PREVENTING INSIDER THREADS IN NETWORKS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

AKILANDESWARI A	**312819205005**
KAVIYA DHARSHINI S	**312819205012**
MANOJ KUMAR N	**312819205018**

In partial fulfilment for the award of the degree

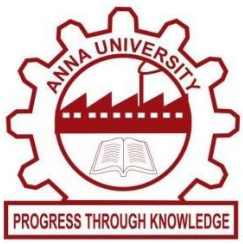
of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

AGNI COLLEGE OF TECHNOLOGY, THALAMBUR



ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **PREVENTING INSIDER THREADS IN NETWORKS USING MACHINE LEARNING** is the bonafide work of **AKILANDESWARI A (312819205005), KAVIYA DHARSHINI S (312819205012), MANOJ KUMAR N (312819205018)** who carried out the project work under my supervision.

SIGNATURE
Dr. S. GEERTHIK, M.E, Ph.D.,
ASSOCIATE PROFESSOR,
HEAD OF THE DEPARTMENT,
Department of Information Technology.
Agni College Of Technology,
Thalambur, Chennai – 6000130.

SIGNATURE
DR.G.A. SENTHIL, M.TECH, PH.D.,
ASSOCIATE PROFESSOR,
SUPERVISOR,
Department of Information Technology.
Agni College Of Technology,
Thalambur, Chennai – 6000130.

Submitted for the project viva held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to the management of "**AGNI COLLEGE OF TECHNOLOGY"** and would like to thank our respectedPrincipal **Dr. SRINIVASAN ALAVANDAR, M.E., Ph.D.,** for his words of inspiration and for providing necessary facilities to carry out our project work successfully.

We are immensely thankful to **Dr. S. GEERTHIK, M.E., Ph.D.,** Associate Professor, Head of the Department, Information Technology for his words of wisdom and his constant source of inspiration.

I would like to offer our heartfelt thanks to our guide **DR.G.A. SENTHIL, M.TECH, PH.D.,** Assistant Professor, Department of Information Technology who molded us accordingly and gave valuable suggestions for completing our project work successfully.

We extended our warmest thanks to all the faculty members of our Department for their assistance and we also thank all our friends who helped us in bringing out our project in good shape and form.

Finally, we express our sincere benevolence to our beloved parents for their perpetual encouragement and support in all endeavors.

ABSTRACT

IoT networks have become an increasingly valuable target of malicious attacks due to the services increased amount of valuable user data they contain where cyber-attacks attempt to compromise the security principles of confidentiality, integrity, and availability. In response, network intrusion detection systems (IDS), or systems that monitor and detect cyber-attack patterns over networking environments based on machine learning algorithms and models, have been developed to detect suspicious network activity. These systems monitor network traffic for suspicious activities and issue alerts in the case of detected attack types. Machine learning (ML) algorithms are being investigated as potential IDS frameworks as current or traditional IDS capabilities have concerns. The increasing complexity of computer networks has made it difficult to detect and prevent network attacks using traditional methods. Software-Defined Networking (SDN) has emerged as a promising technology that enables the centralized management and control of network traffic. In this project, we investigated the use of machine learning techniques in conjunction with SDN for predicting network attacks. We developed a system that uses SDN to capture network traffic and extract relevant features, which were then used to train and test a variety of machine learning models. This approach has important implications for network security, as it provides a more efficient and effective way to detect and prevent cyber-attacks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SDN	**SOFTWARE DEFINED NETWORK**
IDS	**INTRUSION DETECTION SYSTEM**
MLP	**MULTI LAYER PERCEPTRON**
UML	**UNIFIED MODELING LANGUAGE**
ML	**MACHINE LEARNING**

CHAPTER 1

1.1 AIM

SDN (Software-Defined Networking) with machine learning can be used for network attack prediction, aiming to detect and prevent potential cyber threats. Machine learning algorithms can be used to analyze network traffic and detect patterns that may indicate an ongoing or potential attack. These algorithms can be trained using historical data to identify common attack signatures and behaviors. SDN can provide a centralized control plane for network management, making it easier to detect and respond to potential attacks. SDN controllers can use machine learning algorithms to analyze network traffic in real-time, looking for anomalies or patterns that could indicate an attack. By combining SDN and machine learning, network administrators can create a more proactive approach to network security, identifying and responding to potential attacks before they can cause significant damage. Some of the techniques that can be used to implement network attack prediction using SDN with machine learning are:

Anomaly detection: Machine learning algorithms can be used to identify anomalous patterns in network traffic, such as unusual spikes in traffic or unexpected connections. Anomalous traffic patterns can be a sign of an ongoing or potential attack, and the system can be configured to raise an alert or take preventive measures in such situations. Signature-based detection: Machine learning algorithms can be trained to identify known attack signatures or patterns, which can help in detecting known types of attacks. Signature-based detection can be more efficient than anomaly detection since it is more specific and can identify attacks that are known to have occurred before.

1.2 INTRODUCTION

This model helps primary care properly identify biological anomalies in the biomedical field, where machine learning already has a substantial impact in a

number of fields. Let's say there is a person with respiratory or neurological issues, and we are unsure if that individual has a neurological mutation. Now that we have this model, we can use it to anticipate events with the best degree of accuracy. It is difficult to assess and determine the precision of a genetic characteristic. Based on birth abnormalities, maternal genes, paternal genes inherited from the father, maternal genes, maternal genes on the mother's side, and blood test results, we utilize this model to forecast genetic illnesses. Both numerical and category target variables can be used with this approach. In this research, we provide an algorithm that learns from previous data and makes predictions. In order to successfully recruit the data set, this model aims to select the best fit model. Then, we forecast future data using the previous data with the highest accuracy. More importantly, our model could produce accurate forecasts and wise conclusions when compared to traditional methods.

1.3 OBJECTIVE

The objective for Intrusion Detection System (IDS) using SDN (Software Defined Networking), MLP (Multilayer Perceptron), Decision Tree, and XGBoost can be to improve the accuracy and efficiency of detecting network attacks and anomalies.

Specifically, the objectives could include:

Developing an IDS using SDN that allows for centralized network management and control, making it easier to detect and respond to network threats.

Using MLP, Decision Tree, and XGBoost machine learning algorithms to improve the accuracy of detecting network attacks and anomalies by analyzing network traffic and identifying patterns that indicate malicious behavior.

CHAPTER 2

LITERATURE SURVEY

[1] Project Title: An Intrusion Detection Model

Authors: D.E. Denning

A model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse is described. The model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model includes profiles for representing the behavior of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behavior from audit records and for detecting anomalous behavior

[2] Project Title: Data Mining Approaches for Intrusion Detection

Authors: Wanke Lee and Salvatore J.Stolfo

In this paper we discuss our research in developing general and systematic methods for intrusion detection. The key ideas are to use data mining techniques to discover consistent and useful patterns of system features that describe program and user behavior, and use the set of relevant system features to compute (inductively learned) classifiers that can recognize anomalies and known intrusions. Using experiments on the send mail system call data and the network tcp dump data, we demonstrate that we can construct concise and accurate classifiers to detect anomalies. We provide an overview on two general data mining algorithms that we have implemented: the association rules algorithm and the frequent episodes algorithm.

[3] Project Title: Is the future Web more insecure? Distractions and solutions of new-old security issues and measures

Authors: Federico Maggi and Stefano Zanero

The world of information and communication technology is experiencing changes that, regardless of some skepticism, are bringing to life the concept of "utility computing". Thenostalgics observed a parallel between the emerging paradigm of cloud computing and the traditional time-sharing era, depicting clouds as the modern reincarnation of main-frames available on a pay-per-use basis, and equipped with virtual, elastic, disks-as-a-service that replace the old physical disks with quotas.

[4] Project Title: Evaluating Intrusion Detection Systems

Authors: Richard P. Lippmann, David J. Fried and Isaac Graf

This work was sponsored by Rome Laboratory and the Department of Defense Advanced Research Projects Agency under Air Force Contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force. An intrusion detection evaluation test bed was developed which generated normal traffic similar to that on a government site containing 100's of users on 1000's of hosts. More than 300 instances of 38 different automated attacks were launched against victim UNIX hosts in seven weeks of training data and two weeks of test data. Six research groups participated in a blind evaluation and results were analyzed for probe, denial-of-service (DoS), remote-to-local (R2L), and user to root (U2R) attacks.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing system

Preventing insider threats in networks using SDN and machine learning is a relatively new field, and there are not many existing systems that provide a comprehensive solution. However, some systems are currently available that can help in preventing insider threats in networks.One such system is the Security Information and Event Management (SIEM) system, which provides real-time monitoring and analysis of network data to identify suspicious activities. It uses machine learning algorithms to detect anomalous behavior and provides alerts to security personnel. SIEM systems can also integrate with SDN controllers to block or quarantine suspicious devices or users.Another system is the User and Entity Behavior Analytics (UEBA) system, which uses machine learning algorithms to analyze user behavior patterns and identify anomalies that might indicate insider threats. UEBA systems can provide insights into user activity, such as access to sensitive data or unauthorized login attempts, and provide alerts to security personnel.SDN controllers can also provide access control mechanisms that limit user access to sensitive data and network resources. These controllers can integrate with identity and access management (IAM) solutions to verify the identity of users and ensure that only authorized users can access sensitive data.In summary, while there are not many existing systems that provide a comprehensive solution for preventing insider threats in networks using SDN and machine learning, some systems such as SIEM, UEBA, and SDN controllers can be used in combination to provide a robust security posture.

3.2 Proposed system

A proposed system for preventing insider threats in networks using SDN (Software-Defined Networking) and machine learning would involve the following components:

Network Infrastructure: The system would require a compatible network infrastructure that supports SDN, including switches and routers that can be controlled by an SDN controller.

SDN Controller: The system would require an SDN controller that can manage the network infrastructure and apply access control policies. The controller would also integrate with machine learning models to detect anomalous behavior and provide alerts to security personnel.

Machine Learning Models: The system would use machine learning models to detect anomalous behavior and identify potential insider threats. These models would analyze user behavior patterns, such as access to sensitive data or unusual login times, and provide alerts to security personnel when anomalous behavior is detected.

Access Control Policies: The system would implement access control policies that limit user access to sensitive data and network resources. These policies would be managed by the SDN controller and integrated with identity and access management solutions to verify user identity.

User Behavior Analytics: The system would use user behavior analytics to identify patterns and anomalies in user behavior. These analytics would be used to train machine learning models and identify potential insider threats.

Real-Time Monitoring: The system would provide real-time monitoring of network traffic and user activity to detect insider threats as soon as possible. This monitoring would be performed by the machine learning models and SDN controller.

Alerting System: The system would provide an alerting system that notifies security personnel when anomalous behavior is detected. These alerts would include details about the detected behavior and recommended actions to take.In summary, a proposed system for preventing insider threats in networks using SDN and machine learning would include a compatible network infrastructure, an SDN controller, machine learning models, access control policies, user behavior analytics, real-time monitoring, and an alerting system.

3.3 Advantages of proposed system

There are several advantages to preventing insider threats in networks using SDN(Software-Defined Networking) and machine learning:

Increased Security: SDN and machine learning can work together to provide a more comprehensive and proactive security posture.

Quick Response: The combination of SDN and machine learning enables organizations to respond quickly to potential insider threats.

Reduced False Positives: Machine learning models can reduce false positives by accurately identifying anomalous behavior and filtering out benign activities.

Efficient Resource Allocation: SDN allows organizations to efficiently allocate network resources by controlling network traffic and access.

Scalability: SDN and machine learning can scale to accommodate the changing needs of organizations.

Improved Compliance: By implementing access control policies and monitoring user behavior, organizations can ensure compliance with regulatory requirements and industry standards.

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

- Operating system: Windows 11
- Coding Language: PYTHON
- IDE: SPYDER

4.2 HARDWARE REQUIREMENTS

- Processor: Intel(R) Core(TM)
- Speed: 1.6 GHz and Above
- RAM: 4 GB and Above
- Hard Disk: 120 GB

CHAPTER 5
SYSTEM DESIGN

5.1 UML DIAGRAMS:

5.1.1 Introduction

Unified Modeling Language (UML) is a standardized general- purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development. There are given as below:

- Sequence diagram
- Use case diagram
- Class diagram

5.2 Sequence Diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct ofMessage Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

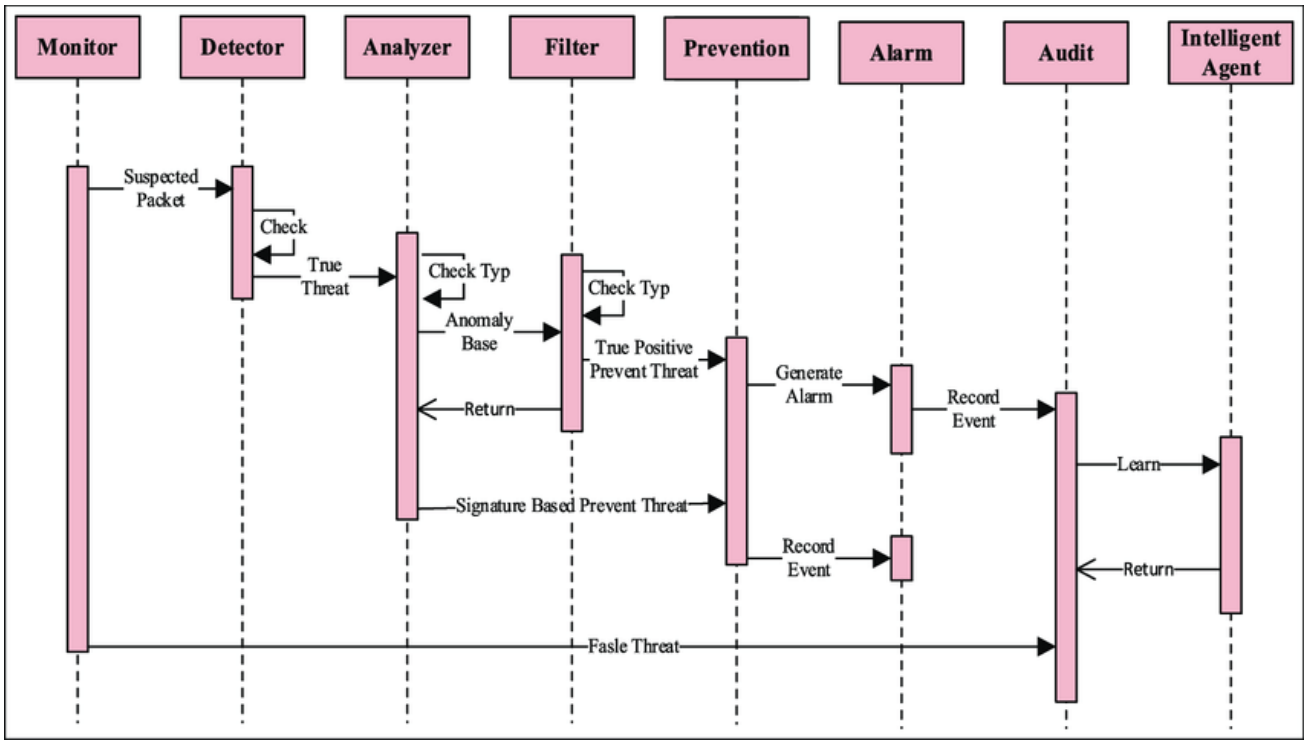


Fig 5.2 .Sequence diagram

5.3 Use Case Diagram:

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case:

A use case describes a sequence of actions that provided somethingof measurable value to an actor and is drawn as a horizontal ellipse.

Actor:

An actor is a person, organization or external system that plays a rolein one or more interaction with the system.

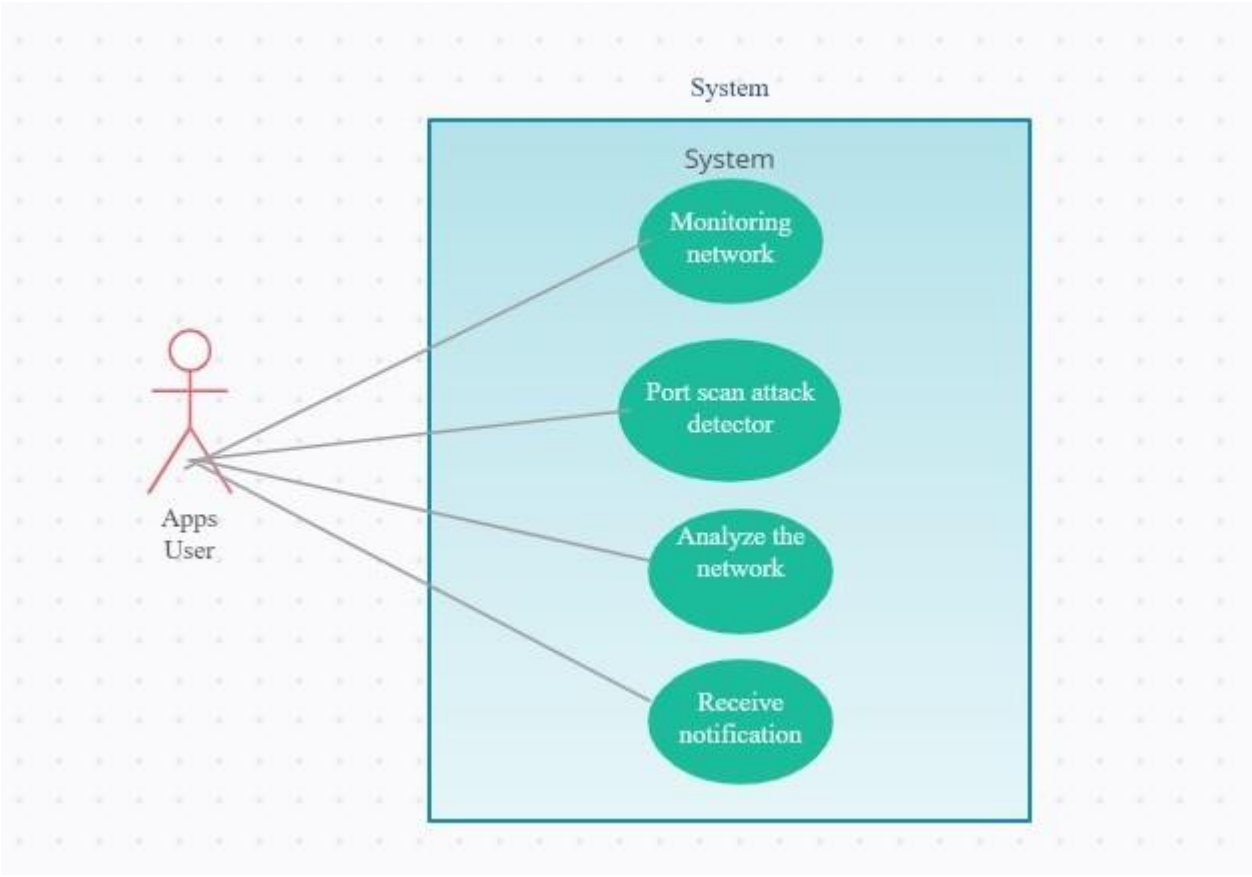


Fig.5.2 Use case diagram

5.4 Class Diagram:

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

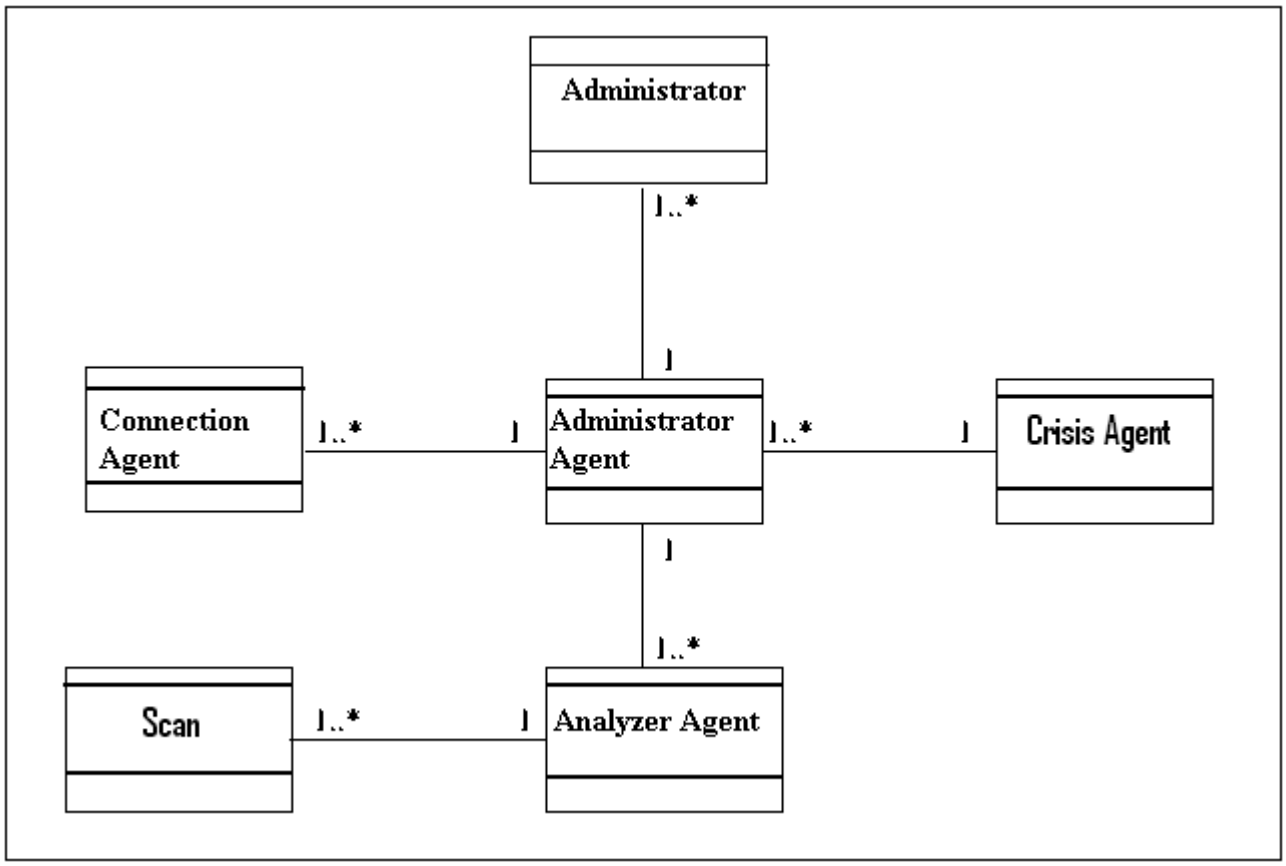


Fig 5.3 Class Diagram

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 MODULE DESCRIPTION

Modules:

1) SDN

2) Multi Layer Perceptron

3) XG Boost

4) Decision Tree

5) Admin

SDN:

SDN (Software-Defined Networking) is a network architecture that separates the control plane from the data plane in network switches and routers. This separation allows for the centralized management and configuration of network devices, rather than having to configure each device individually.

SDN networks consist of three main components: the application layer, the control layer, and the infrastructure layer.

Multi Layer Perceptron:

Multilayer Perceptron (MLP) is a type of artificial neural network (ANN) that is widely used for supervised learning tasks, such as classification and regression. It consists of multiple layers of interconnected nodes, or neurons, where each node in one layer is connected to every node in the next layer.

The MLP algorithm is a feedforward neural network, meaning that data flows through the network in one direction, from input layer to output layer, without any feedback connections.

XG Boost:

XG Boost is a popular open-source machine learning library that implements gradient boosting algorithms. Gradient boosting is a technique for building ensemble models by iteratively adding weak learners to the ensemble. The resulting model is a weighted sum of the weak learners. XG Boost is known for its high performance and scalability, making it a popular choice for large-scale machine learning problems. It supports both regression and classification tasks, and provides a range of options for customizing the model's behavior.

Decision Tree:

A decision tree is a machine learning algorithm that builds a tree-like model of decisions and their possible consequences. It is a popular supervised learning algorithm used for both classification and regression tasks.

In a decision tree, the root node represents the entire dataset, and it is split into sub-nodes based on the value of a feature that is most informative for the classification task. Each sub-node is then further split into smaller sub-nodes, and this process is repeated recursively until a stopping criterion is met. The stopping criterion could be a pre-specified maximum depth of the tree or a minimum number of samples required to make a split.

ADMIN:

The administrative module allows the IDS administrator to configure various aspects of the IDS system, such as network settings, logging options, and notification settings. This module enables the IDS administrator to manage alerts generated by the IDS, such as filtering or prioritizing alerts, assigning alerts to specific users, or sending alerts to external systems.

6.2 PROPOSED SYSTEM METHODOLOGY

Data Collection: The first step in the proposed system methodology is to collect data from various sources within the organization's network. This data can include network traffic data, user behavior data, and system log data.

Data Preprocessing: The collected data needs to be preprocessed to remove any irrelevant or redundant data. The preprocessing step involves data cleaning, data normalization, and data transformation.

Feature Selection: The next step is to select the relevant features from the preprocessed data. The selected features should have a high correlation with insider threat activities.

Machine Learning Model Development: Once the relevant features are selected, the next step is to develop a machine learning model to detect insider threats. The model should be trained on a labeled dataset, where each data point is labeled as either normal or an insider threat.

Model Integration with SDN: The developed machine learning model should be integrated with the SDN controller to monitor network traffic in real-time. The SDN controller should be programmed to send alerts to the security team when suspicious activity is detected.

6.2.1. ARCHITECTURE DIAGRAM

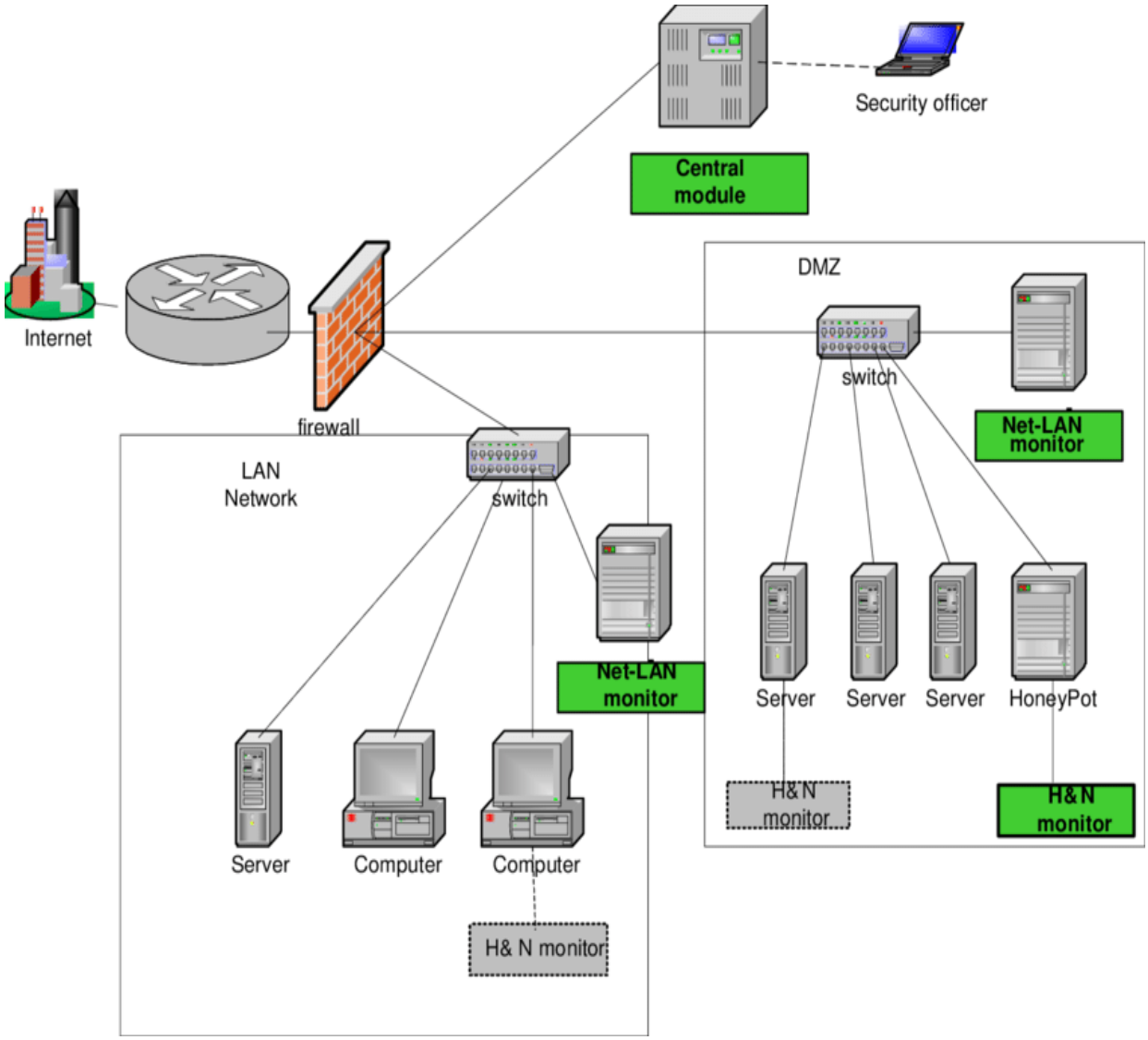


Fig.6.1 Architecture diagram

6.3 PREVENTING INSIDER THREADS IN NETWORKS USING MACHINE LEARNING

The project of "Preventing Insider Threats in Networks using SDN and Machine Learning" aims to develop a system that can detect and prevent insider threats in computer networks. Insider threats refer to any malicious or unintentional activities carried out by authorized users, such as employees or contractors, that could harm the organization's assets, including information, systems, and networks.

The proposed solution leverages Software-Defined Networking (SDN) and Machine Learning (ML) techniques to identify and mitigate insider threats. SDN provides a centralized control plane that allows network administrators to configure and manage network devices programmatically, enabling quick response to network events and policy enforcement. Machine learning algorithms can be used to analyze network traffic and user behavior, detect anomalies, and predict potential insider threats.

The system will be developed in several stages, including data collection, pre-processing, feature extraction, and machine learning model training. Data will be collected from various sources, including network logs, user access logs, and application logs. The pre-processing stage involves cleaning and normalizing the data, and the feature extraction stage will identify relevant features that can be used to train machine learning models.

The ML models will be trained using various algorithms, including supervised and unsupervised learning, to classify user behavior as either normal or anomalous. Once trained, the models will be deployed in the SDN controller to monitor network traffic continuously.

Whenever an insider threat is detected, the system will trigger an alert and take appropriate action, such as quarantining the user or device.

The project aims to improve the security of computer networks by preventing insider threats, which are becoming more prevalent in today's interconnected world. By using SDN and machine learning, the system can provide real-time threat detection and prevention, reducing the risk of data breaches and network disruptions caused by insider attacks.

CHAPTER7

CODING AND TESTING

7.1 CODING

Once the design aspect of the system is finalizes the system enters intothe coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

7.2 TEST PROCEDURE SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specificationhas been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not haveany logic faults in the design is detected before coding commences, otherwisethe cost of fixing the faults will be considerably higher as reflected. Detection ofdesign faults can be achieved by means of inspection as well as walkthrough.

7.3 TEST DATA AND OUTPUT

7.3.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.3.2USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

No	Test Scenario	Expected Result	Test Result
1	Username is correct. Password is incorrect.	Username and Password is incorrect.	Username and Password is incorrect.
2	Username is incorrect. Password is correct.	Username and Password is incorrect.	Username and Password is incorrect.
3	Username is empty. Password is correct.	Username is required.	Username is required.
4	Username is correct. Password is empty.	Password is required.	Password is required
5	Both Username	Username and Password is	Username and Password is

	and Password is incorrect.	incorrect.	incorrect.
6	Both Username and Password is empty.	Username and Password is required.	Username and Password is required.
7	Both Username and Password is correct.	Login Successful.	Login Successful.

CHAPTER 8

SOFTWARE DESCRIPTION

8.1 PYTHON

8.1.1 INTRODUCTION TO PYTHON

About Python:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatics in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989; Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coughlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project. Guido van Rossum has since then withdrawn his nomination for the 2020 Steering council.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

8.2 HTML

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produce web pages that included text, graphics and pointer to other web pages (Hyperlinks).

HTML is not a programming language, but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext

and adapted to the Web. The idea behind Hypertext one point to another point.

We can navigate through the information based on out interest and preference. A markup language is simply a series of items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

8.3 CSS

CSS stands for Cascading Style Sheets, which is a stylesheet language used to describe the presentation and style of web pages written in HTML or XML. CSS allows web developers to separate the content of a web page from its presentation, making it easier to manage and modify the design and layout of a website.

8.4 JAVA SCRIPT

JavaScript is a compact, object-based scripting language for developing client and server internet applications. Netscape Navigator 2.0 interprets JavaScript statements embedded directly in an HTML page. And Livewire enables you to create server-based applications similar to common gateway interface (CGI) programs.

CHAPTER 9

CODE, RESULTS AND CONCLUSION

9.1 CODE

HTML PAGES:

LOGIN PAGE:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Login Page</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div class="container">
      <h1>Login</h1>
      <form id="login-form" action="#" method="post">
        <div class="form-group">
          <label for="username">Username:</label>
          <input type="text" id="username" name="username" required>
        </div>
        <div class="form-group">
          <label for="password">Password:</label>
          <input type="password" id="password" name="password" required>
        </div>
        <div class="form-group">
```

```
        <button type="submit">Login</button>
      </div>
    </form>
  </div>
  <script src="script.js"></script>
 </body>
</html>
```

style.css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}

.container {
  margin: 50px auto;
  max-width: 500px;
  padding: 20px;
  border: 1px solid #ccc;
  box-shadow: 0 0 10px #ccc;
}

h1 {
  text-align: center;
}

.form-group {
  margin-bottom: 20px;
}
```

```
label {
  display: block;
  margin-bottom: 5px;
}

input {
  display: block;
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  font-size: 16px;
}

button[type="submit"] {
  display: block;
  width: 100%;
  padding: 10px;
  border: none;
  border-radius: 5px;
  background-color: #333;
  color: #fff;
  font-size: 16px;
  cursor: pointer;
}

button[type="submit"]:hover {
  background-color: #444;
}
```

Script.js

```
const form = document.getElementById('login-form');
form.addEventListener('submit', function(event) {
  event.preventDefault();
  const username = form.elements.username.value;
  const password = form.elements.password.value;
  // Add login logic here
  console.log('Username:', username);
  console.log('Password:', password);
});
```

HOME PAGE:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>IDS Homepage</title>
    <link rel="stylesheet" type="text/css" href="style1.css">
  </head>
  <body>
    <header>
      <h1>Intrusion Detection System</h1>
    </header>
    <nav>
      <ul>
        <li><a href="#algorithm1">Algorithm 1</a></li>
        <li><a href="#algorithm2">Algorithm 2</a></li>
        <li><a href="#algorithm3">Algorithm 3</a></li>
        <li><a href="#algorithm4">Algorithm 4</a></li>
```

```
    </ul>
  </nav>
  <main>
    <section id="algorithm1">
      <h2>Algorithm 1</h2>
      <p>This is a description of algorithm 1.</p>
      <p>Here's some sample code:</p>
      <pre><code>// JavaScript code for algorithm 1
function algorithm1(input) {
  // Code goes here
}</code></pre>
    </section>
    <section id="algorithm2">
      <h2>Algorithm 2</h2>
      <p>This is a description of algorithm 2.</p>
      <p>Here's some sample code:</p>
      <pre><code>// JavaScript code for algorithm 2
function algorithm2(input) {
  // Code goes here
}</code></pre>
    </section>
    <section id="algorithm3">
      <h2>Algorithm 3</h2>
      <p>This is a description of algorithm 3.</p>
      <p>Here's some sample code:</p>
      <pre><code>// JavaScript code for algorithm 3
function algorithm3(input) {
  // Code goes here
}</code></pre>
    </section>
```

```
      <section id="algorithm4">
        <h2>Algorithm 4</h2>
        <p>This is a description of algorithm 4.</p>
        <p>Here's some sample code:</p>
        <pre><code>// JavaScript code for algorithm 4
function algorithm4(input) {
  // Code goes here
}</code></pre>
      </section>
    </main>
    <footer>
      <p>Copyright © 2023 IDS</p>
    </footer>
    <script src="script1.js"></script>
  </body>
</html>
```

Style1.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  header {
    background-color: #333;
    color: #fff;
    padding: 20px;
  }
```

```
header h1 {
  margin: 0;
}

nav {
  background-color: #444;
  color: #fff;
  padding: 10px;
}

nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
}

nav li {
  display: inline-block;
  margin-right: 10px;
}

nav a {
  color: #fff;
  text-decoration: none;
}

main {
  padding: 20px;
}
```

```
 section {
   border: 1px solid #ccc;
   margin-bottom: 20px;
   padding: 20px;
 }

 section h2 {
   margin-top: 0;
 }
```

script1.js

```
document.querySelectorAll('a[href^="#"]').forEach(anchor => {
   anchor.addEventListener('click', function (e) {
      e.preventDefault();
      document.querySelector(this.getAttribute('href')).scrollIntoView({
         behavior: 'smooth'
      });
   });
});

// Toggle active class on navigation links
const sections = document.querySelectorAll('section');
const navLinks = document.querySelectorAll('nav a');

window.addEventListener('scroll', () => {
   let current = '';
   sections.forEach(section => {
      const sectionTop = section.offsetTop;
      const sectionHeight = section.clientHeight;
      if (pageYOffset >= (sectionTop - sectionHeight / 3)) {
```

```
            current = section.getAttribute('id');
        }
    });

    navLinks.forEach(link => {
        link.classList.remove('active');
        if (link.getAttribute('href').substring(1) === current) {
            link.classList.add('active');
        }
    });
});
```

ADMIN

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
get_ipython().run_line_magic('matplotlib', 'inline')

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, cross_validate, StratifiedKFold
from sklearn.metrics import precision_recall_curve, precision_score, recall_score, f1_score, accuracy_score
from sklearn.metrics import roc_curve, auc, roc_auc_score, confusion_matrix, classification_report
```

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets, ensemble, model_selection
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

initial_data = pd.read_csv(r'C:\Users\lenovo\Documents\IoT-Network-Intrusion-Detection-and-Classification-using-Explainable-XAI-Machine-Learning-main (1)\UNSW_NB15_training.csv')

initial_data.head(n=5)

initial_data.info()

initial_data.isnull().sum()

data_to_use = initial_data.dropna()

data_to_use.shape

X = data_to_use.drop(axis=1, columns=['attack_cat'])
X = X.drop(axis=1, columns=['label'])
y1 = data_to_use['attack_cat'].values
y2 = data_to_use['label'].values

def data_ratio(y2):
    '''
    Calculate Y2's ratio
    '''
    unique, count = np.unique(y2, return_counts=True)
```

```
    ratio = round(count[0]/count[1], 1)
    return f'{ratio}:1 ({count[0]}/{count[1]})'

print('The class ratio for the original data:', data_ratio(y1))
plt.figure(figsize=(13,5))
sns.countplot(y1,label="Sum")
plt.show()
print('The class ratio for the original data:', data_ratio(y2))
sns.countplot(y2,label="Sum")
plt.show()

test_data = pd.read_csv(r'C:\Users\lenovo\Documents\IoT-Network-Intrusion-Detection-and-Classification-using-Explainable-XAI-Machine-Learning-main (1)\UNSW_NB15_testing-set.csv')
X_test = test_data.drop(axis=1, columns=['attack_cat'])
X_test = X_test.drop(axis=1, columns=['label'])

y1_test = test_data['attack_cat'].values
y2_test = test_data['label'].values

X_train = X
y1_train = y1
y2_train = y2

numerical_cols = X_train.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = X_train.select_dtypes(include=['object', 'bool']).columns

numerical_cols

t = [('ohe', OneHotEncoder(drop='first'), categorical_cols),
```

```
    ('scale', StandardScaler(), numerical_cols)]
col_trans = ColumnTransformer(transformers=t)
col_trans.fit(X_train)

X_train_transform = col_trans.transform(X_train)

X_test_transform = col_trans.transform(X_test)

X_train_transform.shape

X_test_transform.shape

pd.unique(y1)

pd.unique(y2)y1_train_transform = target_trans.transform(y1_train)
y1_test_transform = target_trans.transform(y1_test)

target_trans = LabelEncoder()
target_trans.fit(y1_train)

y1_train_transform = target_trans.transform(y1_train)
y1_test_transform = target_trans.transform(y1_test)

y1_train_transform

target_trans = LabelEncoder()
target_trans.fit(y2_train)
y2_train_transform = target_trans.transform(y2_train)
y2_test_transform = target_trans.transform(y2_test)
```

```
y2_train_transform

DTclf = tree.DecisionTreeClassifier()
DTclf.fit(X_train_transform, y2_train_transform)

feature_names = np.array(numerical_cols)
feature_names

importances = DTclf.feature_importances_
indices = np.argsort(importances)
features = np.array(numerical_cols)
plt.title('Feature Importances of Decision Tree Classifier')
count = 10 # top # importance
plt.barh(range(count), importances[indices][len(indices)-count:], color='g',
align='center')
plt.yticks(range(count), [features[i] for i in indices[len(indices)-count:]])
plt.xlabel('Relative Importance')
plt.show()

DTclf.feature_importances_

!pip install eli5

DTclf = DecisionTreeClassifier(criterion="entropy", max_depth=5)
DTclf = DTclf.fit(X_train_transform, y2_train_transform)
y_pred = DTclf.predict(X_test_transform)

from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y2_test_transform, y_pred))
report=metrics.classification_report(y2_test_transform,y_pred)
```

```
DTclf_name=['Decision Tree Classifer','RegLog']
print('Reporting for %s:'%DTclf_name)
print(report)

fig = plt.figure(figsize=(35, 30))
DTtree = tree.plot_tree(DTclf, feature_names = np.array(numerical_cols), class_names = ['Normal', 'Attack'], fontsize=14, proportion=True, filled=True, rounded=True)
for o in DTtree:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('red')
        arrow.set_linewidth(3)

fig.savefig('Decision Tree Classifier XAI Visualization Part 2.png')

import eli5
from eli5.sklearn import PermutationImportance
perm = PermutationImportance(DTclf, random_state=1).fit(X_test_transform, y2_test_transform)
eli5.show_weights(perm, feature_names = np.array(numerical_cols))

fig = plt.figure(figsize=(20, 16))
DTtree = tree.plot_tree(DTclf, feature_names = features, class_names = ['Normal', 'Attack'], fontsize=12, proportion=True, filled=True, rounded=True, max_depth=3)
for o in DTtree:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('red')
        arrow.set_linewidth(3)
```

```
fig.savefig('Decision Tree Classifier (Depth = 3 Nodes) Explainable AI Visualization.png')

fig = plt.figure(figsize=(20, 16))
DTtree = tree.plot_tree(DTclf, feature_names = features, class_names = ['Normal', 'Attack'], fontsize=12, proportion=True, filled=True, rounded=True, max_depth=5)
for o in DTtree:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('red')
        arrow.set_linewidth(3)
fig.savefig('Decision Tree Classifier (Depth = 5 Nodes) Explainable AI Visualization.png')

fig = plt.figure(figsize=(20, 16))
DTtree = tree.plot_tree(DTclf, feature_names = features, class_names = ['Normal', 'Attack'], fontsize=10, proportion=True, filled=True, rounded=True, max_depth=8)
for o in DTtree:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('red')
        arrow.set_linewidth(3)
fig.savefig('Decision Tree Classifier (Depth = 8 Nodes) Explainable AI Visualization.png')

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=123)
DTclf = tree.DecisionTreeClassifier()
scoring = ['accuracy', 'precision_micro', 'recall_micro', 'f1_micro', 'roc_auc_ovr']
cv_results = cross_validate(estimator=DTclf,
                            X=X_train_transform,
```

```
                    y=y2_train_transform,
                    scoring=scoring,
                    cv=cv,
                    return_train_score=False)

metrics.accuracy_score(y2_test_transform, gbtree.predict(X_test_transform))

predict_fn = lambda x: gbtree.predict_proba(x).astype(float)

features = np.array(numerical_cols)
explainer = lime.lime_tabular.LimeTabularExplainer(X_train_transform
,feature_names = features, class_names = ['Normal', 'Attack'], kernel_width=3)

np.random.seed(1)
i = 1653
exp = explainer.explain_instance(X_test_transform[i], predict_fn, num_features=5)
exp.show_in_notebook(show_all=False)

i = 10
exp = explainer.explain_instance(X_test_transform[i], predict_fn, num_features=5)
exp.show_in_notebook(show_all=False)

!pip install ryu

import pandas as pd

!pip install scikit-learn pandas numpy matplotlib pyshark

import pandas as pd
df = pd.read_csv(r'C:\Users\lenovo\Downloads\train\UNSW-NB15_1.csv')
```

```
from sklearn.model_selection import train_test_split

X = df.drop(columns=["label"])
y = df["label"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from sklearn.externals import joblib

class IDSController(app_manager.RyuApp):

    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(IDSController, self).__init__(*args, **kwargs)
        self.logger.info("IDS Controller initialized")
        # load the trained machine learning model
        self.clf = joblib.load("/path/to/trained/model.pkl")

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
```

```
        parser = datapath.ofproto_parser
        # add flow rules to divert traffic to the controller
        # the following code installs a flow rule to capture all traffic
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
ofproto.OFPCML_NO_BUFFER)]
        self.add_flow(datapath, 0, match, actions)

    @staticmethod
    def add_flow(datapath, priority, match, actions, buffer_id=None):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
actions)]
        if buffer_id:
            mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
priority=priority,
                                    match=match, instructions=inst)
        else:
            mod = parser.OFPFlowMod

from operator import attrgetter
from ryu.app import simple_switch_13
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub

class SimpleTrafficMonitor(simple_switch_13.SimpleSwitch13):
```

```
    def __init__(self, *args, **kwargs):
        super(SimpleTrafficMonitor, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.monitor_thread = hub.spawn(self._monitor)

    # Handling Asynchronous messages
    @set_ev_cls(ofp_event.EventOFPStateChange,[MAIN_DISPATCHER,
DEAD_DISPATCHER])
    def _state_change_handler(self, ev):
        datapath = ev.datapath
        tmFile = open("tmFile.txt","a+")
        if ev.state == MAIN_DISPATCHER:
            if datapath.id not in self.datapaths:
                self.logger.debug('register datapath: %016x', datapath.id)
                tmFile.write('register datapath: ' + str(datapath.id))
                self.datapaths[datapath.id] = datapath
        elif ev.state == DEAD_DISPATCHER:
            if datapath.id in self.datapaths:
                self.logger.debug('unregister datapath: %016x', datapath.id)
                tmFile.write('unregister datapath: '+ str(datapath.id))
                del self.datapaths[datapath.id]
                    tmFile.close()

    # Monitoring and collecting statistic information
    def _monitor(self):
        while True:
            for dp in self.datapaths.values():
                self._request_stats(dp)
            hub.sleep(10)
```

```
def _request_stats(self, datapath):
    self.logger.debug('send stats request: %016x', datapath.id)
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    req = parser.OFPFlowStatsRequest(datapath)
    datapath.send_msg(req)

    req = parser.OFPPortStatsRequest(datapath, 0, ofproto.OFPP_ANY)
    datapath.send_msg(req)

    req = parser.OFPGroupStatsRequest(datapath, 0, ofp.OFPG_ALL)
    datapath.send_msg(req)

    req = ofp_parser.OFPMeterStatsRequest(datapath, 0, ofp.OFPM_ALL)
    datapath.send_msg(req)


@set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
def _flow_stats_reply_handler(self, ev):
    '''body = ev.msg.body

    tmFile = open("tmFile.txt","a+")
    self.logger.info('datapath         in-port  eth-dst           out-port packets  bytes')
    self.logger.info('---------------- -------- ----------------- -------- -------- --------')
    tmFile.write('datapath         in-port  eth-dst           out-port packets  bytes')
    tmFile.write('---------------- -------- ----------------- -------- -------- --------')
    for stat in sorted([flow for flow in body if flow.priority == 1],
                       key=lambda flow: (flow.match['in_port'],
                                         flow.match['eth_dst'])):
```

```
    self.logger.info('%016x %8x %17s %8x %8d %8d',
                     ev.msg.datapath.id,
                     stat.match['in_port'], stat.match['eth_dst'],
                     stat.instructions[0].actions[0].port,
                     stat.packet_count, stat.byte_count)
    tmFile.write(
                     str(ev.msg.datapath.id)+" "+
                     str(stat.match['in_port'])+" "+ str(stat.match['eth_dst'])+ " "+
                     str(stat.instructions[0].actions[0].port)+ " "+
                     str(stat.packet_count)+ " "+ str(stat.byte_count))
tmFile.close()"'
tmFile = open("tmFile.txt","a+")
flows = []
for stat in ev.msg.body:
    flows.append('table_id=%s '
            'duration_sec=%d duration_nsec=%d '
            'priority=%d '
            'idle_timeout=%d hard_timeout=%d flags=0x%04x '
            'cookie=%d packet_count=%d byte_count=%d '
            'match=%s instructions=%s' %
            (stat.table_id,
             stat.duration_sec, stat.duration_nsec,
             stat.priority,
             stat.idle_timeout, stat.hard_timeout, stat.flags,
             stat.cookie, stat.packet_count, stat.byte_count,
             stat.match, stat.instructions))
self.logger.debug('FlowStats: %s', flows)
tmFile.write(str(flows))
tmFile.close()
```

```
    @set_ev_cls(ofp_event.EventOFPPortStatsReply, MAIN_DISPATCHER)
    def _port_stats_reply_handler(self, ev):
        body = ev.msg.body
        tmFile = open("tmFile.txt","a+")
        self.logger.info('datapath         port     rx-pkts  rx-bytes rx-error tx-pkts  tx-bytes tx-error')
        self.logger.info('---------------- -------- -------- -------- -------- -------- -------- --------')
        tmFile.write('datapath         port     rx-pkts  rx-bytes rx-error tx-pkts  tx-bytes tx-error')
        tmFile.write('---------------- -------- -------- -------- -------- -------- -------- --------')
        for stat in sorted(body, key=attrgetter('port_no')):
            self.logger.info('%016x %8x %8d %8d %8d %8d %8d %8d',
                             ev.msg.datapath.id, stat.port_no,
                             stat.rx_packets, stat.rx_bytes, stat.rx_errors,
                             stat.tx_packets, stat.tx_bytes, stat.tx_errors)
            tmFile.write(
                             str(ev.msg.datapath.id)+" "+ str(stat.port_no) + " "+
                             str(stat.rx_packets)+"          "+          str(stat.rx_bytes)+"          "+ str(stat.rx_errors)+ " "+
                             str(stat.tx_packets)+       "       "+       str(stat.tx_bytes)+"       "+ str(stat.tx_errors))
                tmFile.close()

    @set_ev_cls(ofp_event.EventOFPGroupStatsReply, MAIN_DISPATCHER)
    def group_stats_reply_handler(self, ev):
        tmFile = open("tmFile.txt","a+")
        groups = []
        for stat in ev.msg.body:
```

```
        groups.append('length=%d group_id=%d '
                      'ref_count=%d packet_count=%d byte_count=%d '
                      'duration_sec=%d duration_nsec=%d' %
                      (stat.length, stat.group_id,
                       stat.ref_count, stat.packet_count,
                       stat.byte_count, stat.duration_sec,
                       stat.duration_nsec))
    self.logger.debug('GroupStats: %s', groups)
    tmFile.write(str(groups))
    tmFile.close()

@set_ev_cls(ofp_event.EventOFPMeterStatsReply, MAIN_DISPATCHER)
def meter_stats_reply_handler(self, ev):
    tmFile = open("tmFile.txt","a+")
    meters = []
    for stat in ev.msg.body:
        meters.append('meter_id=0x%08x len=%d flow_count=%d '
                      'packet_in_count=%d byte_in_count=%d '
                      'duration_sec=%d duration_nsec=%d '
                      'band_stats=%s' %
                      (stat.meter_id, stat.len, stat.flow_count,
                       stat.packet_in_count, stat.byte_in_count,
                       stat.duration_sec, stat.duration_nsec,
                       stat.band_stats))
    self.logger.debug('MeterStats: %s', meters)
    tmFile.write(str(meters))
    tmFile.close()
```

9.2 RESULT

The system uses Supervised Machine Learning Algorithm to predict network attacks with higher accuracy and efficiency.

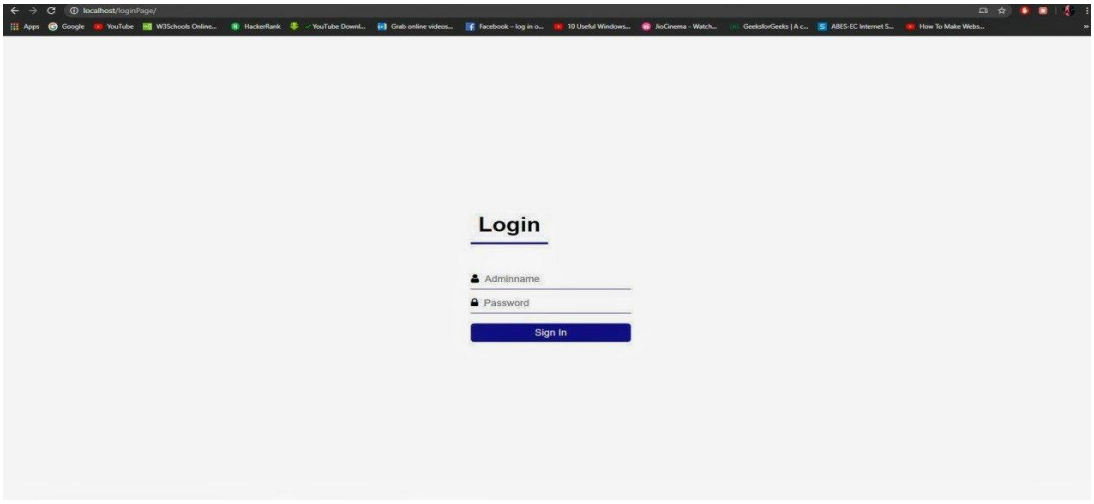


Fig 9.1 Login page

In this, the user needs to login with their registered mail id and password tender into the prediction page.

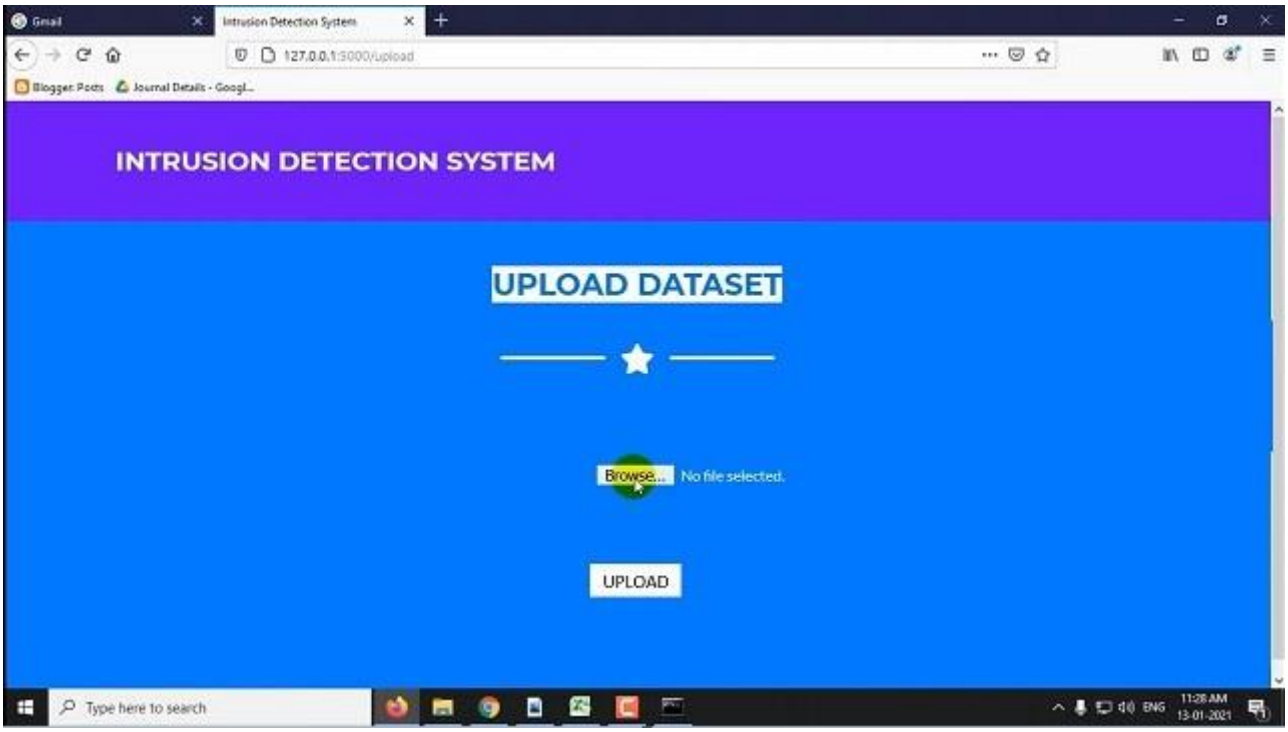
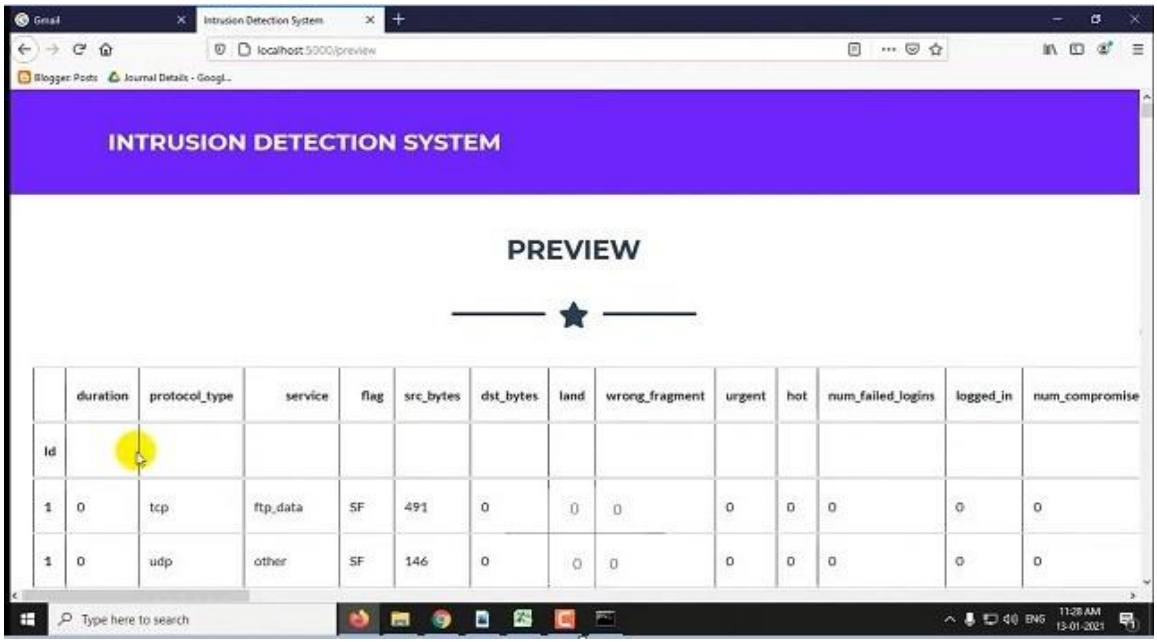


Fig 9.2. Uploading dataset



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromise
Id													
1	0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0	0
1	0	udp	other	SF	146	0	0	0	0	0	0	0	0

Fig.9.3 Dataset Preview

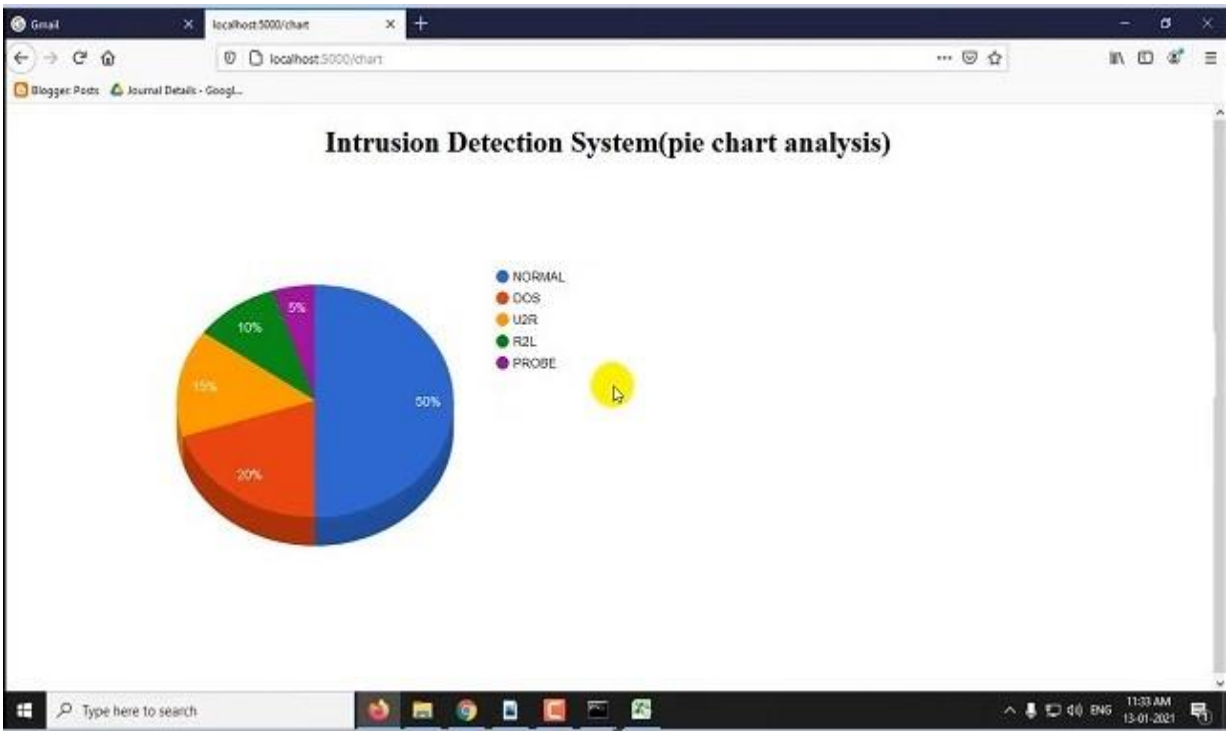


Fig.9.4 Final Output

ANALYSIS:

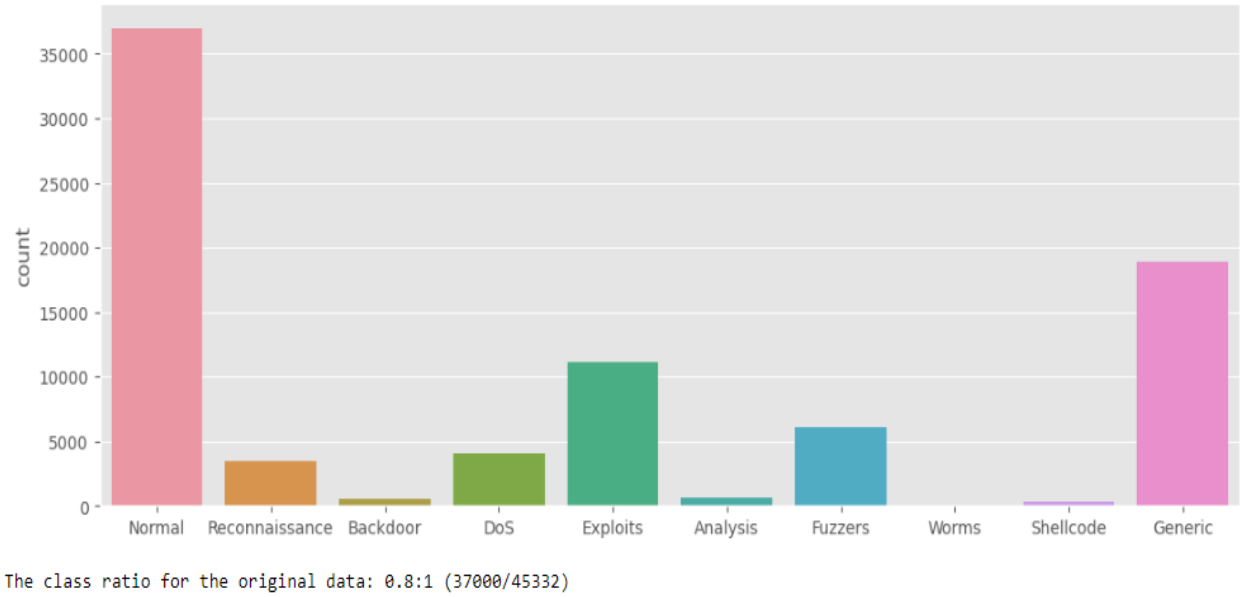


Fig 9.5 Analysis using MLP

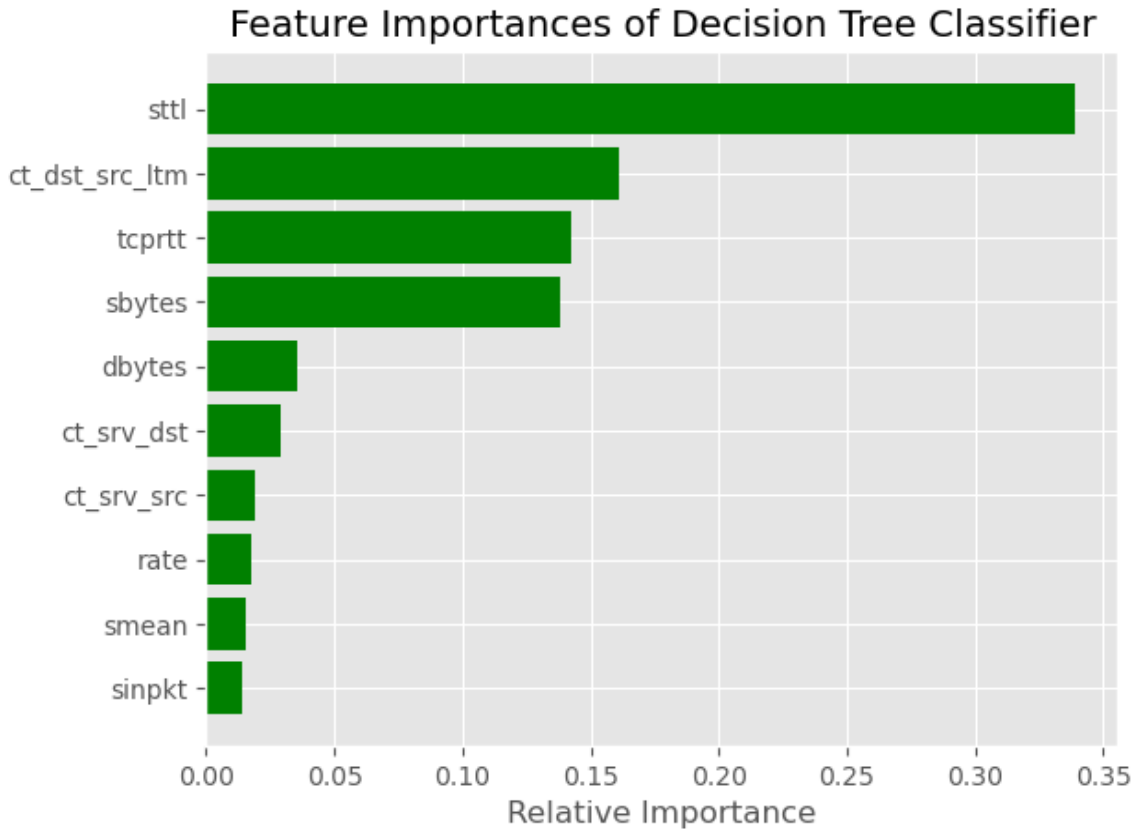


Fig 9.6 Features of Decision tree classifier

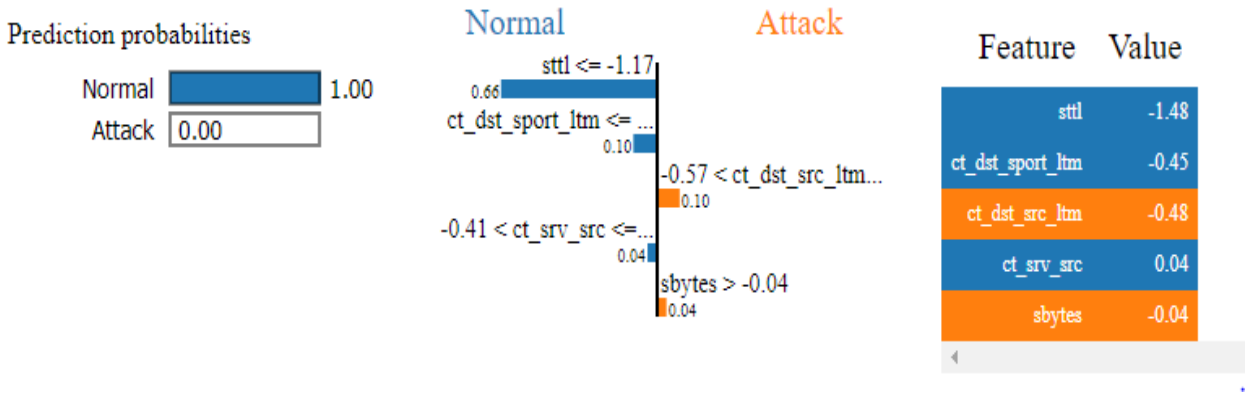


Fig 9.7 Analysis using XG Boost Case-1

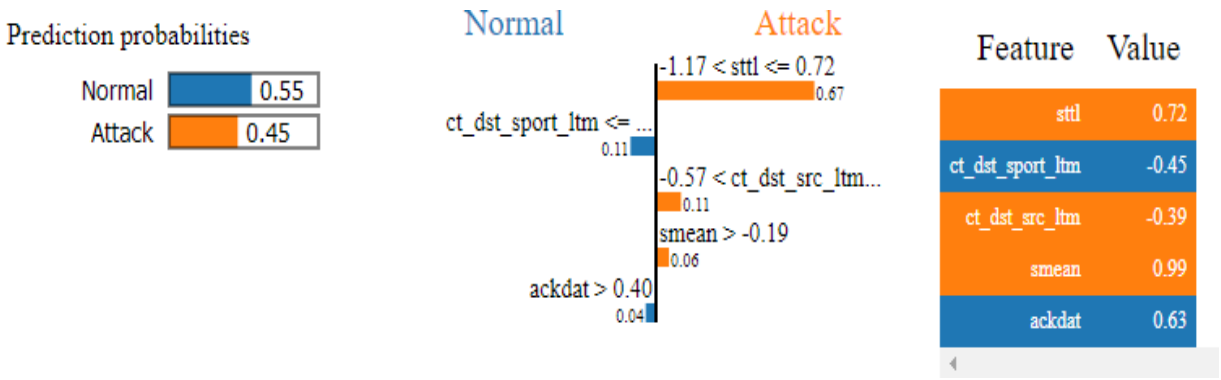


Fig 9.8 Analysis using XG Boost case-2

9.3 CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, the integration of Intrusion Detection Systems (IDS) with Software-Defined Networking (SDN) holds great promise for enhancing network security. The ability of SDN to centralize network control and provide programmability enables more efficient and effective IDS deployment, management, and response. By decoupling the control plane from the data plane, SDN allows for dynamic and flexible security policies, rapid threat detection, and mitigation.

The combination of IDS and SDN enables several key advantages. Firstly, SDN provides a global view of the network, allowing IDS to monitor and analyze traffic flows across multiple network segments and devices. This holistic perspective enhances the accuracy and effectiveness of intrusion detection. Secondly, SDN facilitates real-time communication between the IDS and network devices, enabling automated response mechanisms such as flow redirection, rate limiting, or isolation of compromised hosts. This reduces the time to detect and respond to security incidents, minimizing potential damage and limiting the spread of threats. Lastly, SDN's programmability allows for the dynamic adaptation of security policies based on changing network conditions and threat landscapes.

IDS algorithms need to evolve to effectively handle the dynamic and complex nature of SDN environments. Machine learning and artificial intelligence approaches can be leveraged to improve the accuracy of intrusion detection by analyzing large volumes of network data and identifying new and evolving threats.

CHAPTER 10

10. REFERENCES

[1] Mehdi S.A., Khalid J., Khayam S.A. International Workshop on Recent Advances in Intrusion Detection. Springer; Berlin/Heidelberg, Germany: 2011. Revisiting Traffic Anomaly Detection Using Software Defined Networking; pp. 161–180.

[2] Garcia-Teodoro P., Diaz-Verdejo J., Maciá-Fernández G., Vázquez E. Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges. Comput. Secur. 2009;28:18–28. doi: 10.1016/j.cose.2008.08.003.

[3] Ahmed M.E., Kim H., Park M. Mitigating DNS Query-Based DDoS Attacks with Machine Learning on Software-Defined Networking; Proceedings of the MILCOM 2017–2017 IEEE Military Communications Conference (MILCOM); Baltimore, MD, USA. 23–25 October 2017; pp. 11–16.

[4] Dawoud A., Shahristani S., Raun C. Deep Learning and Software-Defined Networks: Towards Secure IoT Architecture. Internet Things. 2018;3:82–89. doi: 10.1016/j.iot.2018.09.003.

[5] Herrera A., Camargo J.E. A Survey on Machine Learning Applications for Software Defined Network Security; Proceedings of the International Conference on Applied Cryptography and Network Security; Bogota, Colombia. 5–7 June 2019; Berlin/Heidelberg, Germany: Springer; 2019. pp. 70–93.

[6] Hu F., Hao Q., Bao K. A Survey on Software-Defined Network and Openflow: From Concept to Implementation. IEEE Commun. Surv. Tutor. 2014;16:2181–2206. doi: 10.1109/COMST.2014.2326417.

[7] Sultana N., Chilamkurti N., Peng W., Alhadad R. Survey on SDN Based Network Intrusion Detection System Using Machine Learning Approaches. Peer-Peer Netw. Appl. 2019;12:493–501. doi: 10.1007/s12083-017-0630-0.

[8] Hodo E., Bellekens X., Hamilton A., Tachtatzis C., Atkinson R. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey.
[9] Tiwari S., Pandita V., Sharma S., Dhande V., Bendale S. Survey on Sdn Based Network Intrusion Detection System Using Machine Learning Framework.
[10] Xie J., Richard Y.F., Huang T., Xie R., Liu J., Wang C., Liu Y. A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges.
[11] Chalapathy R., Chawla S. Deep Learning for Anomaly Detection: A Survey. ArXiv.
[12] Aldweesh A., Derhab A., Emam A.Z. Deep Learning Approaches for Anomaly-Based Intrusion Detection Systems: A Survey, Taxonomy, and Open Issues. Knowl.-Based Syst.
[13] Ahmad Z., Khan S., Shiang W., Abdullah J., Ahmad F. Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches.
[14] Injadat M., Moubayed A., Nassif A.B., Shami A. Systematic ensemble model selection approach for educational data mining. Knowl. -Based Syst.
[15] Singh D., Ng B., Lai Y.-C., Lin Y.-D., Seah W.K. Modelling Software-Defined Networking: Software and Hardware Switches. J. Netw.
[16] Krongbaramee P., Somchit Y. Implementation of SDN Stateful Firewall on Data Plane Using Open VSwitch; Proceedings of the 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE); Nakhonpathom, Thailand.
[17] Lockwood J.W., McKeown N., Watson G., Gibb G., Hartke P., Naous J., Raghuraman R., Luo J. NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing; Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education (MSE'07); San Diego, CA, USA.

[18] Lu G., Guo C., Li Y., Zhou Z., Yuan T., Wu H., Xiong Y., Gao R., Zhang Y. ServerSwitch: A Programmable and High Performance Platform for Data Center Networks; Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11);
[19] Anwer M.B., Motiwala M., Tariq M.B., Feamster N. Switchblade: A Platform for Rapid Deployment of Network Protocols on Programmable Hardware; Proceedings of the ACM SIGCOMM 2010 Conference; New Delhi, India
[20] Medved J., Varga R., Tkacik A., Gray K. Opendaylight: Towards a Model-Driven Sdn Controller Architecture; Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks; Sydney, NSW, Australia.
[21] Itoh T., Sakai M., Okada M. Floodlight. Application CA 139471.
[22] Gude N., Koponen T., Pettit J., Pfaff B., Casado M., McKeown N., Shenker S. NOX: Towards an Operating System for Networks.
[23] Tootoonchian A., Ganjali Y. Hyperflow: A Distributed Control Plane for Openflow; Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking; San Jose, CA, USA.
[24] Prakash A., Priyadarshini R. An Intelligent Software Defined Network Controller for Preventing Distributed Denial of Service Attack; Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT); Coimbatore, India.