ABSTRACT

Large companies have been increasingly relying on Software-defined Networks (SDN) to meet their programmable network needs in recent years. However, SDN, like other networks, has some security issues. To solve such problems, a variety of applied sciences are employed, with machine learning being one of the most effective. Where other applied sciences struggled, machine learning demonstrated the ability to discover data patterns. This makes it an excellent candidate for anomaly-based detection and intrusion detection systems (IDS).

In this project, I propose a new anomaly-based IDS that takes advantage of SDN's ability to provide statistical elements for any flow that passes through the group and passes these facets to a vote casting computer comprised of countless machine learning algorithms, giving the system the ability to learn about users' behavior and anticipate any potential intrusion. The balloting machine is trained and tested using NSL-KDD, and the results indicate an increase in accuracy and a decrease in the false-positive rate.

LIST OF FIGURES

CONTENTS

Chapter 1

Introduction

The want for programmable networks has drawn the attention of data-centers and big-data corporations to new paradigms like software program defined network “SDN”, which turns into a trend in the ultimate few years. SDN’s main purpose is to separate the manager and the records planes, with the controller being in a position to generate and adjust the flow tables to go well with the network services which are running as applications within the controller. But like each new paradigm, SDN faces many challenges, especially with security. Intrusion detection systems “IDS” are viewed as one of the fine solutions for network security, as it’s capable to predict and alarm the network administrator about feasible intrusions.

In the closing decade, IDSs are more interested in analyzing the users’ behavior to predict intrusions. Using new technologies like machine learning, which is ideal for such troubles as it is in a position to extract new facts with each and every interaction between the users and the network, and it is capable to find patterns in this records which helps analyzing the users’ behavior. The problem of the usage of machine mastering with community safety is the want of statistics extraction from the network, which increases the overhead in it, but after SDN used to be introduced, it is viable to gain from its properties, as it affords statistical aspects about the traffic that passes the network.

1.1 SDN

1.1.1 What’s SDN

“The decoupling of control and packet-forwarding planes in the network” is how SDN is described. It enables networks to connect directly to applications through application

programming interfaces (APIs), improving overall application performance and security while also allowing for the development of a scalable, dynamic network architecture that can be modified as required.

SDN is a network paradigm that enables programmatic management and control, as well as network resource optimization, by using open APIs. As SDN decouples network configuration and traffic engineering from their critical hardware infrastructure, this network control is established.

1.1.2 Basic SDN architecture

Control and data planes, as well as early SDN implementation

By breaking down the network into the following different planes, SDN assists users in virtualizing their hardware and working to build a computer network:

1. Net Flow's control plane provides performance and fault management, and it's often used for handling system configurations that are remotely connected to a software-defined network, much like protocols.

2. The data plane is responsible for directing traffic to its final destination. Before traffic enters the data plane, the control plane uses the flow protocol to determine which direction it will follow. When a network administrator deals with a software-defined network and manages the network, solutions, cost-effectiveness, and creative thinking are all important considerations.

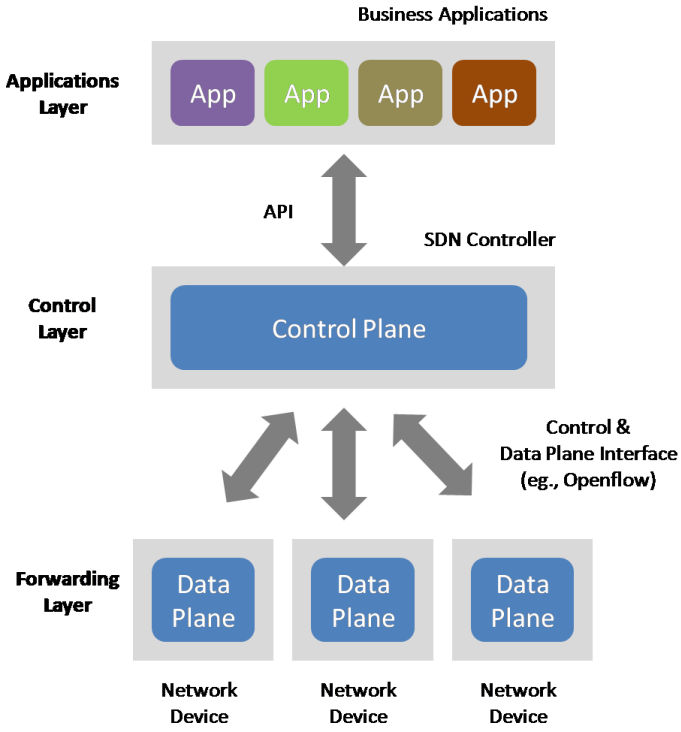

Figure 1.1 SDN architecture

1.1.3 Traditional Network

Existing network machines, such as switches and routers, are the foundation of traditional networking. Each of these devices has a specific purpose that works well with the others and helps to support the network. As the functions of a network are implemented as hardware constructs, the network's speed is normally increased.

Traditional networks face a constant challenge in terms of flexibility. Provisioning APIs are few, and most switching hardware and software are proprietary. Traditional networks also operate well with proprietary provisioning software, but this software cannot be changed as quickly as required.

The following characteristics characterize traditional networking:

1. Typical networking functions are mainly performed by dedicated devices such as switches, routers, and application delivery controllers, which use one or more switches.
2. Typical networking functionality is often implemented in dedicated hardware, such as application-specific integrated circuits (ASICs) (ASIC). The limitations of

conventional hardware-centric networking are one of its drawbacks.

1.1.4 Difference between traditional network and SDN

	SDN	TRADITIONAL NETWORK
01.	A virtual networking solution is called Software Defined Networking (SDN).	The term "traditional network" refers to an approach to networking that has been around for a long time.
02.	A Software Defined Network (SDN) is a network that is controlled from a central location.	Control is distributed in a traditional network.
03.	This network can be customized.	This network cannot be programmed.
04.	A Software Defined Network (SDN) is a network with an open interface.	The traditional network interface is closed.
05.	In a Software Defined Network, software separates the data plane and control plane.	The data plane and control plane of a standard network are built on the same plane.
06.	It allows for automated setup, which saves time.	It allows for static/manual configuration, which takes longer.
07.	It has the ability to prioritize and block those network packets.	It does not help prioritization and leads all packets in the same direction.
08.	It is simple to program as needed.	It is difficult to reprogram and replace an outdated application that is no longer in operation.
09.	Software Defined Networks have a	The traditional network has a high

	low cost.	cost.
10.	Software Defined Networks have a low structural complexity.	Traditional Network has a high level of structural complexity.
11.	Software Defined Networks have a lot of extensibility.	Traditional Network has a limited amount of extensibility.
12.	Since it is centralized, troubleshooting and reporting are simple in SDN.	Since traditional networks are distributed regulated, it is difficult to troubleshoot and report on them.
13.	It has a lower operating cost than a conventional network.	The cost of traditional network maintenance is higher than that of SDN.

Table 1.1 Difference between traditional network and SDN

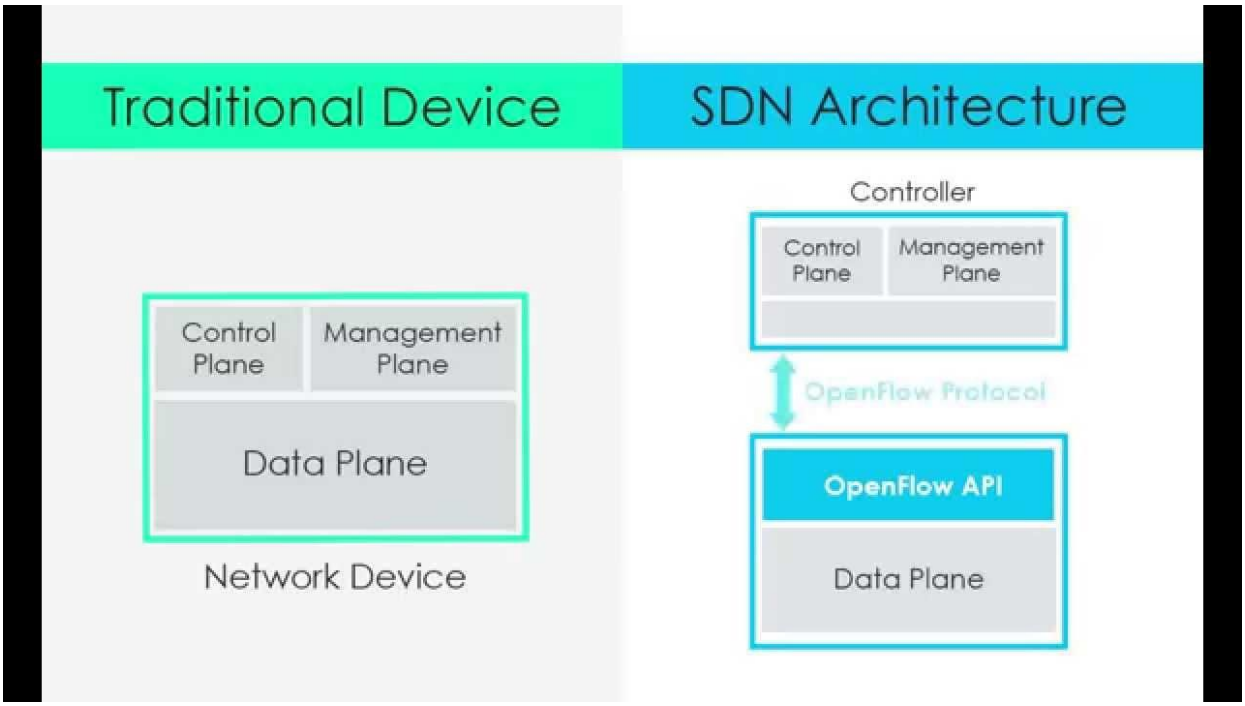

Figure 1.2 Traditional vs. SDN

1.2 Intrusion Detection System

1.2.1 NIDS

In a network, an Intrusion Detection System (IDS) is created to detect threats by tracking packets transmitted via it. IDSs detect unusual and malicious activity from both inside and outside intruders. IDS must be able to cope with issues like large network traffic volumes and extremely unequal data delivery.

The most important feature of IDS is that it exposes data sources, such as computers or networks, for unauthorized access to activities. IDSs collect data from one-of-a-kind systems and network sources, and then analyze it for potential threats. Network intrusion detection systems (NIDS) and host-based intrusion detection systems are two types of IDSs (HIDS).

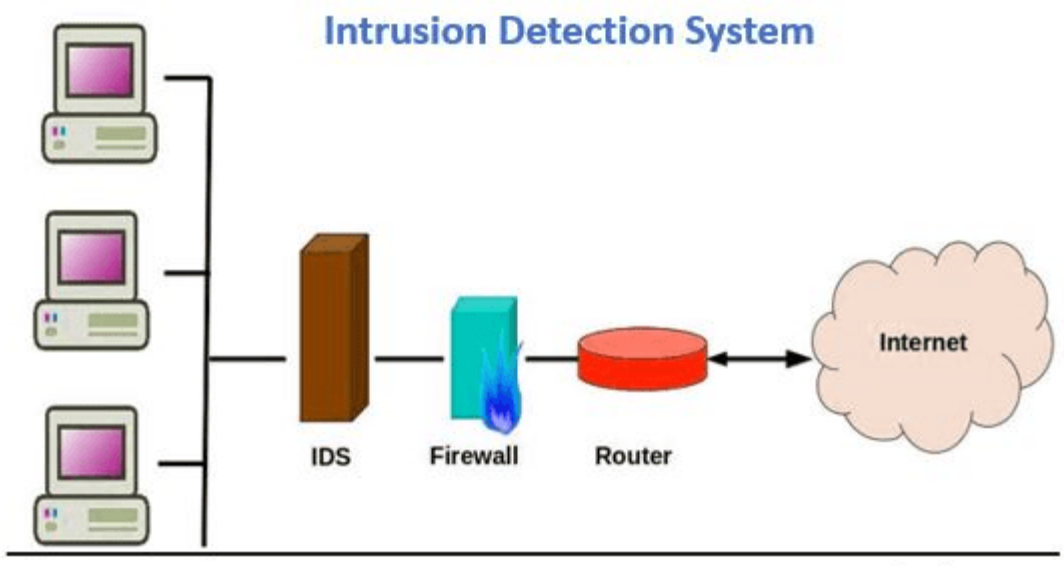

Figure 1.3 IDS

1.2.2 Implementation of NIDS

Signature-based detection, anomaly-based detection, and anomaly-based detection are the three detection methods that can be used for NIDS. A signature-based NIDS is limited to detecting malicious threats that have been identified. The detection system uses a combination of packet header and packet content material inspection rules to identify anomalous traffic flows via signature specification. For signature-based NIDS, anomaly detection techniques are designed to robotically apprehend assaults that are uncertain and unpredictable. Anomaly-based intrusion detection approaches include machine learning methods, for example.

1.2.3 Evaluation

Accuracy, false-negative rate (FNR), false-positive rate (FPR), time used, memory use, and kappa records are some of the comparative metrics used to determine the overall performance of algorithms in NIDS. The NIDS often uses accuracy, FNR, and FPR as reference standards. The following image shows a comparison of three detection strategies for NIDS based entirely on different overall performance levels. In this project, we focused on reviewing machine learning algorithms in order to incorporate NIDS.

Detection technique	Alarm Rate	Speed	Flexibility	Reliability	Scalability	Robustness
Signature	Low	High	Low	High	Low	Low
Anomaly	High	Low	High	Moderate	High	High

Table 1.2 Comparison detection methods

Chapter 2

Machine Learning

2.1 Machine Learning in NIDS

Machine learning (ML) is devoted to the creation of systems that can automatically analyze data and detect hidden patterns without being specifically programmed to do so. The learning styles that ML algorithms use, as well as the functional similarity of how they function, are used to mark them. Machine learning approaches are thought to be environmentally friendly because they improve detection rates, reduce false alarm rates, and reduce computing and communication costs. Machine learning methods may be classified as supervised, unsupervised, or semi-supervised.

2.2 ML based NIDS observation

Artificial Neural Networks (ANN), Support Vector Machines (SVM), Naive-Bayesian (NB), Random Forests (RF), self-organizing map (SOM), and other ML/DL methods have been used to advance NIDSs. Implemented an NIDS that was entirely based on a restricted Boltzmann machine (RBM) for feature reduction and a support vector machine (SVM) for classification. The system's accuracy is about 87 percent. To gather knowledge from training data, a network anomaly detection device was developed using discriminative RBM in conjunction with generative models with appropriate classification accuracy abilities.

Decision Tree (DT): The algorithm chooses the function with the highest knowledge gain as the root node, then calculates the ‘gini index' to find the best partition, and repeats the process until the required maximum depth is reached.

Random Forest (RF): A number of decision trees are created, each based on a different subset of the dataset, and the output of all the trees is then summed to obtain the algorithm's final result.

XGBoost: A decision tree is constructed by first using a subset of the data set to create a level 1 decision tree, and then using other subsets in order until the desired level is reached. After a certain amount, the algorithm could choose the best of several initial decision trees.

Support Vector Machine (SVM): By calculating the distances between the dataset's points, the best geometric separator between the dataset's classes can be found. After projecting the dataset points to another dimension, the kernel trick can be used by using kernel functions to calculate the distances between them.

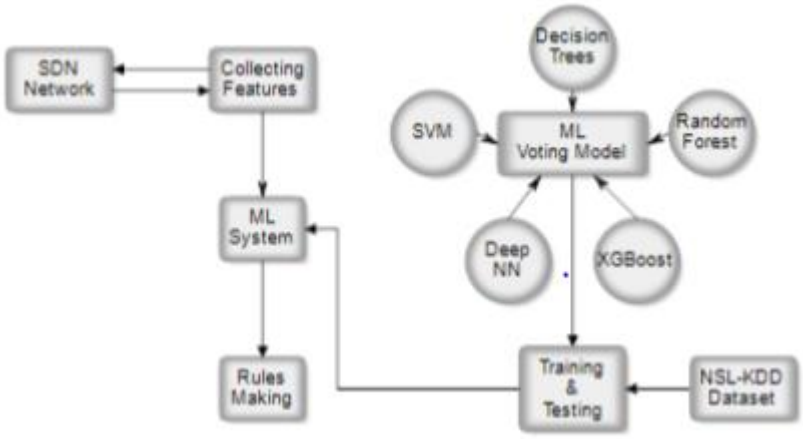

Figure 2.1 SDN based IDS architecture using ML

Chapter 3

Why SDN?

3.1 Software-defined networking based NIDS

The separation of control and data planes is one of the characteristics of the Software-Defined Networking (SDN) architecture, which simplifies packet forwarding. SDN's centralized controller allows for real-time input control and opens interfaces that allow for modular plug-in features. The centralized controller offers an abstract network view, as well as the ability to define tasks using APIs and increase network programmability. It can incorporate security devices into the network topology, resulting in improved accuracy, faster detection of security incidents, and easier management.

3.1.1 Why using SDN based system is helpful?

In terms of security compliance, virtual management, and Quality of Service, an SDN-based intrusion detection system that uses an ML/DL approach offers numerous benefits (QoS). SDN allows us to improve our network security while also allowing us to program network devices more easily and eliminating hardware dependence.

Simulation and emulation systems can be used to build an SDN network with software switch implementations and programmable features. One of the most widely used protocol standards for implementing the SDN definition in both hardware and a software environment is Open Flow. Other simulation tools include NS-2, Mininet, NS-3, and OMNeT++. The SDN controller, also known as a network operating system, is a

critical component of SDN networks. The SDN controller is in charge of concentrating communications with all programmable network components, resulting in a unified view of the network. Several SDN controllers, such as NOX and POX, are currently available. Figure, SDN-based NIDS architecture as depicted.

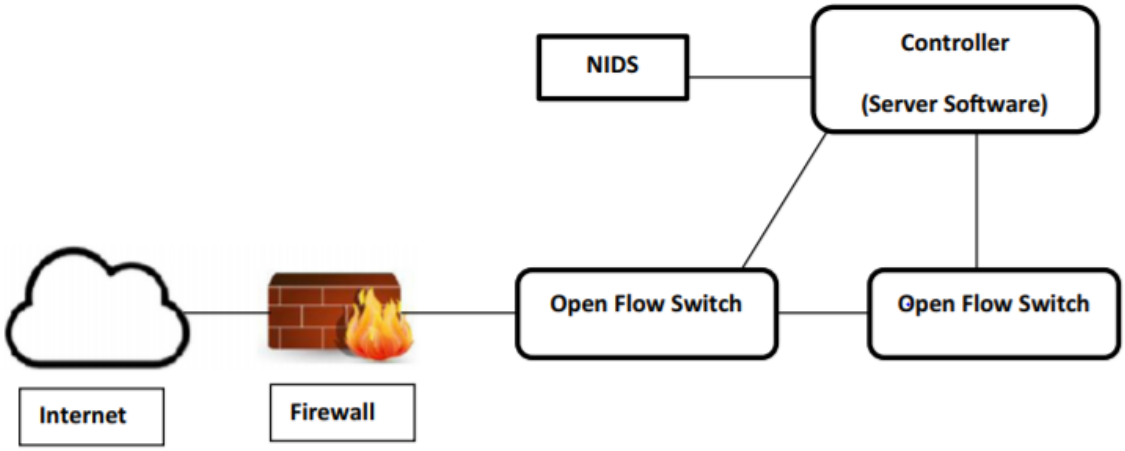

Figure 3.1 NIDS architecture

Chapter 4

Implementation

4.1 Working Environment and flow

- Operating System – Windows 10 Home
- Language – Python 3.8
- Experiments done –Google Colab/ Kaggle
- Nvidia K80 GPUs

4.2 Dataset

The KDD Cup is the world's most prestigious data mining competition. The NSL-KDD Dataset was created to address some of the issues with the KDD Cup 1999 Dataset. It is still a decent guide to compare the NIDS models, despite the fact that it is very old and not a perfect representation of actual real networks. Many researchers have used it in the past to assess the efficiency of NIDS, so there are important performance assessment data to compare. The KDDTrain+ Dataset contains 125,973 network traffic samples, while the KDDTest+ Dataset contains 22,554 network traffic samples. There are forty-one features in each traffic sample, which are divided into three categories: basic features, content-based features, and traffic-based features. The attacks in the dataset are divided into four groups based on their characteristics.

Within the data set exists 4 different classes of attacks: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). A brief description of each attack can be seen below:

- A denial-of-service (DoS) attack attempts to halt traffic flow to and from the target system. The IDS receives an unusual amount of traffic that it cannot accommodate, and it shuts down to protect itself. This makes it impossible for regular traffic to access a network. An online store could experience a surge in online orders on a major selling day, and since the network can't accommodate all of the requests, it will shut down, preventing paying customers from making purchases. In the data collection, this is the most common attack.

- A probe or surveillance attack attempts to obtain data from a network. The aim is to impersonate a thief and steal vital information, such as client personal information or banking information.

- U2R is a form of attack that begins with a regular user account and attempts to gain super-user access to the device or network (root). The attacker tries to obtain root privileges/access by exploiting device vulnerabilities.

- R2L is a method of gaining local access to a remote computer. An attacker that does not have local access to the system/network attempts to "hack" their way in.

4.2.1 Data Extraction

The act or method of extracting data from data sources for further processing or storage is known as data extraction. Prior to export to the next stage of the data workflow, data transformation and likely metadata addition are normally performed after import into the intermediate extracting method.

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_s
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07

Figure 4.1 Data Extraction

4.3 Data Transformation and Classification

The first transformations that we'll want to do are around the attack field. We'll start by adding a column that encodes 'normal' values as 0 and any other value as 1. We will use this as our classifier for simple binary models that identifies any attack.

We'll classify each of the attacks according to attack type for a more granular prediction model.

Denial of Service attacks	Probe attacks	Privilege escalation attacks	Remote access attacks
▪ apache2 ▪ back ▪ land ▪ neptune ▪ mailbomb ▪ pod ▪ processtable ▪ smurf ▪ teardrop ▪ udpstorm ▪ worm	▪ ipsweep ▪ mscan ▪ nmap ▪ portsweep ▪ saint ▪ satan	▪ buffer_overflow ▪ loadmdoule ▪ perl ▪ ps ▪ rootkit ▪ sqlattack ▪ xterm	▪ ftp_write ▪ guess_passwd ▪ http_tunnel ▪ imap ▪ multihop ▪ named ▪ phf ▪ sendmail ▪ snmpgetattack ▪ spy

Table 4.1 Types of attacks

4.3.1 Data Profiling

Some initial investigations of what we have in the set. First is a simple table of attack by protocol. In network traffic analysis protocol is a simple tool to create some initial buckets to categorize our data. 'Normal' is left in the set at this point as a benchmark.

That helps us see that most attacks are going to target a specific protocol. There are several (Satan, nmap, ipsweep) that are cross-protocol attacks.

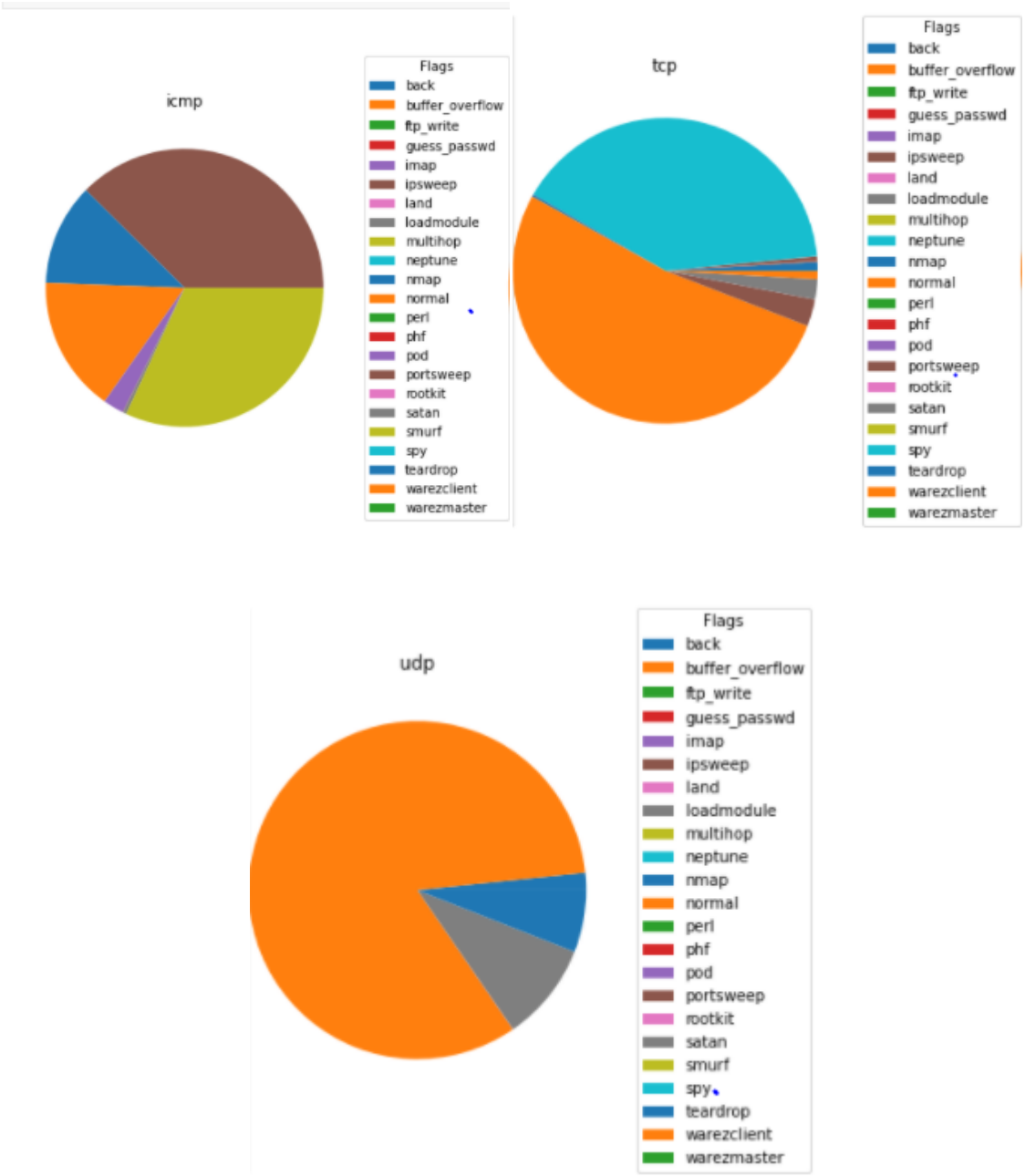

Figure 4.2 Data Profiling

4.3.2 Feature Engineering/ Model Fitting

Checking behavior of dataset based on flag classification.

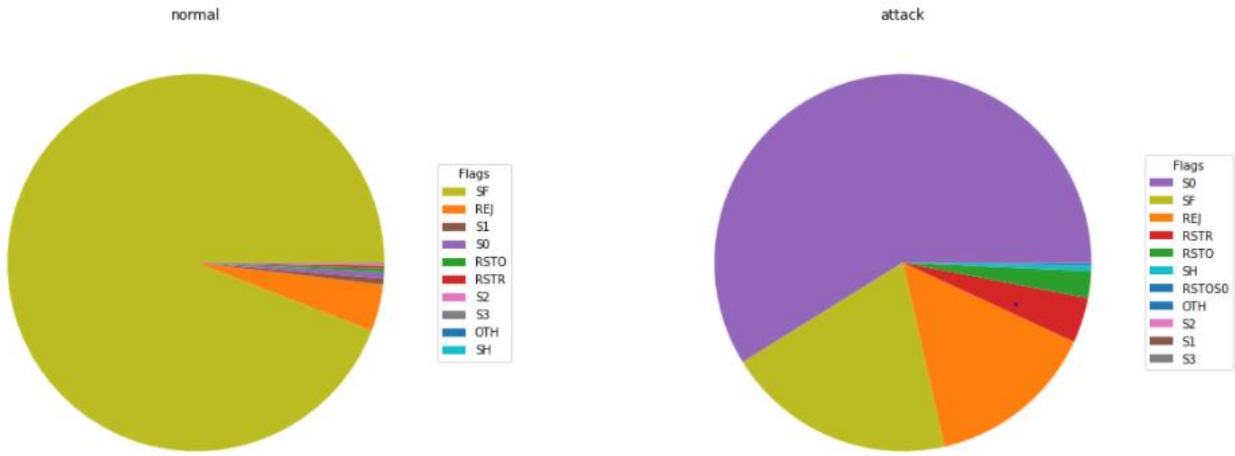

Figure 4.3 Flag based data profiling

Classification using:-

RandomForestClassifier

Logistic Regression

KNeighborsClassifier

```
binary_model = RandomForestClassifier()
binary_model.fit(binary_train_X, binary_train_y)
binary_predictions = binary_model.predict(binary_val_X)


base_rf_score = accuracy_score(binary_predictions,binary_val_y).
base_rf_score
```

```
models = [
    RandomForestClassifier(),
    LogisticRegression(max_iter=250),
    KNeighborsClassifier(),
]

model_comps = []


for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_val_score(model, binary_train_X, binary_train_y, scoring='accurac
y')
    for count, accuracy in enumerate(accuracies):
        model_comps.append((model_name, count, accuracy))
```

Figure 4.4 ML Classification

Chapter 5

Results

5.1 Analysis our Prediction

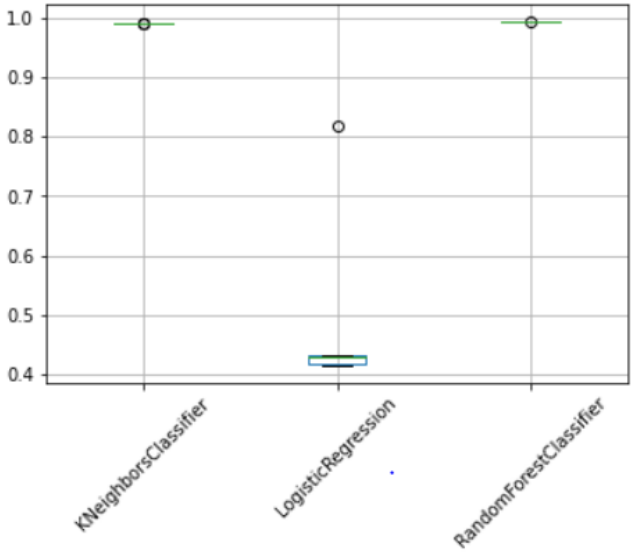

Figure 5.1 Analysis of accuracy

Here as we can see that we achieved nearly 98% accuracy with KNeighborsClassifier and RandonForest but around 80% accuracy with Logistic regression.

What we find is some inconsistency across the models. The random forest and K-nearest neighbors are tight groupings with solid performance. Our logistic regression didn't perform as well. That may be in part because we didn't do sufficient preprocessing on our data to shape it into a form optimized for that model.

5.1.1 Confusion Matrix

To further analyze the results, we used confusion matrix to see the false positives and false negatives.

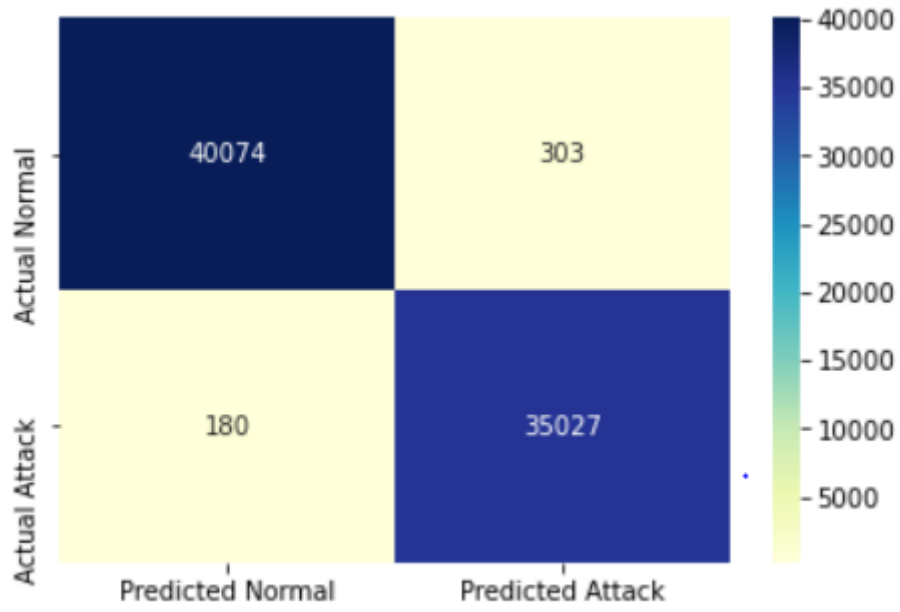

Figure 5.2 Confusion matrix

We see a lot of false positives (normal traffic that got flagged as an attack) and false negatives (attack traffic that got flagged as normal) there.

So we further processed the data set and used the standard deviation method where we drop the cells whose standard deviation is 0. Then we again created the confusion matrix.

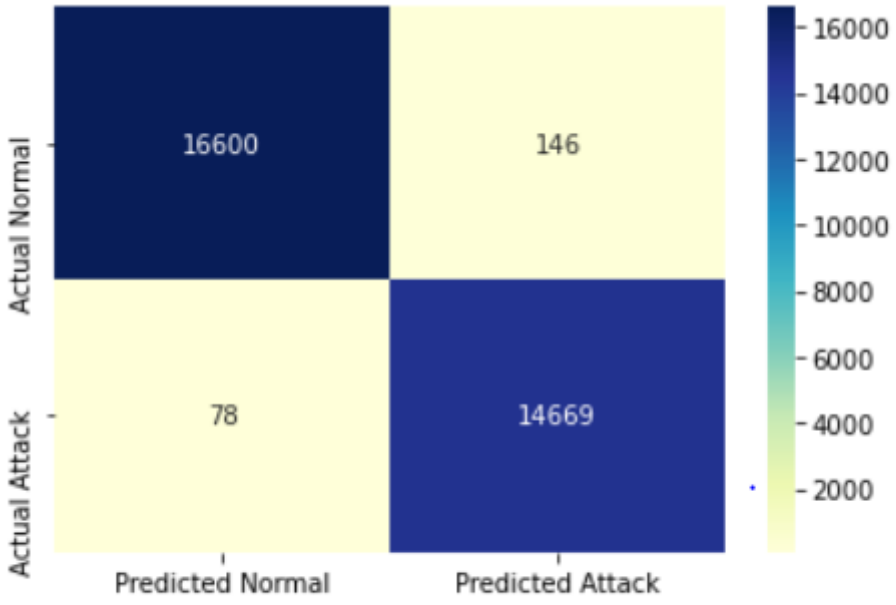

Figure 5.3 Updated Confusion matrix

In the updated Confusion matrix we can see that the false positive and false negative is reduced by one-third.

5.1.2 Reducing the number of false positive

Initially as we can see that the false positive of Neptune and Satan were high around 100 and 23 which were later reduced to below 45 and 10.

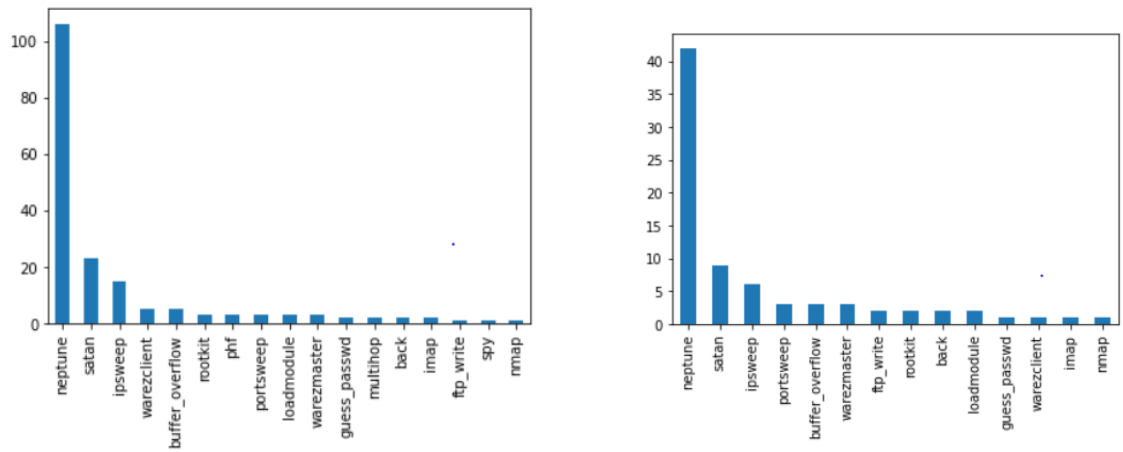

Figure 5.4 False Positive graphs

5.2 Using the model on unseen dataset

```
full_model = RandomForestClassifier(random_state=1)
full_model.fit(to_fit, binary_y)
full_predictions = full_model.predict(test_set)

# get the score
full_score = accuracy_score(full_predictions,test_binary_y)
full_score
```

```
0.8035753892560884
```

Figure 5.5 Final Accuracy

We get around 80% accuracy on unseen dataset.

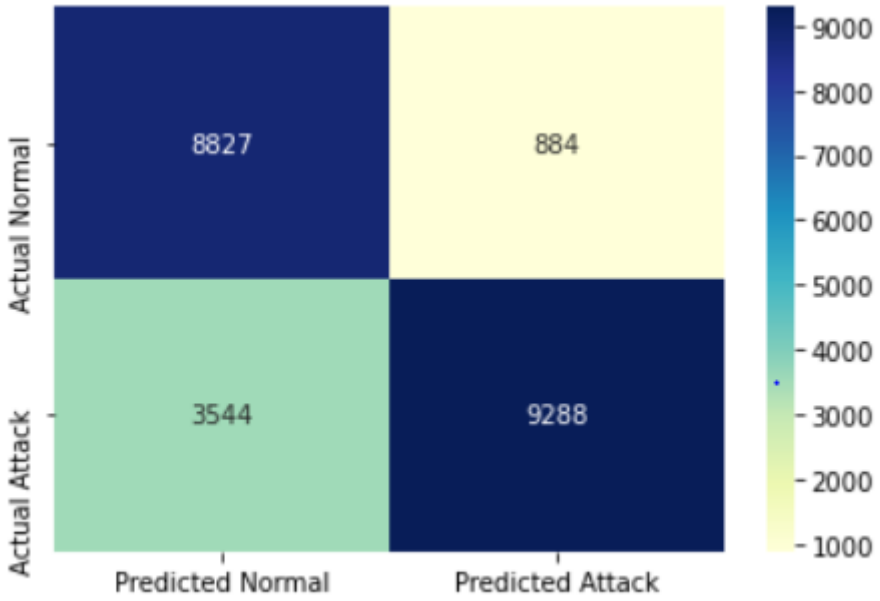

Figure 5.6 Confusion matrixes for unseen dataset

Decision Tree	79%
Random Forest	80.35%
XGBoost	78.3%
SVM	73%
Logistic regression	75%

Table 5.1 Accuracy achieved

Chapter 6

Conclusion

6.1 Summary

We looked at the new field of Software-Defined Networking and gave an overview of programmable networks (SDN). We also discussed how to use machine learning and deep learning to detect intrusions. We highlighted software-defined networking (SDN) technology as a tool for detecting vulnerabilities and monitoring networks using machine learning and deep learning techniques.

6.2 Future Scope

Developing a feature selection system with classifiers that reduces the dataset's dimensions is a never-ending job. Another area of study is using DL techniques to identify appropriate datasets. A potential future path and a difficult task is to develop a centralized SDN controller that can track and enforce real-time intrusion detection in high-speed networks.