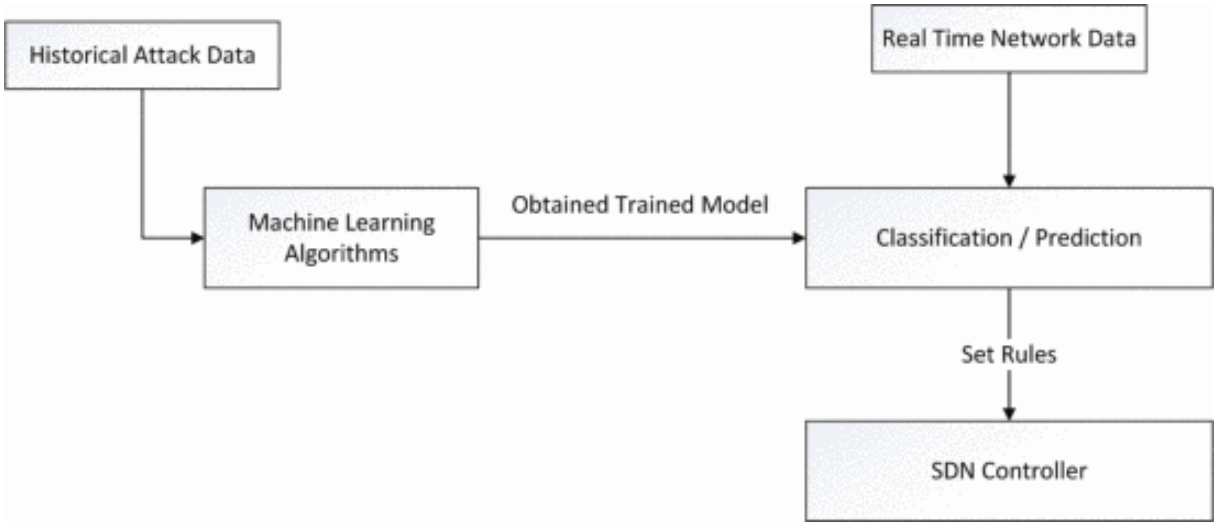


Figure-4 Machine Learning based architecture for defining security rules on SDN controller (S.Nanda,2016)

J.Xie (2019) have proposed how ML algorithms can change the future of SDN from traffic classification, routing optimisation, quality of service, prediction of attacks, resource management and security. The author introduced a dynamic routing architecture which consists of an ML-based meta layer with a heuristic algorithm layer. The ML-based meta layer will solve the problem of dynamic routing and can obtain real-time heuristic values as output, and this model is used to train neural networks. The ML-based meta layer is for the supervised learning-based routing optimisation; by training the supervised model, we can achieve the solution for real-time routing [17].

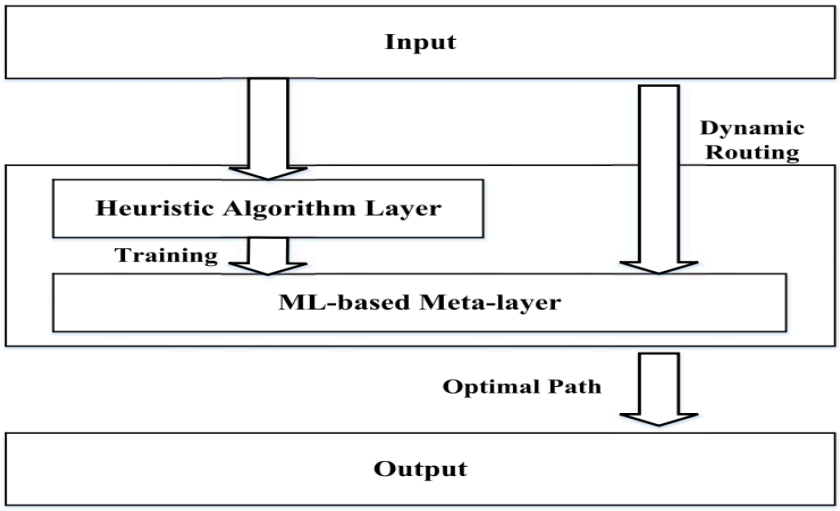


Figure-5 Dynamic routing architecture (J.Xie,2019)

A logically centralised cognitive routing engine is used here for an SDN-enabled inter-data centre overlay network. Neural Networks and Reinforcement Learning techniques are used to find optimal overlay paths between geographically dispersed data centres. It can also work well in chaotic environments [17].

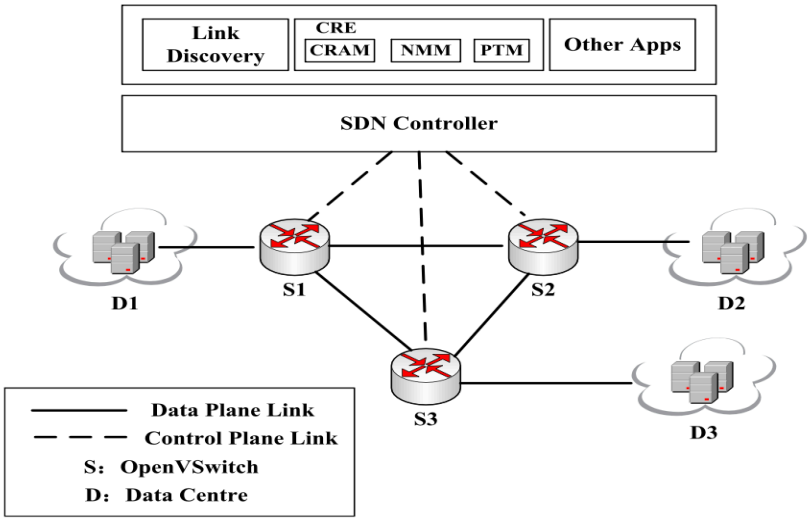


Figure-6 SDN-enabled overlay network with CRE (j.Xie,2019)

The current population is multiplying, and the usage of devices and IoT is also increasing, but the resources in the data plane are limited. To maintain the flow and the QoS, we need admission control

which will manage many service requests. We can program which requests can be accepted and rejected by our resources using ML algorithms [17].

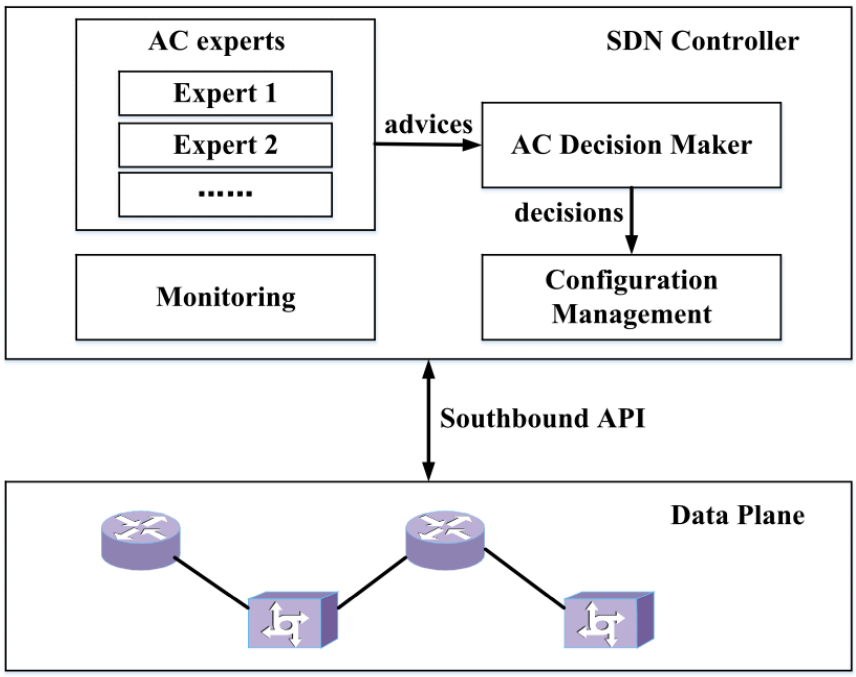


Figure-7 An ML-based Admission Control for SDN controller (j.Xie,2019)

The author has used both supervised learning and unsupervised learning algorithms. He also used SDN's semi-supervised learning and reinforcement learning algorithms and proposed many more solutions that will help in different sectors.

J.A.Pérez-Díaz (2020) proposed a system that will detect attacks on SDN, the system which was developed will be in a separate cloud/system and can only be accessed through API. The primary purpose of this Intrusion detection system is to detect Low-rate DDoS attacks, which are difficult to detect by SDN, the modular architecture designed by the author will use ML algorithms to train the Intrusion detection system. The algorithms used by the author are random tree, REP tree, Random Forest, Multi-Layer perceptron and support vector machine. The ONOS (open network operating system will be used here as a controller, and the mininet will be used for simulating the environment in a virtual machine[20].

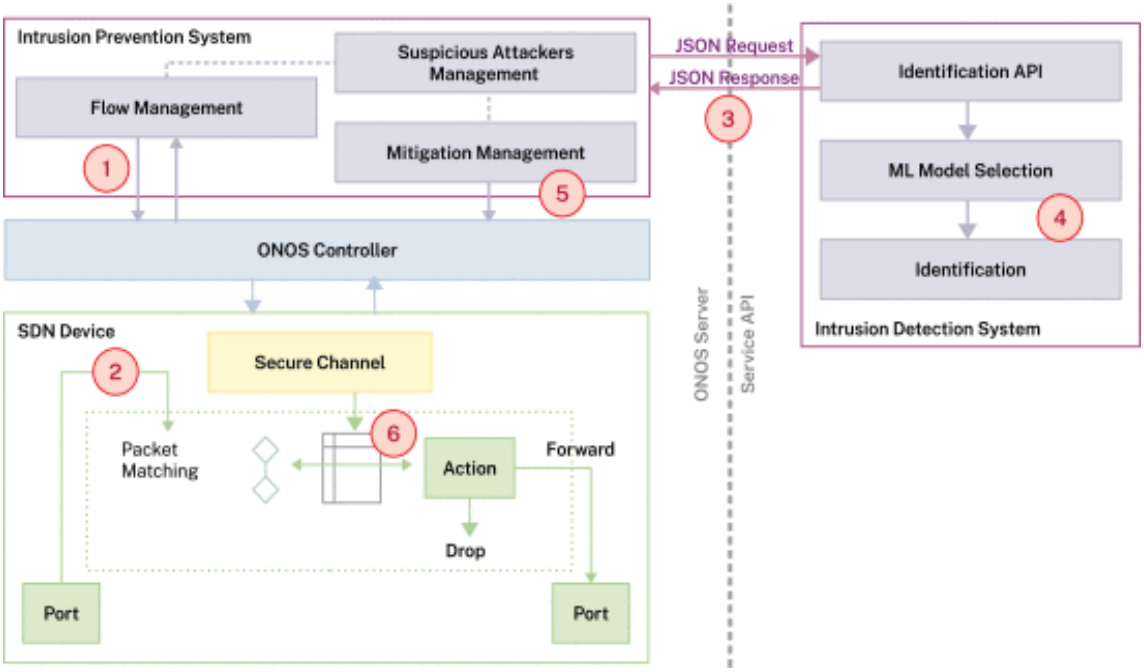


Figure-8 Framework of Intrusion Prevention System and Intrusion detection system (J.A.Pérez-Díaz,2020)

The Intrusion prevention system will have a flow management module responsible for detecting HTTP flows, the suspicious attacker's management is responsible for managing a backlist of potential attackers, and the mitigation management is responsible for generating flow rules which will manage malicious data. This model design is one of the best models, which will give a better edge for reusing the same API for different SDN, and the system performs excellently with an accuracy rate of 95% [20].

A.O.Sangodoyin (2021) proposed using ML algorithms to detect DDoS attacks in SDN by using ML algorithms on SDN. He proposed a model that can be used for basic SDN models and small infrastructure usage. The model is another example of how ML can solve the SDN security problem and prevent DDoS attacks. Here the ML algorithms used are quadratic discriminant analysis, Gaussian Naïve Bayes, k-nearest neighbour, and classification and regression tree (CART). The CART has given the best accuracy of 98% in this experiment with a prediction speed of 5.3×10^5 observations per second [21].

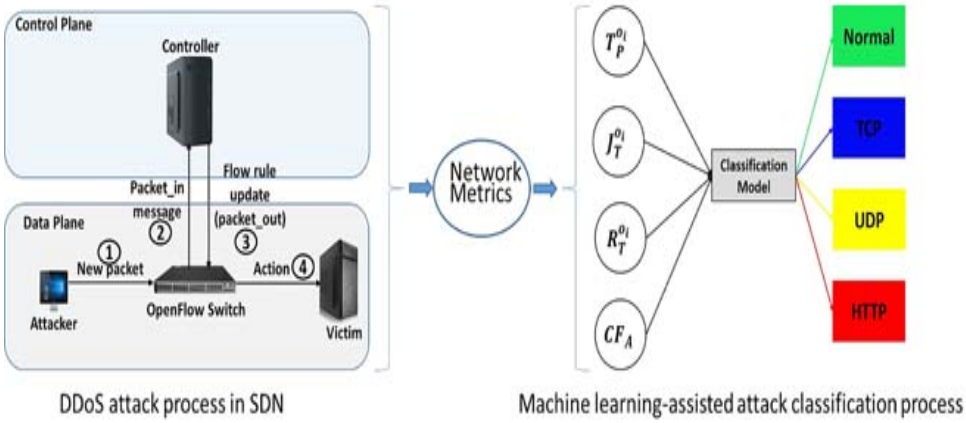


Figure-9 architecture for ML-based detection and classification of DDoS attacks (A.O.Sangodoyin,2021)

Satheesh (2020) has proposed a flow-based ML model with SDN, which acts as an intelligent system to limit the network's traffic flow using reserved bandwidth and make use of extra bandwidth. The model was compared with the schemes available at the network to produce better outcomes with fast routing and the security of SDN architecture to identify the gaps open within it. This model is more of a priority-based model for SDN for controlling the packets. The Intrusion Detection system uses ML algorithms to monitor the traffic flow for low-rate detection attacks, which are nearly impossible to detect and DDoS attacks using bots [23].

2.2 Attacks on SDN:

SDN has two types of interfaces. Namely, they are the Northbound interface and Southbound interface, and Northbound is for applications programming interface (API) or a lower level of the component, which will allow communication with higher level or central components. The southbound is vice-versa northbound allows higher-level components to communicate with lower-level components. These two interfaces can be called the application plane, which supports programmers and developers in deploying their applications to commute with the central plane. The operator software will not directly issue a command to the nodes; instead, the operator uses the application layer to issue commands over the control layer of the northbound interface. The applications can be for networking, security, company management, or storage server deployment. It can be anything due to the application's programming

errors/software bugs, and illegal function calls will give a chance to attackers. To launch their attacks and compromise the network, it is a better chance due to the permissions given to the third-party applications logic. The southbound interface communicates between the central control layer and the lower elements in the data layer; the controller sends commands and configurations to the network devices over the southbound interface. The attacks in SDN are mainly due to the errors in the applications deployed over the application layer, and the Northbound interface is very vulnerable. Most attacks happen at the northbound interface.

Karmakar, K.K (2019) has proposed a technique to deal with security breaches in the northbound interface and used the ONOS controller to implement the new security policies in the data plane. The security application developed will have the specifications and security policies stored in it to make decisions for different types of attacks. This policy-based model will use user-written policies for SDN precisely to control any SDN infrastructure for enforcing these policies; the author has taken a network application-based approach. The security application has two policies: Secure Routing Policy Expression (SPE), which will control the routing behaviour, and Intrusion Detection Policy Expression (IDPE), which will detect and block malicious attacks in the SDN network. This system is efficient for Spoofing attacks, Flooding attacks and signature-based attacks [2].

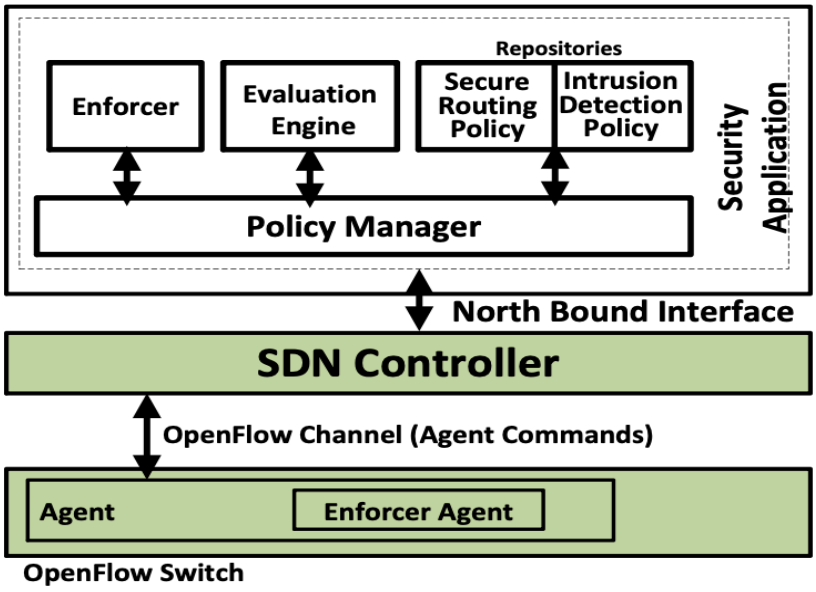


Figure-10 security application architecture for SDN (Karmakar, K.K, 2019)

G. C. Kesavulu (2021) has proposed a prevention mechanism against DDoS attacks on SDN, as he states that there is no perfect security software for SDN, and prevention is better than cure. For

creating a countermeasure to defend SDN from DDoS attacks, he used two filtering techniques, ingress and egress; ingress means incoming traffic and egress means outgoing traffic. The author used mitigation techniques to deal with flooding-based DDoS attacks, which can be achieved using different filters and rate-limiting methods on the incoming traffic. The mitigation framework aims to drop out the incoming attack traffic as much as possible and keep the legitimate traffic intact. The mitigation framework requires communication between the defence system, detection and traceback [4].

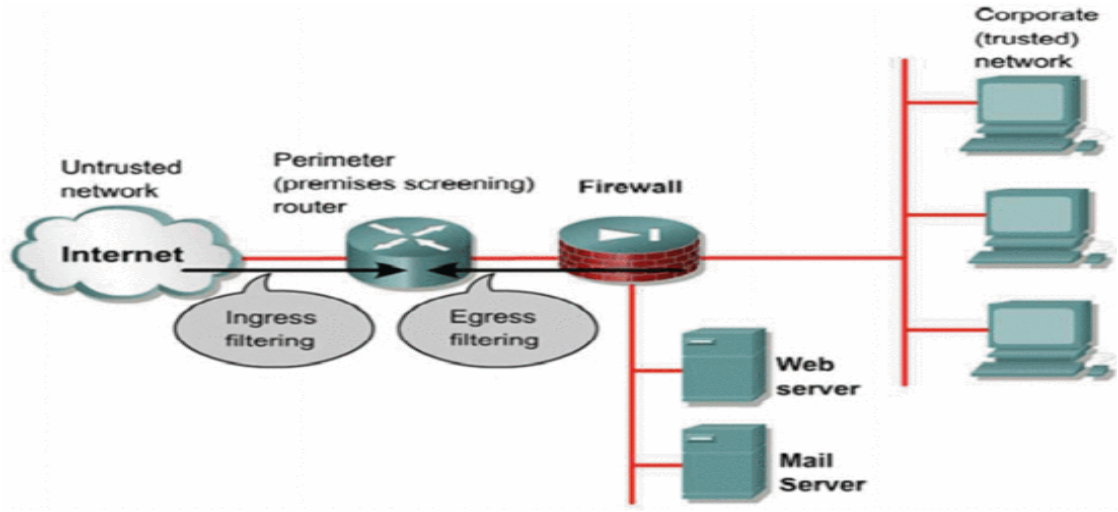


Figure-11 prevention of DDoS attacks using ingress/egress filtering (G. C. Kesavulu, 2021)

S. Khan (2017) has worked on topology discovery in SDN, and problems caused in SDN due to frequent migration of the virtual machines in data centres, lack of authentication mechanism, scarcity of SDN standards and integration of security measures for topology discovery. Due to these problems, topology discovery has become very difficult in vast networks. Due to this problem, the relevant threats are also increasing for different layers of the SDN architecture. The attacks on SDN Data Plane can be Malicious Switches. Malicious hosts in the data plane, attacks on the Control plane can be Malicious modules inside the controller, compromised controllers and Attacks on management consoles. Attacks on the Application plane can be Unauthorized access to applications, Disclosure of information through the application server and Modification of user privileges for application execution, attacks on Interfaces can be Attacks on Southbound, Attacks on Northbound and Attacks on eastbound and westbound [6].

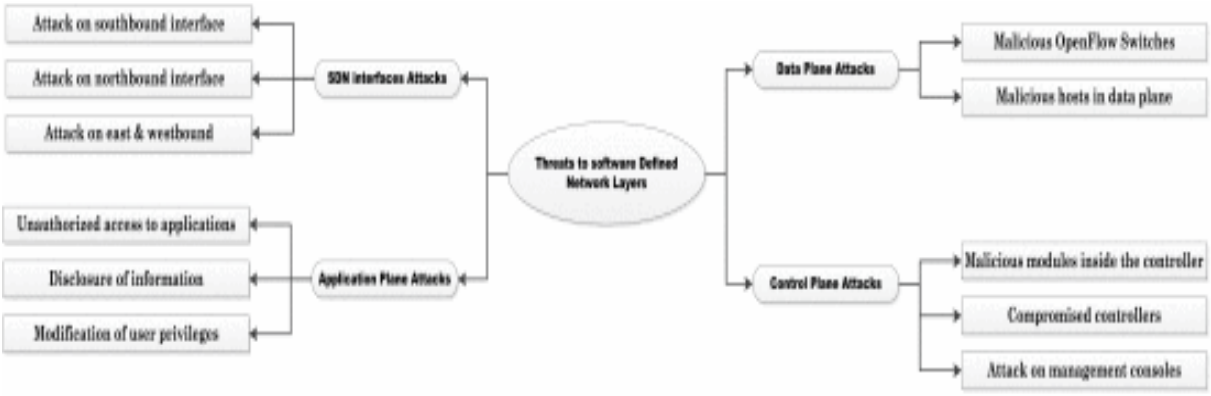


Figure-12 Classification of attacks on SDN (S.Khan,2017)

The below figure-13 gives insight into the methods for Topology Discovery threats and current solutions available to protect SDN from the threats.

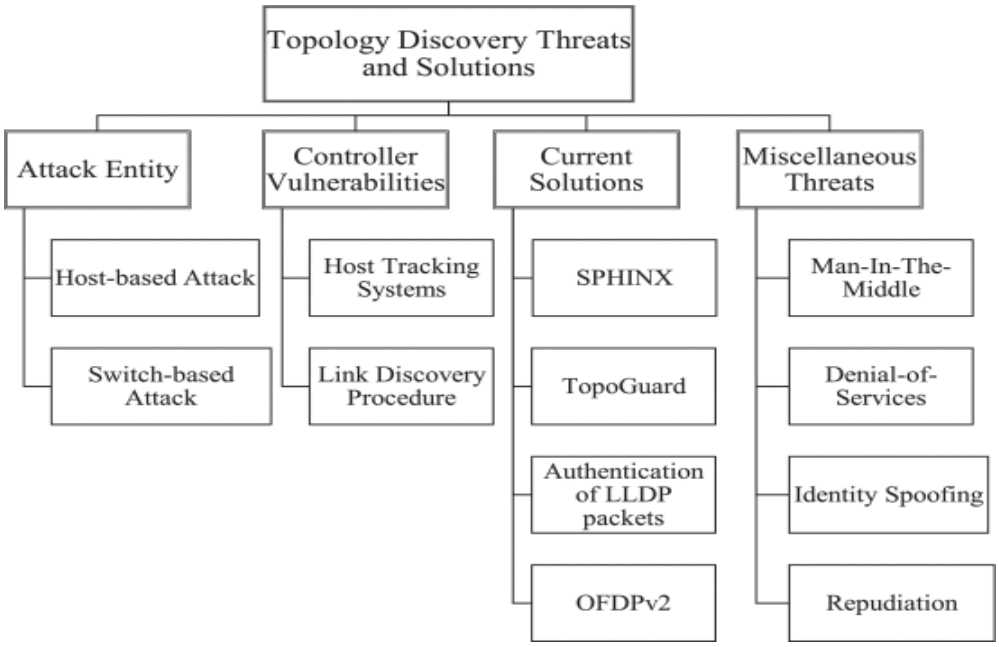


Figure-13 A thematic taxonomy of topology discovery in SDN (S.Khan, 2017)

H. Z. Jahromi (2018) has addressed the challenges in SDN while introducing an application-aware network framework using an arbitrary performance metric and information from ML on the network. The information feedback is taken from the Northbound interface, which will receive during runtime. These metrics will help the network manager understand the effect of his network decisions on the application [9].

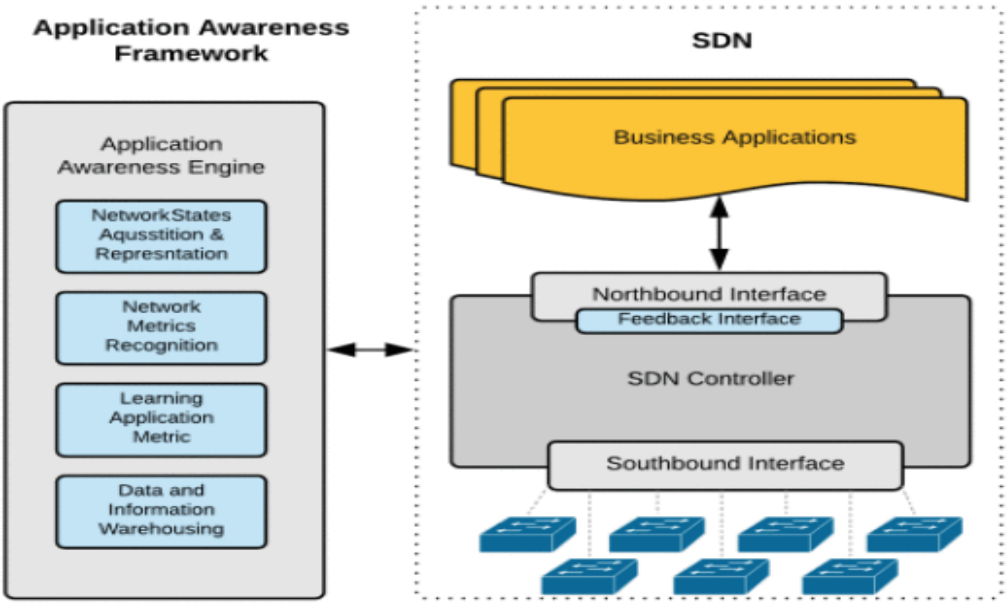


Figure-14 Application Awareness Framework (H.Z.Jahromi, 2018)

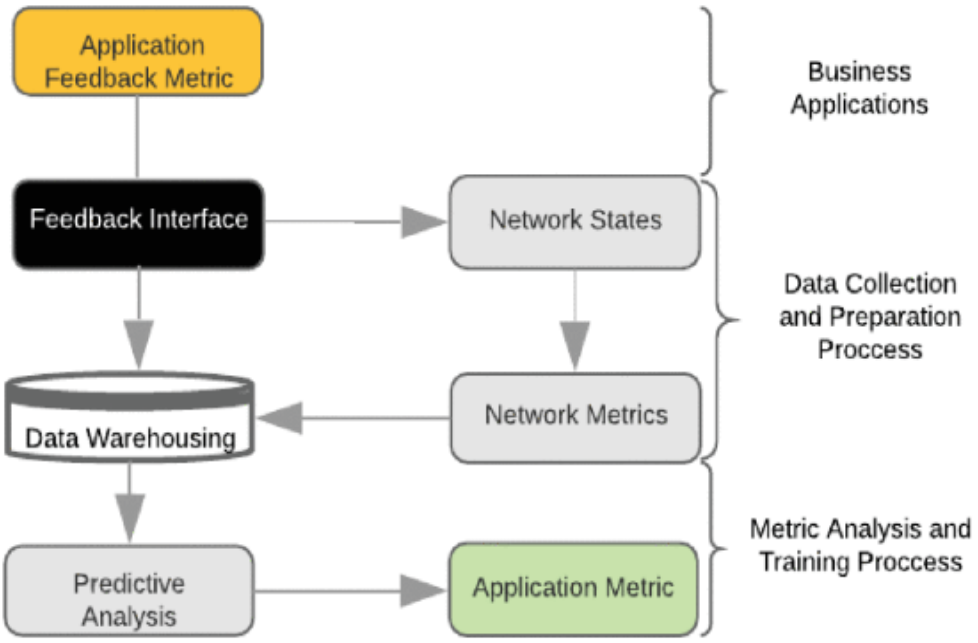


Figure-15 Application Awareness Engine (H.Z.Jahromi, 2018)

The data will be collected whenever a new connection is made with the controller and stored in the database; the data collection and preparation creates a structure for collecting and storing network and application information. The Application Awareness Engine and ML will use the Feedback interface to collect the network metrics and perform predictive analysis, which will be stored in application performance metrics [9].

R.Skowyra (2018) has proposed new features for TopoGuard + to improve its defence. It will add extra security for its predecessor (TopoGuard), which failed to prevent security attacks due to liabilities in the defence mechanism. The attacker will use port amnesia attacks (depending on link fabrication using out-of-band communication channel) and port probing attacks to bypass the security of TopoGuard; the port amnesia attack scenario will be like the attacker will connect TopoGuard network through wireless, then the attacker waits until the controller sends an LLDP packet to the switch and uses the LLDP packet and sent it to the host connected to switch, later the attacker uses port amnesia attack on the host and initiates the port down events. The TopoGuard resets the port to ANY, and the attacker will connect to the switch and bring the port back up. The port probing attack scenario will be like the attacker joining the TopoGuard network with his own Ip and MAC address and sending an ARP ping to the host and getting his Ip and MAC. He will wait for the host to go offline and fabricate his Ip and MAC with the hosts Ip and MAC, then sends an HTTP or ICMP message to make the TopoGuard think the host joined in a new port; the TopoGuard + has created a mechanism called Link Latency Inspector to evaluate the LLDP packets because the LLDP will be different from one host to another[11].

S. Hong (2015) has proposed the defence mechanism TopoGuard against attacks like Host Hijacking Attacks and Link Fabrication attacks. The defence became very popular in the networking field for SDN, and it uses port status (up and down) and LLDP packet verification as its primary strategy for defence mechanisms. The attacker will hijack and impersonate the host to gather data from the network and compromise the entire network. The hosts in the network cannot differentiate between the original server and the attacker server. In this attack, the attacker injects fake LLDP packets into the server, saying a new connection between the switches. The server thinks it is a new valid connection; it sends its original LLDP packets to the attacker. From the LLDP packets, the attacker will get the network information, size, and syntax of LLDP, which are the SDN's most vulnerable features. The attacker can use ARP poisoning to get the credentials and information on the switches and hosts, so the security measures will check to validate the topology whenever it gets updated, which depends on the information from the port manager and host tracker [12].

N. M. Abd Elazim (2018) has worked on how the traditional network differs from SDN, provided challenges that SDN currently faces, and later discussed the security challenges and their countermeasures. The author has discussed all the available countermeasures and attacks in the SDN, the attacks he mainly discussed will be like port amnesia and port stealing in SDN; he has done vast research on Link Latency attacks and Host Hijacking attacks, and he has classified attacks for SDN for each layer, and how the layer works and what is its vulnerability then he discussed the countermeasure.

It is like awareness research for people who want to work on SDN. Their research is based on the security of SDN and the use of ML [15].

U. Tupakula (2020) has worked on SDN switches and attacks on SDN; the author has developed a Switch Security Application (SSA) for SDN controllers, which will use trusted computing technology and additional components for detecting attacks on switches. The TPM attestation is used here to ensure that the switches are in a trusted state during boot time before configuring them. Flow rules will be attached to the switches and stored in the additional components; if there is any change in the flow of rules in the switches, then that switch will be considered under attack [22].

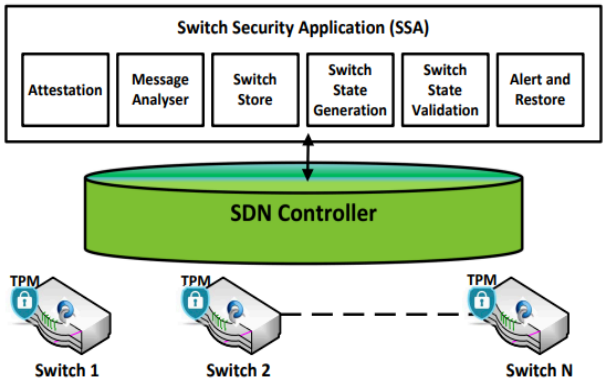


Figure-16 Logical architecture of switch security (U. Tupakula,2020)

2.3 Summary

This chapter concludes with the research on the SDN and its attacks and how ML is being used to improve its performance and agility. There is still research and different papers in IEEE Xplore, and other papers and documentation are available. However, the main agenda of this thesis and research is to show SDN's advantages and the tools discovered for SDN, simultaneously, how difficult it is to protect it from invaders. The review has proved to us that so many researchers are trying to implement ML on the SDN for various reasons, and only a few have developed a real-time solution that has a success rate.

3. Requirements and Analysis

This dissertation aims to learn and understand SDN and its security issues, implement the link latency attack, and use ML algorithms to create a defence for the SDN from the base research paper by Soltani [1]. The primary research is based on the TopoGuard, and TopoGuard + research papers, how these defence mechanisms work against the LLDP and ARP attacks, and how the design works against Link Latency attacks as the TopoGuard + is an upgrade for its predecessor, which could not detect port amnesia and port probing attacks. However, the TopoGuard + has a backlash in its security system, the Link Latency Attack, which will add a fake link between switches through an out-of-band channel; the malicious host will use an ARP poisoning attack by sending unwanted traffic to the switches, which will increase the processing time of the switch [1]. To protect SDN from this kind of attack, we use ML algorithms to identify and predict the attacker packets, and an ML-based guard will be created for the system by taking the example from the base research. The implantation and testing will be discussed in further sessions, giving a clear conscience about the experiment, and results will be discussed.

The main requirements of the system are :

- i5 or i7 intel processor or AMD Ryzen 5 or Ryzen 7 processor
- 16Gb RAM
- 500 GB Hard Disk
- 2GB Graphic Card
- Windows or Ubuntu Operating System 64-bit

The main components of this experimental research are

- Eclipse IDE
- Mininet VM or an Installation file for Ubuntu
- VM Ware Workstation
- Ettercap Tool
- Wireshark

Moreover, python 3.6, HTML and java 7 are required for the programming end, and to run python, we need jupyter notebook or use colab by google online.

The dissertation is divided into several tasks to achieve the desired results; the tasks follow as follows:

Task -1:

Learning about SDN and the number of tools there to support SDN and different types of controllers

Time required: 2-3 weeks

Difficulty: Easy

Importance: High

Prerequisites: Must have basic knowledge of networking and network attack scenarios.

Task -2:

Research on TopoGuard and TopoGuard +, then find related research papers on SDN with machine learning and attacks on SDN.

Difficulty: Medium-Hard

Importance: High

Time required: 4-5 weeks

Task -3:

Practical deployment of mininet and floodlight controller and learning commands and their usage.

Time required: 1-2 weeks

Difficulty: Medium

Importance: High

Task -4:

Learn python script for mininet topology deployment and write HTML pages for the host and attacker scenario.

Time required: less than one week.

Difficulty: Easy

Importance: Medium

Task -5:

Launch the attacks on basic floodlight controllers in VM from traditional attacking methods to using attack tools.

Time required: 1-2 weeks

Difficulty: Medium

Importance: Moderate

Task -6:

Launch the attack on TopoGuard and try to understand the defence mechanism and break its defence mechanism.

Time required: 1 week.

Difficulty: Medium

Importance: Moderate

Task -7:

Launch the attack on TopoGuard + and try to understand the defence mechanism and break its defence mechanism.

Time required: 1 week.

Difficulty: Hard

Importance: High

Task -8:

Collect the data from the attack scenario and start building the ML model.

Time required: 2-3 weeks.

Difficulty: Hard

Importance: High

Task -9:

Prepare AUC and ROC curves for the model and train the model with the second dataset.

Time required: 1 week.

Difficulty: Medium

Importance: High

Task -10:

Write the abstract and introduction with the main page, declaration and acknowledgements.

Time required: less than one week

Difficulty: medium

Importance: High

Task -11:

Writing Literature review from the research papers and findings.

Time required: less than 1 week

Difficulty: medium

Importance: High

Task -12:

Write the remaining sections, like the dataset and requirements.

Time required: 2 days.

Difficulty: Easy

Importance: Moderate

Task -13:

Write the Implementation and testing part, explaining the practical experiment on breaking the defence of TopoGuard +.

Time required: 1-2 days

Difficulty: medium

Importance: Moderate

Task -14:

Discuss the results and conclude the project.

Time required: 1-2 days

Difficulty: medium

Importance: High

These are the tasks, and we need to learn some basic commands for mininet and ubuntu terminals; the commands are:

Mininet:

Sudo means superuser do or substitute user do

mn means mininet

‘sudo mn -h’ help command

‘sudo mn’ command which runs a basic topology with 2 hosts, 1 primary switch and 1 controller.

‘sudo mn -c’ for cleaning and resetting the ports in the current environment or the entire system environment and deleting the host or IP addresses that are connected to that port.

‘sudo mn --topo single,3’ is for the single switch and 3 hosts to create and run in the current environment or terminal.

‘sudo mn --topo linear, 4’ is to create two switches and two hosts for each switch and run in the environment.

‘sudo mn --custom -/mininet/custom/topo.py --topo mytopo’ for running custom topology

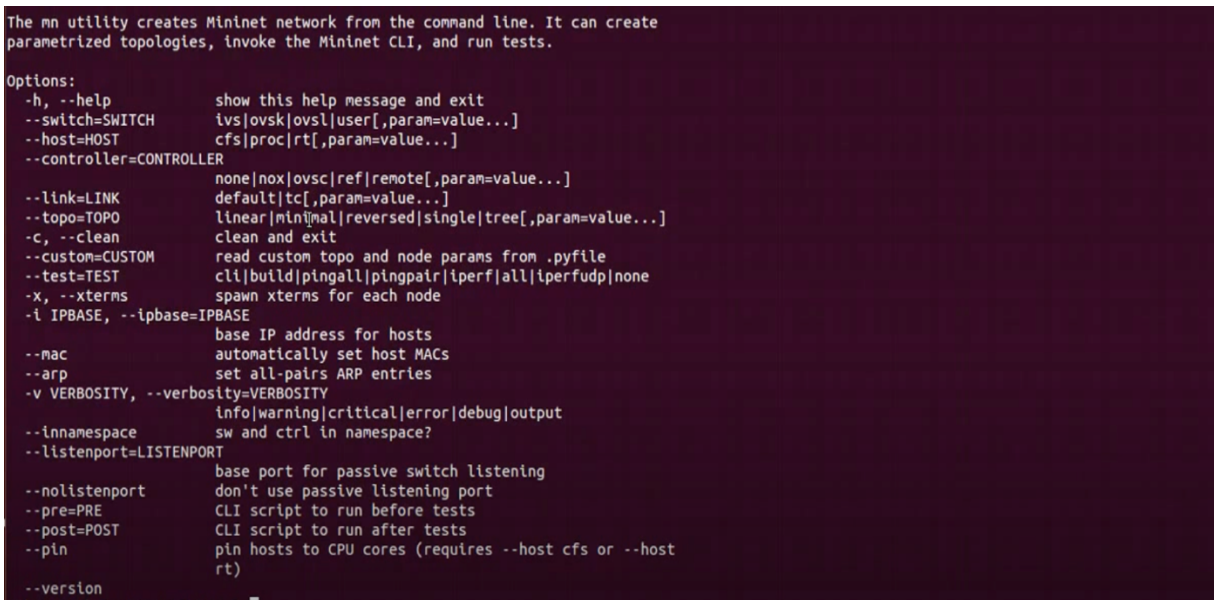


Figure -17 mininet help and different command options.

In Mininet commands:

‘dump’ command dumps information about all nodes

‘h1 ifconfig -a’ Ip information of host 1

‘s1 ifconfig -a’ switch interface details

‘h1 ping -c 1 h2’ pings one time from host 1 to host 2

‘pingall’ pings all the hosts one time in the network

‘xterm h’1 opens Xterm terminal for host 1

In xterm:

‘python -m SimpleHTTPServer 80’ runs a simple HTTP server in host xterm
‘chromium-browser –-no-sandbox’ to open
‘wget Ip address of host -o and Ip address of server’ to send a hello message to a server
‘ettercap -G’ opens an Ettercap graphical interface

The command for creating a mininet topology and connecting to the floodlight controller:

sudo mn --controller=remote,ip=<**controller** ip>,port=6653 --switch ovsk,protocols=OpenFlow13

3.1. Dataset

The data set is created by deploying the topology in mininet and using the Floodlight controller; the TopoGuard and TopoGuard + will be launched in the Floodlight controller to launch the attacks and collect the packets data using Wireshark, and the entire scenario is launched in the Ubuntu/Unix system. However, due to fewer features from the collected data, we use the available dataset to train the algorithm, which will predict the attack pattern from the real-time data.

The available/known data set consists of features like the IP address of the source and sender, port number, switch number, duration, total duration, flows, packet details (inflow, byte per flow, rate etc.), protocol, packet transfer speed and the last column the label(Attack packets (1), regular packets(0)). The dataset consists of 104,346 values packets of data, and the data set size is 12.6 MB; a medium dataset in terms of ML language, the data set is cleaned and removed from unwanted data. The dataset is in CSV format [27].

The data collected in real-time has a size of 15.6 MB with limited features; the data has vital information like the protocols and size of packets, which is primary for validating the packets transmitting in the live network. Hence the model is already trained with the known data set; therefore, it will be easy to predict the attacks since it is an extra defence for TopoGuard +, which has its own defence mechanism, i.e.., Link Latency Inspector; we have two datasets with us now for the model.

4. Implementation and Testing

The implementation will start with choosing the operating system, and Ubuntu is best for this kind of the project; MAC OS cannot support now with the restrictions, and it needs support for running the system without any security issues. Windows do not have the community and developers for the testing and attacking software, but windows can support virtualisation; we can launch multiple VMs in windows and experiment. VM is best for this kind of security experiment. It protects the system from viruses and malware that might spread across the system while we are performing the experiment or using unprotected tools for attacking. With the help of virtualisation, we can prevent the primary system from this kind of attack and perform our experiment in a virtual environment without any concern for data loss and security issues. I have chosen Windows 10 Home single OS with VM ware workstation free version to load ubuntu 17 or 18, which supports some old libraries which are not available for new versions of Ubuntu. People can download a mininet VM from the official website (http://mininet.org/) or install it from the terminal in Ubuntu, and VM software can be downloaded from the official VM Ware website (https://www.vmware.com/in.html).

Once we download the ubuntu OS from the official ubuntu website (https://ubuntu.com/), we can load the OS in the VM ware and allocate the ram, the number of cores, and memory we want for the experiment; I have eight cores in my CPU which is an i7 processor, and I have 16Gb of pre-installed ram, I have chosen 3 cores, 6GB of ram and 100Gb of disk space for the VM. After installing the ubuntu in the VM, we need to update all the libraries and install git, python3, java 7, Ettercap and mininet and their supporting libraries (gtk3, OpenSSL, libcap, CMake, zlib, glib, curl etc.). If you have trouble installing any software or terminal installation errors, you can find help on the ubuntu community website (https://help.ubuntu.com/community/CommunityHelpWiki) or related documentation or videos on the internet. Once everything is installed, you can install the floodlight controller using the git command in the terminal. Download TopoGuard and TopoGuard + from the git hub repository (links for TopoGuard: https://github.com/xuraylei/floodlight_with_topoguard, and TopoGuard +: https://github.com/xuraylei/TopoGuard_Plus) using git pull command. For floodlight installation, you need libraries like Ant and Maven to build the code to run according to the java version and build-essentials, which will support tun in the ubuntu environment (note: some of the build libraries will differ from the flavour of Linux), python development package and eclipse IDE Luna version is preferred you can find related documentation for installation in the website.

Extract the downloaded TopoGuard/TopoGuard + into a folder, open a terminal in the specific folder, and build the code using Ant/Maven; after a successful build, you can run the controller from the terminal or eclipse. However, it is a defence mechanism developed for SDN, so I will run it in eclipse

IDE by importing it as an existing project into the current workspace, then clicking run from the above options in the toolbar and selecting the run configurations; it will pop up a new window, in the window right click java application and select new and it will give three empty boxes fill them with name box as FloodlightLaunch and project box as Floodlight and main box with the net.floodlightcontroller.core.Main and click apply; you can find it in figure 17.

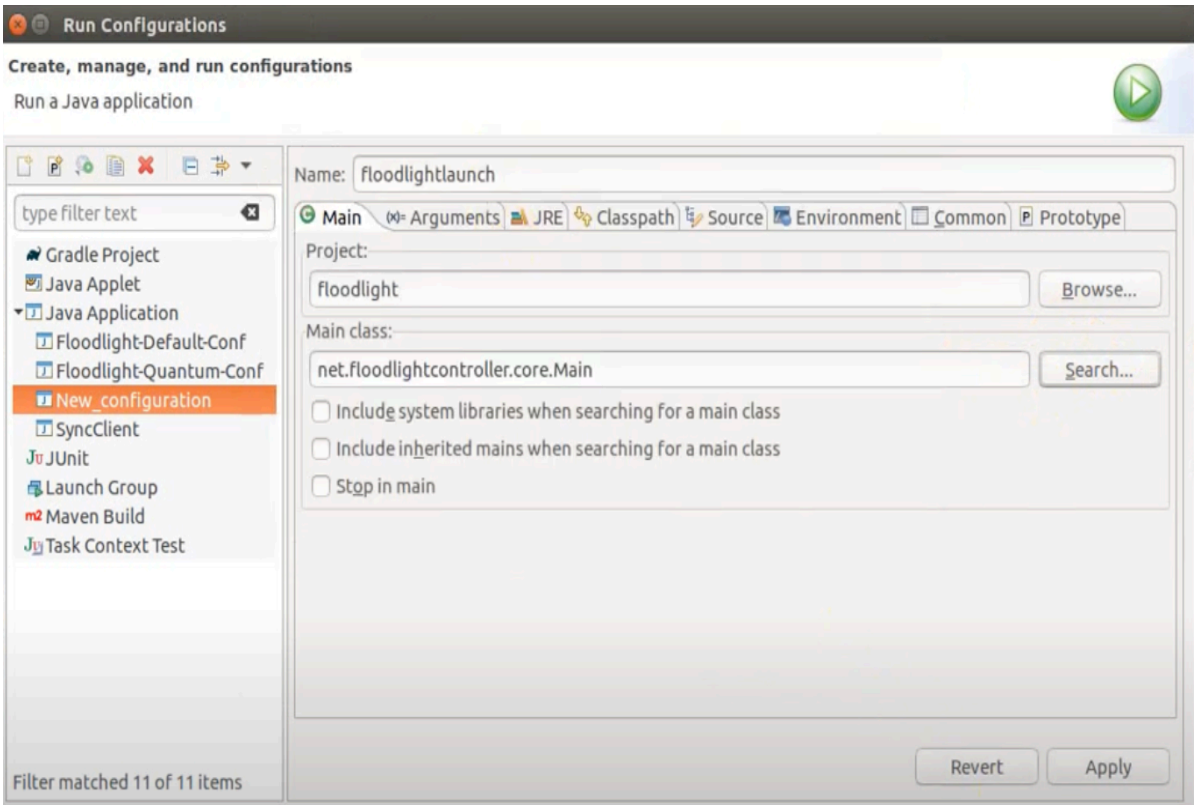


Figure-18 Run Configurations in Eclipse

Now the TopoGuard is running with port listing on 0.0.0.0/0.0.0.0:6633; you can find it in figure 19, and we can create our topology with the number of switches and hosts. I have created a custom topology using the python language; the topology consists of 6 hosts and 3 switches, and the controller Ip address is the Ip address of the system that the controller is running, i.e., 192.168.149.138. When the topology is created from the terminal, the links between hosts and switches will be automatically created; when you write a custom topology, you need to add the links manually between switches and hosts and add a link between the switch and controller to start the topology.

```
15:50:26.439 INFO [n.f.l.i.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
15:50:26.534 INFO [o.s.s.i.c.FallbackCCProvider:main] Cluster not yet configured; using fallback local configuration
15:50:26.537 INFO [o.s.s.i.SyncManager:main] [32767] Updating sync configuration ClusterConfig [allNodes={32767=Node [hostname=localhost, port=6642,
15:50:26.623 INFO [o.s.s.i.r.RPCService:main] Listening for internal floodlight RPC on localhost/127.0.0.1:6642
15:50:26.920 INFO [n.f.c.i.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
```

Figure – 19 TopoGuard/TopoGuard + running on eclipse IDE

Once the topology is created, we can save the script in python format and execute it from the command prompt using the mininet commands. Once the topology runs perfectly without any errors, you can see the message in the controller that a new switch connection from the IP address has been added, and you can see the in which port the switch is connected; you can find it in figure 20.

```
15:50:35.829 INFO [n.f.c.i.OFChannelHandler:New I/O server worker #2-1] New switch connection from /192.168.149.141:35170
15:50:35.837 INFO [n.f.c.i.OFChannelHandler:New I/O server worker #2-1] Disconnected switch [/192.168.149.141:35170 DPID[?]]
15:50:36.042 INFO [n.f.c.i.OFChannelHandler:New I/O server worker #2-2] New switch connection from /192.168.149.141:35173
15:50:36.109 INFO [n.f.c.i.OFChannelHandler:New I/O server worker #2-2] Switch OFSwitchBase [/192.168.149.141:35173 DPID[00:00:00:00:00:00:00:01]] b
15:50:36.115 INFO [n.f.c.OFSwitchBase:New I/O server worker #2-2] Clearing all flows on switch OFSwitchBase [/192.168.149.141:35173 DPID[00:00:00:00
15:50:36.117 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:00:01 connected.
15:50:36.285 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] Device:00:00:00:00:00:03 is added on:
15:50:36.286 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] sw:1,port:3
15:50:36.519 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] Device:00:00:00:00:00:02 is added on:
15:50:36.520 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] sw:1,port:2
15:50:36.664 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] Device:00:00:00:00:00:04 is added on:
15:50:36.665 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] sw:1,port:4
15:50:36.911 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:00:01 port s1-eth1 changed: UP
15:50:36.987 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] Device:00:00:00:00:00:01 is added on:
15:50:36.988 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] sw:1,port:1
15:50:57.283 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] Device:00:00:00:00:00:05 is added on:
15:50:57.283 INFO [n.f.t.TopoloyUpdateChecker$PortManager:New I/O server worker #2-2] sw:1,port:5
```

Figure – 20 switch connection to the controller

After connecting to the controller, you can check your topology in GPU mode by visiting the local IP address of your system (localhost:8080). You can find the page developed by the floodlight community, which can show you the topology in a graphical interface, and you can see the switches and Ip addresses of the host and mac addresses. You can find it in Figures 21 and 22. The GPU details switches and hosts without using commands in the terminal.

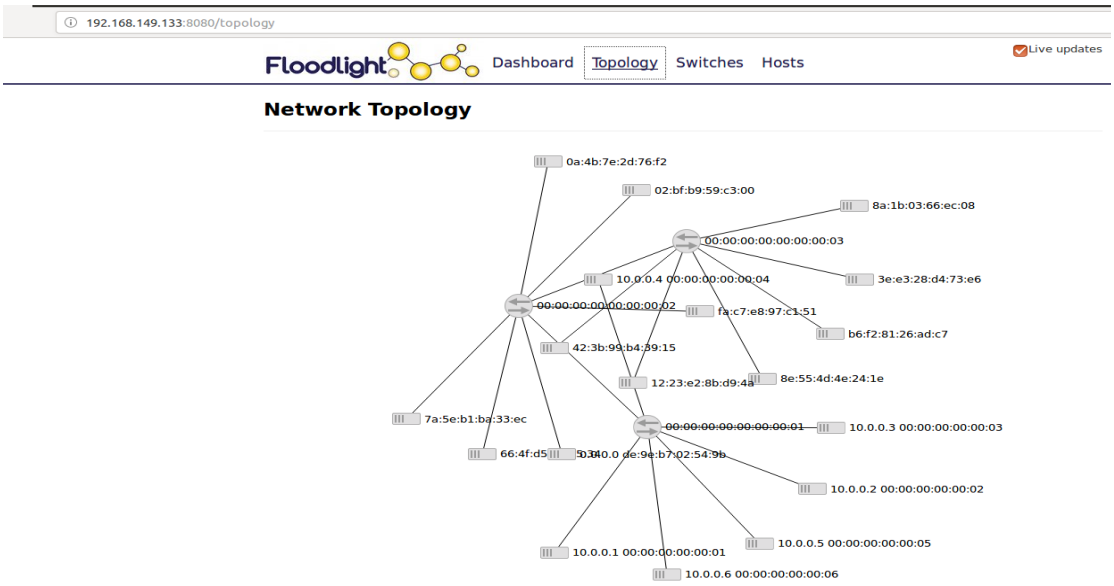
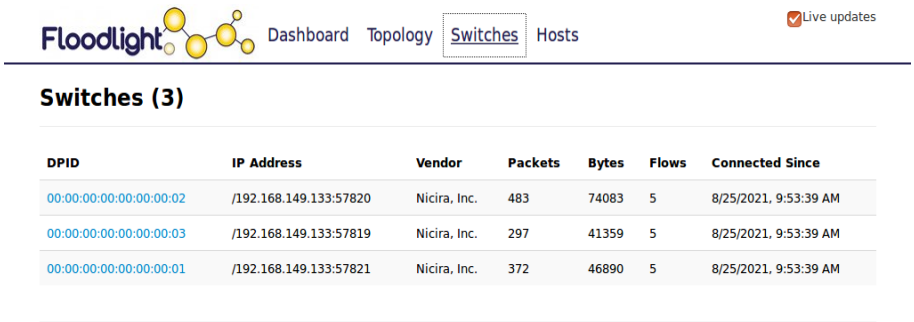
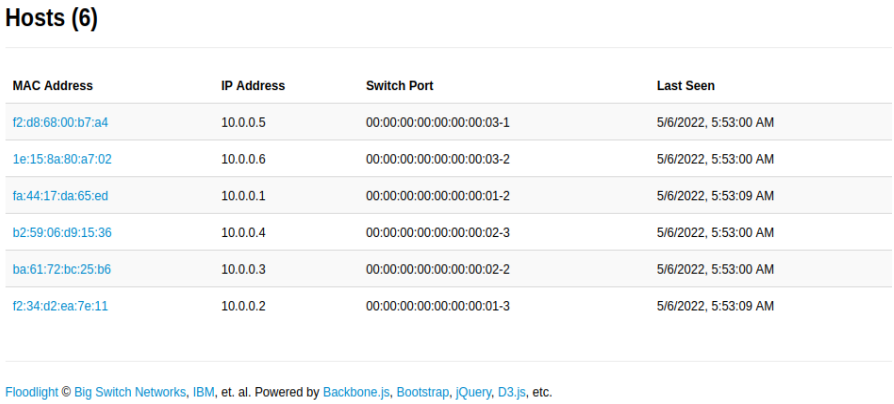


Figure – 21 GPU of floodlight and topology of the network



22 (a) switches info



22 (b) hosts info

Figure - 22 switches (a) and hosts information (b) in floodlight GPU

The switches, hosts, and controller are ready for performing the attack scenario on it; there are different methods to perform the attack scenario, we can use script language with the help of programming languages to launch an attack, or we can use built-in tools/libraries in the Linux which will perform the scenario, or we can use attack tools like Nmap, Metasploit, john the ripper, sqlmap, autopsy and ettercap. I am using Ettercap for my attack scenario, which can perform ARP poisoning, MITM, DNS spoofing, DoS attack, and credentials capture. Our attack mainly depends on the MITM (Man In The Middle) attack, and we use ARP poisoning in MITM to collect the information on the network and the host we selected to attack. In this scenario, the attacker already has a host under his control and is performing his attack from that host on the selected one.

For our convenience, we will choose the above scenario, and the attacker host will be host 3 (in switch 2)and the target hosted will be host 1 (in switch 1). Host 3 (the attacker) will try to send a standard ping LLDP message to all the hosts in the network to identify the number of devices connected to it. Once the number of hosts and Ip address are discovered, the attacker will choose one host or all the hosts. Here we choose to host 1; the attacker now subdues the host 1 traffic and redirects to himself by pretending to be host 1; he can get data (credentials, traffic data etc.,.) from all the hosts pretending to be host 1. This is how the attacker can compromise an SDN and collect the required information and credentials.

The ettercap will do all the manual work, which was mentioned in the above scenario, and the ettercap will be opened in a new xterm window of the attacker host; the attacker will run the ettercap in the graphical interface and run the network search with the subnet of 255.255.255.0 because the Ip address of all the hosts will be in the range of 10.0.0.1 to 10.244.244.244. The ettercap will give the hosts list in the host's list bar, and the attacker will choose the hosts to attack from the list and select the ARP poisoning MITM attack from the list of attacks. The ettercap has two options unified sniffing and bridged sniffing for network searching; unified sniffing uses a single network device, and sniffing will happen on the same network port, while bridged sniffing uses multiple networking devices and sniffs the traffic across the bridge from one device to another. The ettercap has an unoffensive mode, which will scan the network and find the IP and mac addresses within the network. The ettercap will give the information on data packets like Wireshark, but it is less potent than Wireshark. However, it can still capture the port number, destination, sender port, and number of packets. The attacker uses unified sniffing with offensive mode and then chooses the network interface (i.e., wlan0, eth0, eth1, etc.); the ettercap will start sniffing from that interface and scan all the ports to find the connected devices. The attacker wants a brief view of the ports, state and protocols used by the hosts in the network, and the connections tab will give the attacker insight into each host, giving him the power to snoop. The attacker can find what the attacker is browsing, and their manufacturer and all the basic to primary information

can be monitored and captured. The attacker chooses the host 1 IP address in the target pane; the IP address of host 1 is 10.0.0.1.

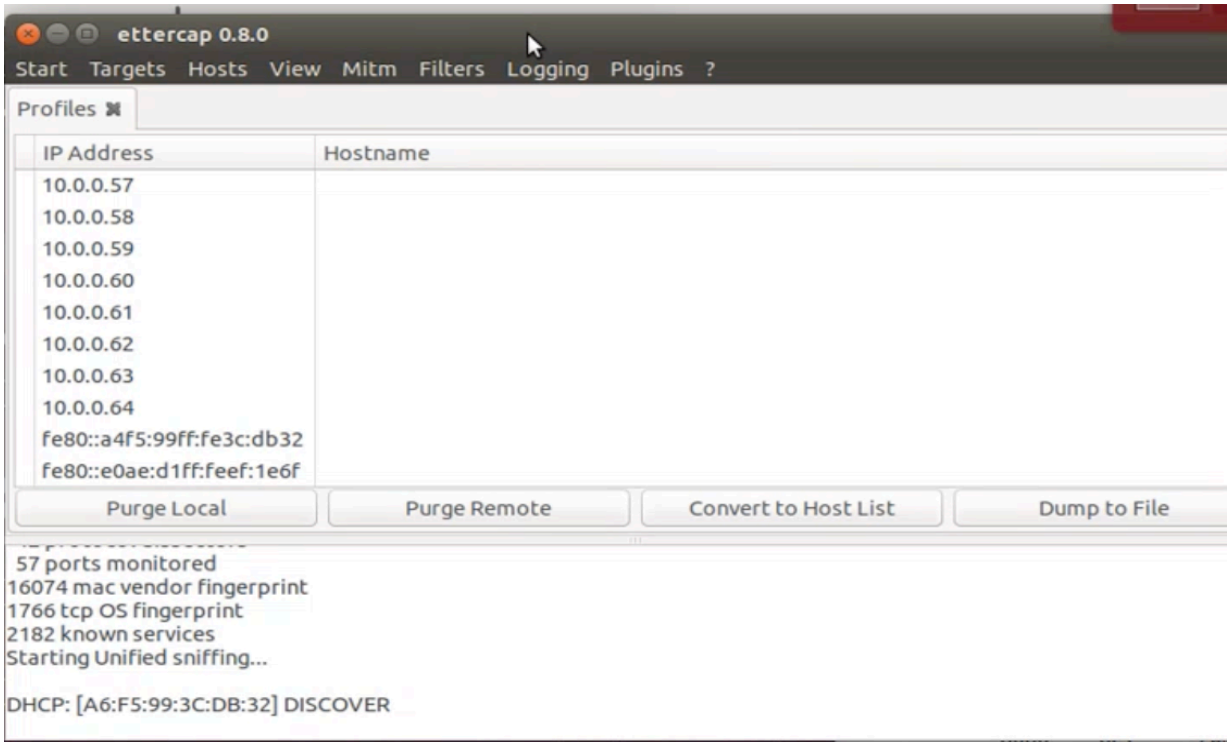
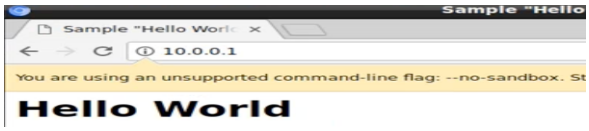


Figure – 23 Ettercap GUI and hosts list

The attacker is now sending the ICMP and LLDP packets as host 1 by dropping the packets of host 1 from the network. The hosts who think the host 1 server is working correctly will send their request to Host 1, but they will reach host 3 and host 3 will send back a response as host 1. This web page example is for understanding the readers; for research purposes, there will be no difference in web pages on real-world attacks.

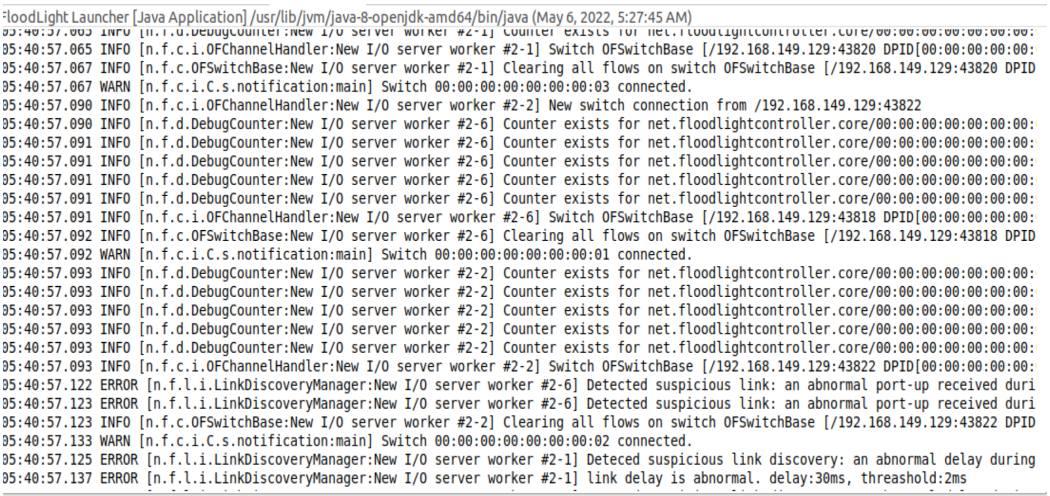


24 (a) genuine host/ real host

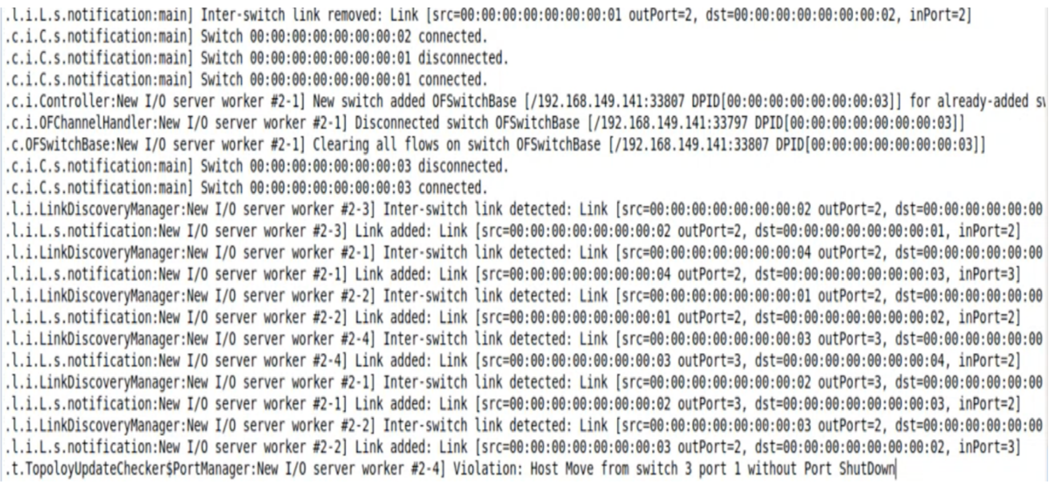
24(b) attacker/compromised host

Figure – 24 genuine host 1-24(a) and hacked by host 3 on host 1-24(b)

The TopoGuard will recognise the attack because it is a host location hijacking attack, and the TopoGuard's primary purpose is to provide defence against this kind of attack. Because the attacker is sending the packets as host 1 from another switch and port without shutting down the port used by host 1, the TopoGuard can detect it. You can find it in figure 25.



25 (a) Host Hijack Detection



25 (b) security counter

Figure -25 (a) Host Hijacking detection by TopoGuard 25 and (b) its counter

To overcome the defence of TopoGuard, we use a port probing attack in which the attacker will try to impersonate the host but will be using the host's mac address to join the same switch by shutting down the host first. The scenario will be like the attacker joining the network with his mac and IP

address, and then he will send the Arp messages to the hosts nearby to fetch the mac address of the target. The attacker waits until the target has initiated a port-down message and is no longer in the network and bound to the port. The attacker will spoof the packet by using the ifconfig command to change his Ip and MAC addresses to the targets and join the network switch by sending an ICMP or HTTP packet over the network. The TopoGuard thinks that it is a valid one and lets the attacker join the network, and now the host hijacking is successful, and the network does not know that it is compromised until the target host re-joins the network. The controller will assign a new port for the re-joined victim and oscillate between the attacker and target host based on the last sent packet route; thus, the TopoGuard has failed to provide a defence against these attacks. To overcome this, an extra feature is added to TopoGuard to inspect the processing time of LLDP packets (LLDP construction and LLDP processing) will be monitored and it is named as TopoGuard +; if there is any slight difference in the packets, the host will be shut down the host and remove it from the topology. The ettercap has successfully bypassed the security of TopoGuard +, and the attacker has successfully gained control over the host by using Link Latency Attack.

Since TopoGuard + relies on the LLDP processing time, the attacker will use bots to inject unwanted traffic from the end hosts to increase the processing time of the network and the switches. Due to an increase in the processing time in the network, the switches will try to match, but the processing time also increases and the TopoGuard + relies on probe packets to keep the view of the topology updated. The attacker will use this chance to add a fake link among the switches, and the TopoGuard + has a mechanism to identify the fake links, and the name of the defence is Link Latency Inspector. The LLI will have the threshold value of the previous packets and the time limit. However, the attacker has overloaded the network with unwanted traffic, increasing the LLDP packets' threshold limit. This attack can be named an overload phase; now the attacker will move on to the next phase i.e., the relay phase, where he stops flooding after creating a fake link to the network; the network controller will issue LLDP packets to verify the topology and the hosts by calculating the LLDP latency, due to the previous flooding the Round-Trip Time is increased. LLI will get a negative value considered valid by the TopoGuard + and add fake links to the switches. The leading cause of the problem is due to authenticity of packets between the controller and switches and hosts, which leads to the vulnerability of network topology and the cause of poisoning attacks like ARP. To overcome this problem, we use Machine Learning as an authenticator to check the authenticity of the packets flowing in the network.

We have used Wireshark to collect data packets from the above scenario. The data will be converted into CSV format; hence, CSV is the most used format for data saving and reading for programming languages. Wireshark will be listing the ports and network adapter of the topology; we

use filters in the Wireshark to capture TCP and LLDP packets in the network; if we want, we can write our own rules and filters according to the functionality we want. The primary analysis of the packet can be done in the Wireshark. If you click on the packet, it will give the packet information like the sender's IP, receiver's IP, protocol, MAC of the sender and receiver, size of the packet and when it was created. The data will be raw, and the system cannot understand how to classify between attacker and actual packets; we will create a new feature in the dataset and name it as attack/label to classify with 1 and 0 (1 will be for attackers and 0 will be actual) which will make the classification easy and help the model to train powerfully and precisely. Now that the data collection is completed and we have a successful scenario implemented on the TopoGuard +, we must build an ML model using the algorithms, train the model with a known dataset, and use the model on the data collected by the Wireshark in the real-time. Before we build the model, we will clean the dataset using the python programming language and load the required libraries like NumPy, pandas, sklearn, matplotlib and seaborn. We will read the data using pandas, check for any null values, drop the null value rows from the dataset, and create a correlation matrix and scatter and density plots for the dataset. The correlation matrix is used to depict the correlation between all the possible pairs in the dataset, which will give the user an idea of the dataset and understand which feature depends on another, which will give an idea to use which variables for the model. We will encode the data using pre-processing techniques from the sklearn to encode the data for running in the algorithms because the algorithm can only understand binary language, so we encode it using Ordinal Encoder and fit the data into a new data frame by splitting the features we require to make the algorithm running and results we desired. Then we will drop the y column, which is to be predicted by the algorithm and save it in another data frame. Later in the encoding, we will split the data into training and testing using train_test_split from the sklearn library, and we will choose the ratio of splitting by the parameter test_size.

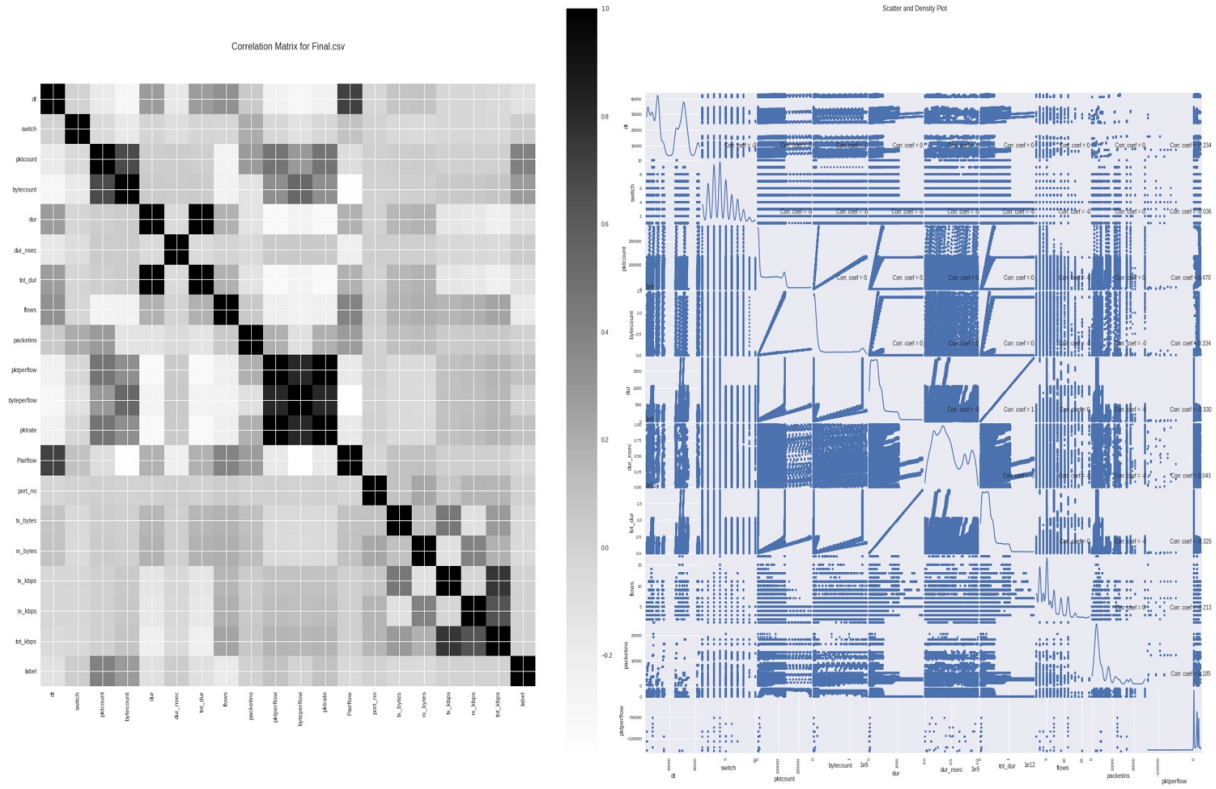


Figure – 26 Scatter plot and density plot

The ML model uses six algorithms: Logistic Regression, Random Forest, K Neighbours Classifier, Decision Tree, Gaussian and Linear SVC (Support Vector Classifier).

- Logistic Regression is also called logit or MaxEnt classifier, and it is supervised learning; it uses one vs rest if the multiclass option in the algorithm is set to OvR, and if the algorithm is meant to use the multinomial option, it uses cross-entropy. We will define the parameters like a penalty (l1 or l2 or both), dual (only used when n_samples>n_features), c (inverse of regularisation strength) and solver (lbfgs, liblinear, newton-cg, newton-cholesky, sag and saga), the solver is used based on the size of datasets and choosing of multiclass or multinomial. We will calculate the accuracy of training and testing data, and later we will build a confusion matrix.

- Random Forest is one of the supervised learning and a meta-estimator that fits several decision tree classifiers on different dataset sub-samples to improve the model's accuracy and control overfitting. We can control the overfitting by using parameters like max_samples and bootstrap in the algorithm. The parameters chosen here are criterion (function to measure), min_samples_split (number of samples to split an internal node), min_samples_leaf (minimum number of samples required to be in leaf node), max_leaf_nodes (maximum number of leaf nodes in a tree), bootstrap and oob_score (to use out of bag samples). We will calculate the

accuracy of training and testing data, and later we will build a confusion matrix. Random Forest will be fitted over one vs rest classifier to achieve better results when using multiclass classification.

- K Neighbours is also called K Nearest Neighbours. It is a supervised learning; the classification of the dataset will be determined by the k nearest neighbours using the unclassified sample of data. If the k neighbours belong to a particular class, then all the unclassified data will be classified into that class; if the value of k is higher, there will be less noise in the classification [17]. The parameter used for tuning the algorithm are n_neighbours (number of neighbours), weights (uniform, distance and callable), p (power parameter), metric (callable or minkowski), leaf_size (size of the leaf for each node). We will calculate the accuracy of training and testing data, and later we will build a confusion matrix.

- Decision Tree is also surprised learning and uses classification through a learning tree; each node in the tree represents the features of the data, and the branches in the data represent the conjunction of features which leads to classification, and each leaf node is the class label [17]. The decision tree is one of the several predictive models used in statistics, data mining and ML. The decision tree has classification and regression tree models, also called CART. The features used in Decision Tree for tuning the algorithm are criterion (function to measure), splitter, min_samples_split (number of samples to split an internal node), min_samples_leaf (minimum number of samples required to be in leaf node), max_features (number of features in the train data). We will calculate the accuracy of training and testing data, and later we will build a confusion matrix.

- The gaussian method is part of the naïve Bayes algorithm. The naïve Bayes is based on conditional probability to calculate the probability of the event based on prior knowledge of the conditions that might relate to the events. The gaussian is used when the predictor values are continuous and follow the gaussian distribution, most of the time, the result given by the naïve Bayes is very satisfactory to the users, and programmers and mathematicians widely use it to solve big problems. We will calculate the accuracy of training and testing data, and later we will build a confusion matrix.

- Support Vector Machine is a part of the supervised learning family and is widely famous, like naïve Bayes. The main idea of SVM is to use high-dimensional feature space to map the input vectors by applying the different kernels according to the requirement. We use a linear kernel,

which supports dense and sparse data and uses multiclass to support one vs rest. We use a pipeline with a standard scalar in the SVC to train the algorithms and process the train and test datasets. We will calculate the accuracy of training and testing data, and later we will build a confusion matrix.

The ML models are built, and each model is saved with a standard prefix model and a number; then, we will calculate the training error and testing error for the model and save the y_test and predict_proba (log probability) values for each model in separate variables for auc_roc curve. The first model will be created and tested using the known dataset to find the errors and tune the algorithm to work against the feature data; later, the same models will be used against the real-time dataset. The results will be discussed in further sections.

5. Results and Discussions

The ML model is booming, and the prediction and accuracy are perfect, as expected. In this section, we will discuss the results that are produced by the ML algorithm.

The below scatter plot figure 27 shows the blob for each example of the cluster in the dataset, and each colour is a unique cluster, and the range is for the extent to which each cluster is formed. The known dataset plot shows each cluster sample for each unique prediction which shows the graph different from the standard KNN clustering plot. However, there are so many unique values in the dataset, and that is why the cluster points are away from each other. The second plot for real-time has very limited unique values, and the centroid of the graph is very common for all the unique prediction values.

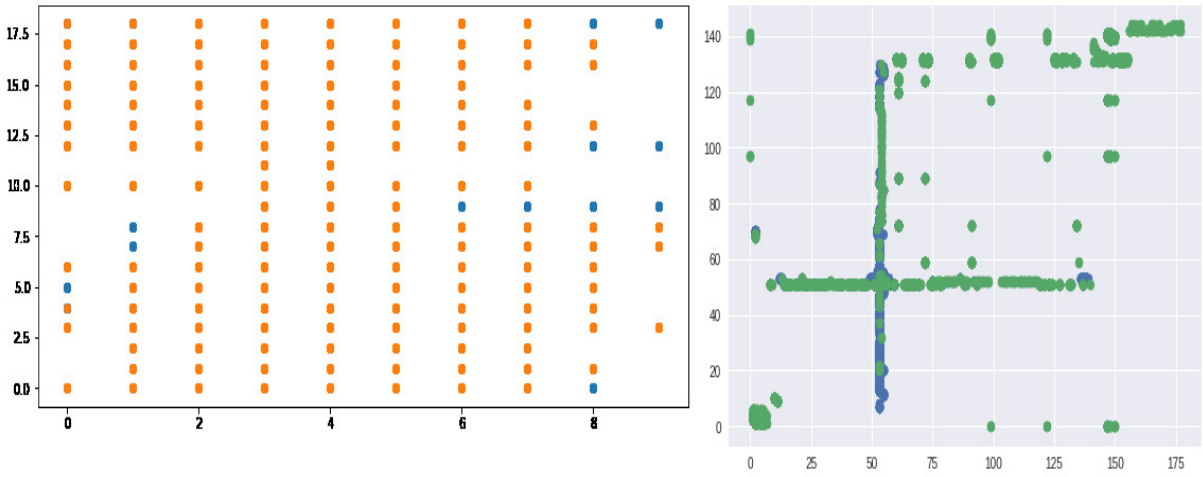


Figure – 27 KNN prediction cluster graph for known dataset and real-time data

The known data correlation and density plot are already discussed in figure 26 above; the correlation matrix aims to find the features' strength and dependency and how one feature will affect the other in the prediction. Below figure 28 are the correlation matrix and density plot for the real-time data; the features are mainly dependent on the Time, Length, Attack and ABF.

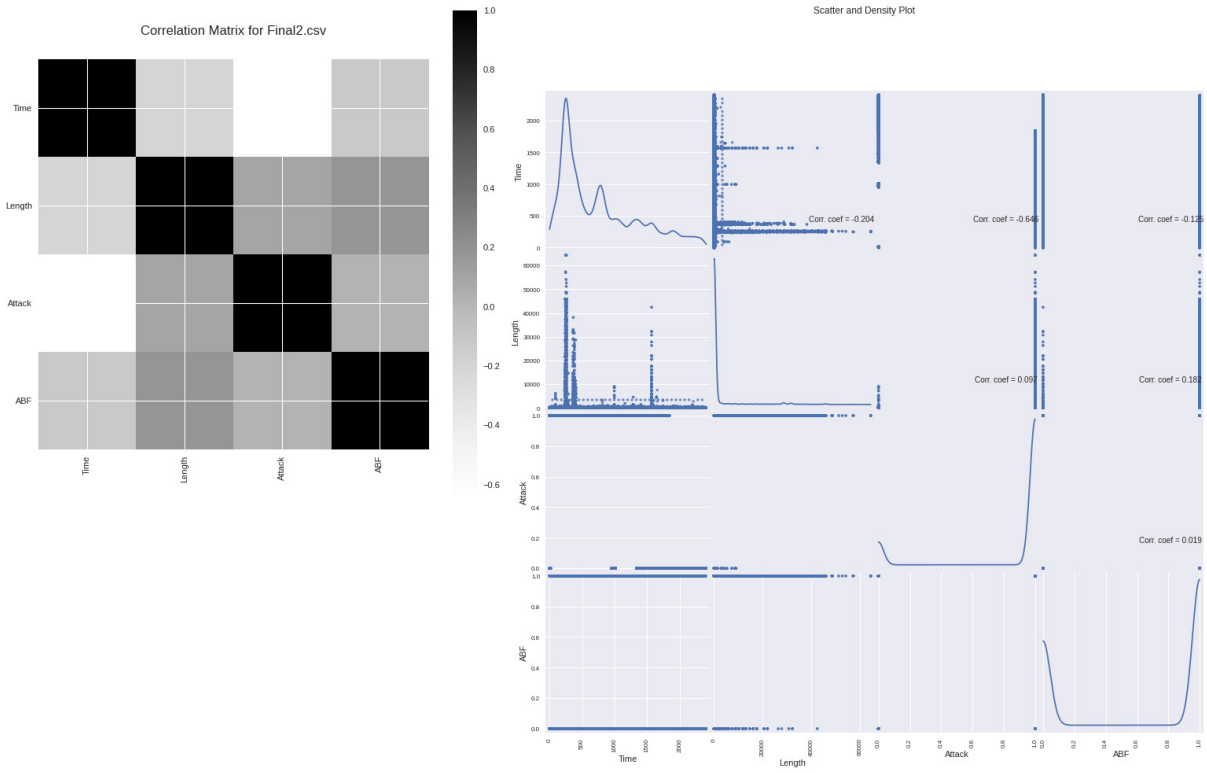


Figure – 28 Correlation matrix and scatter plot for the real-time dataset

Model	**Accuracy**	**Precision**	**Recall**	**F1 -Score**
Logistic Regression	0.8754	0.87	0.88	0.88
Random Forest	0.9431	0.95	0.94	0.94
K Neighbour	0.9827	0.98	0.98	0.98
Decision Tree	0.9214	0.93	0.92	0.92
Gaussian	0.6648	0.69	0.66	0.67
Support Vector Machine	0.8789	0.88	0.88	0.88

Table – I shows the performance of different classifiers on the Known dataset.

The defence mechanism developed through the ML algorithm will be evaluated by factors like Accuracy, Precision, Recall, F1- Score AUC Score and ROC plot. These factors will determine the efficiency of the ML algorithm in the AI world; if the accuracy is high, then the algorithm's performance and prediction are high. According to the research done by Soltani, S in the paper “Link Latency Attack in Software-Defined Networks,” the TopoGuard + accuracy is predicted as 84.28. However, we are not

calculating the accuracy for TopoGuard +; the accuracy of the KNN in the experiment is higher than the TopoGuard +, which shows the efficiency of the system. For the known dataset taken from the internet, the accuracy is 98.27 by KNN, which can be seen in table I, and the most minor performance is by Gaussian, which is 66.48; for the real-time data, we can see that the KNN performed 99.45 and most minors were performed by SVM, which is 91.26 and it can be seen in table II.

The huge difference in results for Known data and real-time data is due to the model trained by a known dataset, which has all the features, and is ready to implement on the same data with different fields, so the accuracy is booming for the real-time dataset. These results are very satisfactory and can prove that ML can help in the defence of the SDN.

Model	**Accuracy**	**Precision**	**Recall**	**F1 -Score**
Logistic Regression	0.8370	0.84	0.84	0.84
Random Forest	0.9817	0.98	0.98	0.98
K Neighbour	0.9945	0.99	0.99	0.99
Decision Tree	0.9840	0.98	0.98	0.98
Gaussian	0.9126	0.91	0.91	0.91
Support Vector Machine	0.9126	0.84	0.84	0.84

Table – II shows the performance of different classifiers on the real-time dataset

The AUC ROC curves will determine the error rate and scores in the ML, and these functions will determine the algorithm's correct prediction of LLA in the TopoGuard +. You can see the graphs and scores below in figure 29, and the plot will show how the algorithms fluctuate between the true positive rate (TPR) and the false positive rate (FPR). You can observe that the KNN performed very well among all the remaining algorithms, and you can see the remaining algorithms' performances in the below graph.

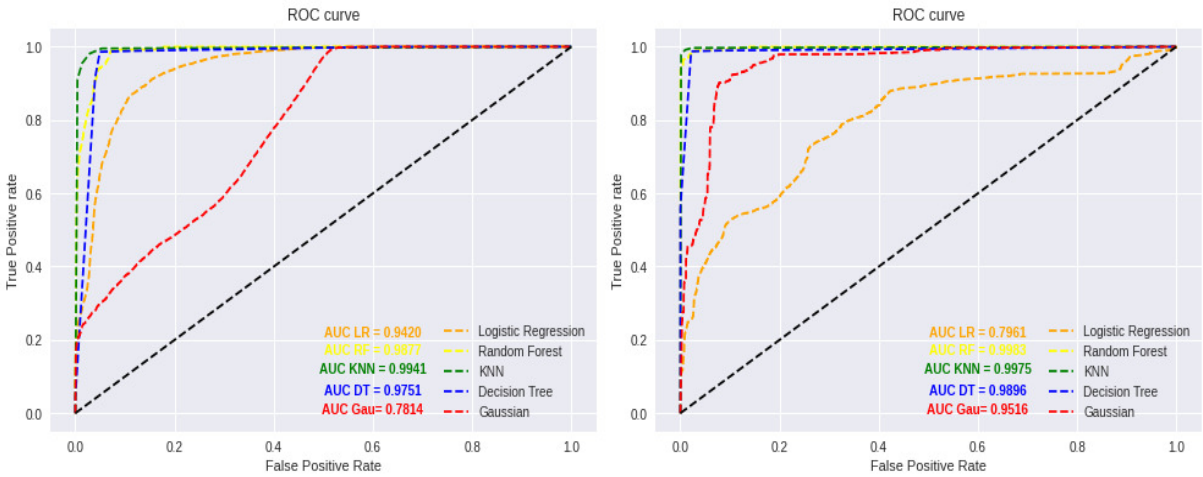


Figure – 29 AUC and ROC curves for Known dataset and real-time data

5.1 CONCLUSION

ML is the growing tool in the world, and on the other hand, SDN is the growing trend in Networking in current industries like telecom, IT, and everywhere where networking is essential. The SDN market is now at 14 billion USD, and it will grow by 32.7 billion USD by 2025 [1]; the market and the business using SDN are increasing at the same time, and the defence and protection against the attacks is the big question in front of the developers and the industry clients. The defence proposed in this paper will be helpful in the future of SDN. We can add more powerful algorithms and better techniques to the current solution, which will support not only TopoGuard + but also other defence mechanisms which will be developed in the future. The more you train the ML with different datasets with the attacking data info, the more influential the model will become.

The future to improve this would be to use this in a real-time environment i.e., telecom services like EE, Vodafone, and three ltd and support the advancements of the services to the clients; as we develop this security measure, we must approach the client and make a deal with him and show him how this defence mechanism will enhance his market value. The defence mechanism needs to be improvised and provided with security patches and bug fixes as there will be new malware attacks in the market. There will be no break for intruders trying to compromise the system and steal the data from the companies; the defence mechanism can be implemented wherever the SDN is used for networking, and there are no limitations for the application. One of the future scopes for this is to integrate with the cloud and deploy as API for the clients, which will be easy to access for any client and can integrate with his main router easily. Another scope is to add unique authentication (random tokenisation / OTP) for hosts whenever they connect with the controller, which will reduce the risk of being compromised by the hijacker.

References:

[1]. Soltani, S., Shojafar, M., Mostafaei, H., Pooranian, Z. and Tafazolli, R. (2021). Link Latency Attack in Software-Defined Networks. 2021 17th International Conference on Network and Service Management (CNSM). doi:10.23919/cnsm52442.2021.9615598.

[2]. Karmakar, K.K., Varadharajan, V. and Tupakula, U. (2019). Mitigating attacks in software-defined networks. Cluster Computing, 22(4), pp.1143–1157. doi:10.1007/s10586-018-02900-

[3]. Elsayed, M.S., Le-Khac, N.-A. and Jurcut, A.D. (2020). InSDN: A Novel SDN Intrusion Dataset. IEEE Access, 8, pp.165263–165284. doi:10.1109/access.2020.3022633.

[4]. G. C. Kesavulu, "Preventing DDoS attacks in Software Defined Networks," 2021 2nd International Conference on Range Technology (ICORT), 2021, pp. 1-4, doi: 10.1109/ICORT52730.2021.9581968.https://ieeexplore.ieee.org/document/9165459 (Attack Detection on the Software Defined Networking Switches)

[5]. W. Wang et al., "BalanceFlow: Controller load balancing for OpenFlow networks," Cloud Computing and Intelligent Systems (CCIS), IEEE 2nd International Conference, vol. 2, pp. 780-785, 2012.

[6]. S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani and M. K. Khan, "Topology Discovery in Software Defined Networks: Threats, Taxonomy, and State-of-the-Art," in IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 303-324, Firstquarter 2017, doi: 10.1109/COMST.2016.2597193.

[7]. Krishnan, P., Duttagupta, S. and Achuthan, K. (2019). VARMAN: Multi-plane security framework for software defined networks. Computer Communications, 148, pp.215–239. doi:10.1016/j.comcom.2019.09.014.

[8]. R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique and Z. Anwar, "Cyberpulse: A Machine Learning Based Link Flooding Attack Mitigation System for Software Defined Networks," in IEEE Access, vol. 7, pp. 34885-34899, 2019, doi: 10.1109/ACCESS.2019.2904236.

[9]. H. Z. Jahromi and D. T. Delaney, "An Application Awareness Framework Based on SDN and Machine Learning: Defining the Roadmap and Challenges," 2018 10th International Conference on Communication Software and Networks (ICCSN), 2018, pp. 411-416, doi: 10.1109/ICCSN.2018.8488328.

[10]. Y. Yu, L. Guo, Y. Liu, J. Zheng and Y. Zong, "An Efficient SDN-Based DDoS Attack Detection and Rapid Response Platform in Vehicular Networks," in IEEE Access, vol. 6, pp. 44570-44579, 2018, doi: 10.1109/ACCESS.2018.2854567.

[11]. R. Skowyra et al., "Effective Topology Tampering Attacks and Defenses in Software-Defined Networks," 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2018, pp. 374-385, doi: 10.1109/DSN.2018.00047.

[12]. S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures." in NDSS, vol. 15,2015, pp. 8-11.

[13]. MININET. An instant virtual network on your laptop (orother pc. [Online]. Available: http://mininet.org/

[14]. Floodlight. Open sdn controller. [Online]. Available: http://floodlight.openflowhub.org/

[15]. N. M. Abd Elazim, M. A. Sobh and A. M. Bahaa-Eldin, "Software Defined Networking: Attacks and Countermeasures," 2018 13th International Conference on Computer Engineering and Systems (ICCES), 2018, pp. 555-567, doi: 10.1109/ICCES.2018.8639429.

[16]. S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa and B. Yang, "Predicting network attack patterns in SDN using machine learning approach," 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 167-172, doi: 10.1109/NFV-SDN.2016.7919493.x

[17]. J. Xie et al., "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 393-430, Firstquarter 2019, doi: 10.1109/COMST.2018.2866942.

[18]. OpenFlow-enabled SDN and Network Functions Virtualization ONF Solution Brief. (2014). [online] Available at: https://www.ramonmillan.com/documentos/bibliografia/OpenFlowEnabledSDN&NFV_ONF.pdf [Accessed 6 Dec. 2022].

[19]. issuu.com. (n.d.). LOAD BALANCING IN SOFTWARE-DEFINED NETWORKS USING ADAPTIVE GENERIC MASTER AND SLAVE ARCHITECTURE by Transtellar Publications - Issuu. [online] Available at: https://issuu.com/tjprc/docs/2-67-1594793754-328ijmperdjun2020328 [Accessed 6 Dec. 2022].

[20]. J. A. Pérez-Díaz, I. A. Valdovinos, K. -K. R. Choo and D. Zhu, "A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning," in IEEE Access, vol. 8, pp. 155859-155872, 2020, doi: 10.1109/ACCESS.2020.3019330.

[21]. A. O. Sangodoyin, M. O. Akinsolu, P. Pillai and V. Grout, "Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning," in IEEE Access, vol. 9, pp. 122495-122508, 2021, doi: 10.1109/ACCESS.2021.3109490.

[22]. U. Tupakula, V. Varadharajan and K. K. Karmakar, "Attack Detection on the Software Defined Networking Switches," 2020 6th IEEE Conference on Network Softwarization (NetSoft), 2020, pp. 262-266, doi: 10.1109/NetSoft48620.2020.9165459.

[23]. Satheesh, Professor N., et al. "Flow-Based Anomaly Intrusion Detection Using Machine Learning Model with Software Defined Networking for OpenFlow Network." Microprocessors and Microsystems, vol. 79, no. 0141-9331, 14 Oct. 2020, p. 103285. science direct, 10.1016/j.micpro.2020.103285.

[24]. McKeown, Nick & Anderson, Tom & Balakrishnan, Hari & Parulkar, Guru & Peterson, Larry & Rexford, Jennifer & Shenker, Scott & Turner, Jonathan. (2008). OpenFlow: Enabling innovation in campus networks. Computer Communication Review. 38. 69-74. 10.1145/1355734.1355746. https://ieeexplore.ieee.org/document/8639429 (Software Defined Networking: Attacks and Countermeasures)

[25]. ETSI, "Network Functions Virtualisation (NFV): Architectural Framework," ETSI GS NFV, vol. 2, p. V1, 2013.

[26] Foundation, Open Networking. "OpenFlow Switch Specification." Opennetworking, 25 June 2012.

[27] Ahuja, Nisha; Singal, Gaurav; Mukhopadhyay, Debajyoti (2020), "DDOS attack SDN Dataset", Mendeley Data, V1, doi: 10.17632/jxpfjc64kr.1

Code references: -

[1] BHANDARI, A. (2020). AUC-ROC Curve in Machine Learning Clearly Explained. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/.

[2] Scikit-learn.org. (2019). scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation. [online] Available at: https://scikit-learn.org/stable/index.html.

[3] GitHub. (n.d.). False review detection and model comparision · harshika14/CMPE-257_ProjectTeam12@0d25e01. [online] Available at: https://github.com/harshika14/CMPE-

257_ProjectTeam12/commit/0d25e010a9387719c414b1589c74ebdfccc6678e?cv=1 [Accessed 1 June. 2022].

[4] Brownlee, J. (2020). 10 Clustering Algorithms With Python. [online] MachineLearningMastery.com. Available at: https://machinelearningmastery.com/clustering-algorithms-with-python/?cv=1&fbclid=IwAR3i9bv5u0l9gPN0FzJ2x3bl01IWJh8LfCQfGoZnFl-AmEY3HC0GGPldu-Q [Accessed 9 Jan. 2023].