**NETWORK THREAT DETECTION AND MITIGATION**

**INT300 – INTERNSHIP PROJECT**

**PROJECT REPORT**

***Submitted by***

**PAVITHRAA D – E0222041**

***In partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**(Cyber Security & Internet of Things)**

**Sri Ramachandra Faculty of Engineering and Technology**

**Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116**

**APRIL, 2024**

**NETWORK THREAT DETECTION AND MITIGATION**

**INT300 – INTERNSHIP PROJECT**

**PROJECT REPORT**

***Submitted by***

**PAVITHRAA D – E0222041**

***In partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**(Cyber Security & Internet of Things)**

**Sri Ramachandra Faculty of Engineering and Technology**

**Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116**

**APRIL, 2024**

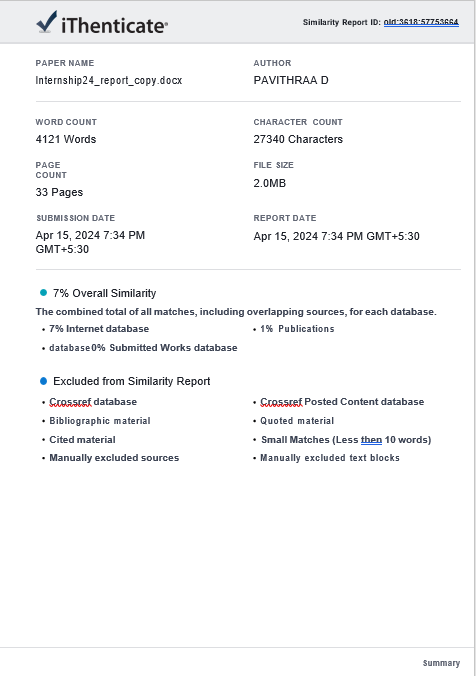
**BONAFIDE CERTIFICATE**

Certified that this project report **“NETWORK THREAT DETECTION AND MITIGATION ”** is the bonafide record of work done by **“ PAVITHRAA D – E0222041 ”** who carried out the internship work under my supervision.

**Signature of the Supervisor Signature of Programme Coordinator**

|  |  |
| --- | --- |
| **Dr. R Somasundaram**  **Assistant Professor,**  Department of Cybersecurity and IoT  Sri Ramachandra Faculty of Engineering and Technology,  SRIHER, Porur, Chennai-600 116. | **Dr. Jayanthi G**  **Associate Professor,**  Department of Cybersecurity and IoT  Sri Ramachandra Faculty of Engineering and Technology,  SRIHER, Porur, Chennai-600 116. |

**Evaluation Date:**



**Signature of the Supervisor**

**ACKNOWLEDGEMENT**

I express my sincere gratitude to our Programme Coordinator **Dr. Jayanthi G** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor(s), **Dr. R Somasundaram ,** Department of Cybersecurity and IoT, Sri Ramachandra faculty of Engineering and Technology for extending help and encouragement throughout the project. Without his continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of Sri Ramachandra Faculty of Engineering and Technology, my beloved parents and friends for extending the support, who helped us to overcome obstacles in the study.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **7** |
|  | **LIST OF FIGURES** | **8** |
|  | **LIST OF ABBREVATION** | **9** |
|  | **LIST OF TABLES** | **9** |
| **1** | **INTRODUCTION** | **10** |
|  | * 1. Introduction | **10** |
|  | 1.2 Problem Statement | **10** |
|  | 1.3 Objective | **10** |
| **2** | **LITERATURE REVIEW** | **11** |
| **3** | **METHODOLOGY** | **14** |
| **4** | **TOOLS AND TECHNOLOGY** | **16** |
| **5** | **IMPLEMENTATION** | **17** |
|  | 5.1 Detection using ML | **17** |
|  | 5.2 Mitigation | **18** |
|  | 5.3 IDS and IPS using Snort | **21** |
|  | 5.4 Resource Demand | **27** |
| **6** | **RESULTS** | **29** |
| **7** | **APPENDICIES** | **30** |
|  | Appendix-1: Code | **30** |
|  | Appendix-2: Screenshots | **34** |
|  | **REFERENCES** | **36** |
|  | **WORKLOG** | **37** |

**ABSTRACT**

This project aims to improve network security using advanced ML techniques. The focus is on developing specific algorithms and optimizations for cybersecurity. The motive is to increase the effectiveness of ML models in detection and response to threats while reducing computational resources. The project integrates advanced anomaly detection, and behavioural analysis to improve network monitoring. The objective of the project is to establish a more robust defence against cyber threats, and to create a safer digital world for people and organizations. In our project, we are deploying Snort tool on our network for both Detection (IDS) and Prevention (IPS). Snort adds an additional layer of security by proactively monitoring for and blocking threats. We are also optimizing resource usage to make sure these security measures are running smoothly without putting too much pressure on our system.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURES** | **PAGE NO** |
| 5.3.1.1 | Setting Up Snort | 16 |
| 5.3.2.1 | Snort Configuration | 17 |
| 5.3.2.3 | Snort Configuration  Completed | 18 |
| 5.3.3.1 | Threat Detection | 19 |
| 5.3.4.1 | Kali-Linux-Threat | 19 |
| 5.3.4.2 | Windows-Threat | 20 |
| 5.3.5.1 | Detecting the Threat | 20 |
| 5.3.6.1 | Windows Recurring  Threat | 21 |
| 5.3.6.2 | Kali Linux Recurring Threat | 21 |
| 5.3.7.1 | Threat-Prevention in  Ubuntu | 22 |
| 7.1.1 | Column-Dataset | 29 |
| 7.1.2 | Confusion-Matrix | 29 |
| 7.1.3 | Accuracy of Attack  Using ML | 30 |
| 7.1.4 | Mitigation using ML | 30 |
| 7.1.5 | Classifier Performance  Matrics | 30 |

**LIST OF ABBREVATION**

1. ML - Machine Learning

2. DT - Decision Tree

3. GBM - Gradient Boosting

4. kNN - K-Nearest Neighbors

5. DOS - Denial of Service

6. IDE - Integrated Development Environment

7. VM - Virtual Machine

8. IDS - Intrusion Detection System

9. IPS - Intrusion Prevention System

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TABLES** | **PAGE NO** |
| 2.1 | Literature Review | 11 |
| 4.1 | Tools and Technology | 16 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **INTRODUCTION**

Cybersecurity is a critical need in today’s digital world, where the threat of online breaches is on the rise. The main purpose of cybersecurity is to identify and stop network intrusions, particularly those that involve malicious activities. Machine learning plays an important role in detecting and stopping such intrusions, but its widespread implementation is a challenge due to its high demand for resources. Our research aims to improve machine learning models’ performance by optimizing resource utilization. We plan to do this by developing powerful algorithms and optimization techniques that are specifically designed to detect and stop intrusions. By doing so, we will be able to increase the efficiency of intrusion detection in the cybersecurity industry, fulfilling a critical need. Our contributions will help strengthen digital defenses against ever-changing threats.

* 1. **PROBLEM STATEMENT**

Enhancing the effectiveness of resource-demanding machine learning models for detecting intrusion is a critical research problem in the field of cybersecurity.

* 1. **OBJECTIVE**

Develop streamlined algorithms and optimization techniques to reduce the resource demands of machine learning models, enhancing their efficiency for robust malware detection in cybersecurity.

**CHAPTER 2**

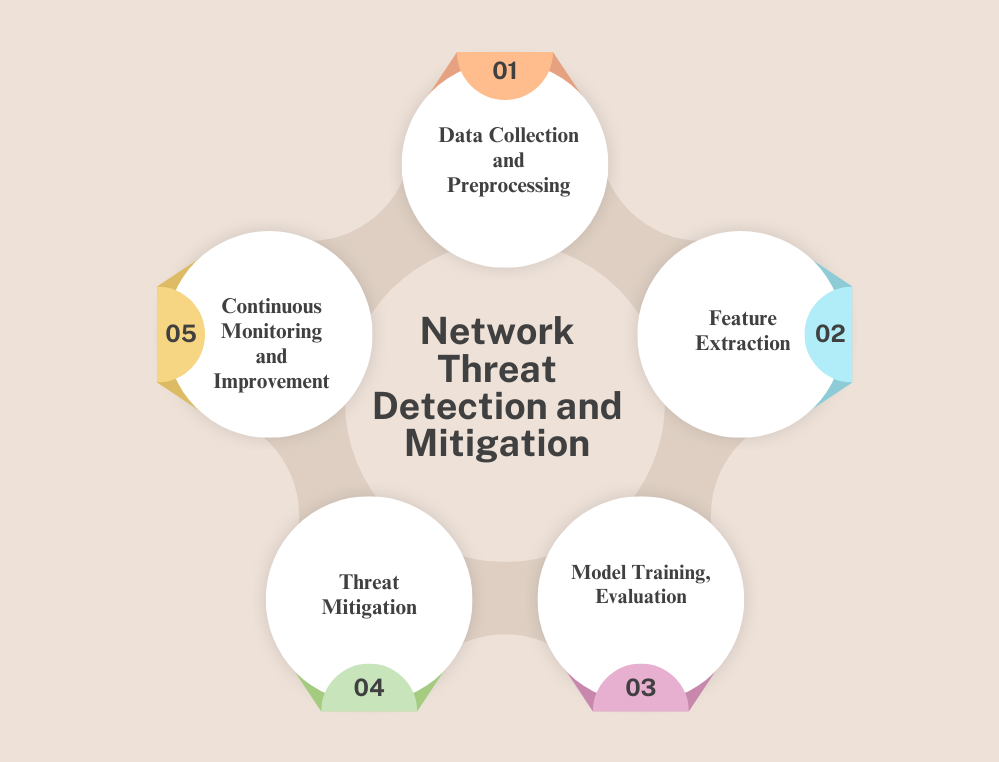
**LITERATURE REVIEW**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Journal Name | Authors | Methodology | Advantage | Disadvantage |
| 1 | Knowledge Graph Based Large Scale Network Security Threat Detection Techniques  (Janunary 31, 2024) | Zhifeng Hu | Network threat Detection using Knowledge graph,  Feature Template extraction and  FT-CNN-BILSTM-CRF | High accuracy and F1-Score in detecting network threats, superior multi-step attack detection compared to other methods. | High computational resource demand limits scalability in large networks. |
| 2 | An Artificial Neural Network Autoencoder for Insider Cyber Security Threat Detection  (23 November, 2023) | Karthikeyan Saminathan  Sai Tharun Reddy Mulk  Sangeetha Damodharan  Rajagopal Maheswar  Josip Lorincz | Network threat Detection using an unsupervised deep learning approach with an artificial neural network (ANN)-based autoencoder | This model is really good at spotting insider cyber threats because it looks closely at how users and systems behave. It's very reliable and can catch these threats with just a few abnormal signs. | Needs a lot of computing power needs improvements to be faster. Also, could be better by analyzing different kinds of assets and using more advanced algorithms. |
| 3 | Malware Analysis and Detection Using Machine Learning Algorithms  (3 November 2022) | Muhammad Shoaib Akhtar  Tao Feng | Involves using machine learning techniques, including Naive Bayes, SVM, J48, RF, and a proposed approach, to analyze and detect polymorphic malware, with a focus on measuring detection accuracy and comparing classifier performances. | Machine learning algorithms like DT, CNN, and SVM show high accuracy in detecting malware, making them effective for cybersecurity. | Machine learning models may require large amounts of labeled data and computational resources, making them resource-intensive to train and deploy. |
| 4 | Malware Detection and Prevention using Artificial  Intelligence Techniques  (13 January 2022) | Md Jobair Hossain Faruk ,  Hossain Shahriar , Maria Valero ,  Farhat Lamia Barsha , Shahriar Sobhan Md Abdullah Khan , Michael Whitman , Alfredo Cuzzocreak , Dan Lo ,  Akond Rahman and Fan Wu | reviewing existing malware detection technologies, identifying weaknesses, and suggesting AI-based futuristic solutions for enhanced detection and prevention efficiency. | Utilizing Artificial Intelligence (AI) for developing Anti-Malware Systems shows promise in effectively detecting and preventing malware attacks | Implementing AI-based solutions for malware detection may require significant computational resources and expertise, potentially increasing complexity and costs. |
| 5 | Malware Detection Using Deep Learning  (December 29, 2023) | Achi Harrisson Thiziers,  Koné Tiémoman, N’guessan Behou Gérard,  Traoré Tiémoko Qouddouss Kabir | Proposed a Deep Learning-based model for malware detection using artificial neural networks trained on machine characteristics data, achieving an 83% accuracy rate. | Developed a highly efficient malware detection model using deep learning and machine characteristics data. | Limited by the reliance on machine characteristics data, potentially missing newer malware variants with different patterns. |
| 6 | Malware Detection using Machine  Learning  (April 2023) | Navya VK,  Jai Krishna Pandey,  Priyanshu Anjney,  Puttam Venkata Sesha Reddy,  Ompuri O | Developed a web-based framework using feature selection and multiple machine learning algorithms to detect Android malware with an 86% detection rate. | Provides a free and accessible solution for detecting malware, potentially reducing the risk of ransomware attacks and aiding users who cannot afford commercial software. | Limited advancement compared to deep learning or natural language processing techniques may restrict accuracy. |
| 7 | Cyber Threat Detection based on Artificial Neural Networks using Event Profiles  (November 2019) | Jonghoon Lee, Jonghyun Kim,  Ikkyun Kim,  Kijun Han | Developed an event profiling method involving data aggregation, decomposition, TF-IDF normalization, and vectorization for input into deep learning models. | Proposed an AI-SIEM system using event profiles and artificial neural networks, improving cyber-threat detection and enabling efficient response to security alerts. | Requires significant effort to manually label raw security events for supervised learning, potentially slowing down the process of dataset construction. |
| 8 | Intrusion Detection System  (April 2017) | Mohit Tiwari,  Raj Kumar,  Akash Bharti,  Jai Kishan | Developed an IDS tool capable of monitoring network or system activities to detect and prevent malicious activity using various techniques, methods, and algorithms. | Provides additional protection beyond firewall technology by monitoring and detecting intrusions, both externally and internally. | Requires strong authentication, policy adherence, and human intervention, potentially increasing complexity and resource needs. |
| 9 | Network Intrusion Detection System using Deep Learning  (June 2021) | Lirim Ashiku,  Cihan Dagli | Proposed the use of deep learning architectures to develop an adaptive network intrusion detection system capable of detecting and classifying network attacks, demonstrated using the UNSW-NB15 dataset. | Proposed deep learning-based network intrusion detection system achieved high accuracy in multiclass classification | Acknowledges the need for improvements, including feature reduction methods and handling zero-day attacks, which may require additional research and development. |
| 10 | Android Malware Detection System using Machine Learning  (April 2023) | Nuren Natasha Maulat Nasri,  Mohd Faizal Ab Razak,  RD Rohmat Saedudin,  Salwana Mohamad,  Ahmad Firdaus | Proposed a machine learning-based Android malware detection system trained with five classifiers and evaluated using WEKA, achieving high accuracy with Random Forest classifiers. | Achieved high detection rates using machine learning classifiers for Android malware detection. | Faces challenges with false alarms and selecting relevant features for improved detection performance. |

Table 2.1

**CHAPTER 3**

**METHODOLOGY**

****

**Data Collection and Preprocessing:**

Collecting data on network traffic from various sources guarantees a representative sample of both legitimate and malicious activity. To get the dataset ready for model training, preprocessing includes activities like data cleaning, managing categorical variables, and handling missing values. After pertinent data is gathered, preprocessing procedures are used to improve the relevance and quality of the data. Techniques like packet capture and log analysis are used in this process.

**Feature Extraction:**

Feature extraction is the process of finding and removing pertinent features from the preprocessed data in order to accurately depict network traffic. In order to provide input variables for the machine learning models, this entails extracting attributes like protocol type, service, and packet size. Informative features that capture the fundamental properties of network traffic are extracted using feature extraction techniques such as statistical analysis, dimensionality reduction, and pattern recognition.

**Model Training, Evaluation:**

Training a model involves choosing and utilizing labeled data and extracted features to train machine learning algorithms. To identify patterns suggestive of malevolent activity, a variety of algorithms, including Random Forests and Decision Trees, are trained on the dataset. The effectiveness of the model is evaluated using metrics like accuracy, precision, recall, and F1-score. After being trained and assessed, the models are implemented in the real-world setting to detect and address threats instantly.

**Threat Mitigation:**

Strategies for mitigating threats entail making recommendations and putting plans into action to address identified risks and lessen their influence on the network. In order to identify and stop malicious activity in real-time, this involves implementing intrusion detection and prevention systems (IDS/IPS) like Snort. Proactive steps like patch management, network segmentation, and access controls are also put in place to lessen the attack surface and mitigate potential vulnerabilities.

**Continuous Monitoring and Improvement:**

Continuous monitoring entails keeping an eye on security alerts and network traffic in real-time in order to quickly identify and address new threats. This entails doing routine security assessments to find potential vulnerabilities as well as examining alerts produced by IDS/IPS systems and analyzing security logs. To improve overall security posture, continuous improvement entails updating threat signatures, improving machine learning models, and modifying security policies in response to changing organizational needs and evolving threats.

**CHAPTER 4**

**TOOLS AND TECHNOLOGY**

|  |  |  |
| --- | --- | --- |
| **HARDWARE/SOFTWARE** | **DEVELOPED BY** | **USE IN THIS PROJECT** |
| R Programming Language | R Development Core Team | data analysis, machine learning, and statistical computing tasks |
| RStudio integrated development environment (IDE) for R | RStudio, PBC | Used as a development environment for writing, executing, and debugging R code |
| R packages/libraries (e.g., rpart, randomForest, e1071, class, caret, keras, naiveBayes) | Various contributors within the R community | Utilized for implementing machine learning algorithms, performing data preprocessing, model evaluation, and other analytical tasks in R programming language |
| Virtual Machine Environment (VM) | Morton L. Heilig | Utilized as a platform to host and run multiple operating systems and tools for network security analysis. |
| Kali Linux operating system | Offensive Security | Pre-installed tools for network reconnaissance, vulnerability assessment, and exploitation. |
| Ubuntu operating system | Canonical Ltd | Employed as an alternative operating system environment for certain tasks within the project |
| Nmap (Network Mapper) tool | Gordon Lyon (Fyodor) | Utilized for network exploration and auditing purposes, including port scanning, service enumeration, and information about network hosts and services. |
| Snort intrusion detection and prevention system (IDS/IPS) | Martin Roesch | Deployed as an intrusion detection and prevention system to monitor network traffic in real-time, detect suspicious activities. |

Table 4.1

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 DETECTION USING ML:**

**Dataset Loading and Data Preparation:**

* The process begins with loading a dataset from a CSV file located at a specified path. The dataset is then examined using functions such as head, print, and summary to gain an understanding of its structure and contents.
* The relevant columns for analysis are chosen using the selected\_columns variable, which consists of features related to network traffic and the target variable, 'attack\_cat', representing various attack categories.
* To ensure data integrity, missing values are identified using the colSums function. The 'attack\_cat' variable is then converted to a factor type using as.factor for classification purposes.

**Model Development and Assessment:**

* Various machine learning models are trained and assessed to predict the attack categories based on the selected features.
* The randomForest, ranger, and naiveBayes libraries are employed to implement Random Forest, Decision Tree, Naive Bayes, Gradient Boosting, and k-Nearest Neighbors classifiers, respectively.
* Each classifier is trained on the training data, and upon evaluation with the test data, a confusion matrix is generated to assess its performance.

**Model Evaluation Metrics:**

* A custom function compute\_metrics is created to compute evaluation metrics based on the confusion matrix produced by the classifiers.
* Metrics such as accuracy, precision, recall, and F1-score are calculated to evaluate the performance of each classifier.
* These metrics offer insights into the model's capability to accurately classify instances across different attack categories.

**Data Visualization with ggplot:**

* The outcomes of the model evaluation are visualized using bar plots generated with the ggplot2 library.
* The performance metrics (Accuracy, Precision, Recall, and F1-Score) of each classifier are displayed side by side in the bar plots for easy comparison.
* These visual representations provide a clear and intuitive way to assess the performance of the classification models, facilitating model selection and decision-making.

**5.2** **MITIGATION :**

**1. Reconnaissance Attacks:**

Techniques used by attackers to gather information about the target network or system, such as port scanning, vulnerability scanning, or network mapping.

Mitigation:

* Implement strict firewall rules to restrict unauthorized access and port scanning.
* Use intrusion detection/prevention systems (IDS/IPS) to monitor and block reconnaissance activities.
* Regularly update and patch software to eliminate known vulnerabilities.
* Employ techniques like IP address obfuscation, honeypots, and deception technologies to confuse attackers.

**2. Backdoor Attacks:**

Malicious code or techniques that allow unauthorized remote access and control over a system or application.

Mitigation:

* Keep software up-to-date and apply security patches promptly.
* Use application whitelisting to prevent unauthorized software from running.
* Implement strict access controls and regularly review user privileges.
* Deploy host-based intrusion detection systems (HIDS) to monitor for suspicious activities.

**3. Denial of Service (DoS) Attacks:**

Attempts to make a system, network, or service unavailable to legitimate users by overwhelming it with traffic or requests.

Mitigation:

* Implement rate-limiting and traffic filtering mechanisms at the network level.
* Configure load balancers and web application firewalls (WAFs) to mitigate DoS attacks.
* Ensure redundancy in network infrastructure and server capacity.
* Have a documented and tested incident response plan for DoS attacks.

**4. Exploit Attacks:**

Exploits leverage vulnerabilities in software or systems to gain unauthorized access or execute malicious code.

Mitigation:

* Regularly update and patch software, operating systems, and applications.
* Deploy web application firewalls (WAFs) to detect and block exploit attempts.
* Implement secure coding practices and conduct regular code reviews.
* Use address space layout randomization (ASLR) and data execution prevention (DEP) to mitigate memory-based exploits.

**5. Fuzzing Attacks:**

Techniques that involve sending malformed or unexpected input to an application or system to identify and exploit vulnerabilities.

Mitigation:

* Implement input validation and sanitization mechanisms.
* Use secure coding practices and perform code reviews to identify and fix vulnerabilities.
* Deploy web application firewalls (WAFs) to detect and block fuzzing attempts.
* Regularly update and patch software and applications.

**6. Worm Attacks:**

Self-replicating malware that spreads across networks and systems, often exploiting vulnerabilities or leveraging social engineering tactics.

Mitigation:

* Deploy antivirus and antimalware solutions with up-to-date signatures.
* Implement network segmentation and access controls to limit worm propagation.
* Regularly update and patch software and operating systems.
* Use intrusion detection/prevention systems (IDS/IPS) to monitor for worm activities.

**7. Shellcode Attacks:**

Malicious code injected into a system or application to execute arbitrary commands or gain unauthorized access.

Mitigation:

* Deploy data execution prevention (DEP) and address space layout randomization (ASLR) mechanisms.
* Implement strict input validation and sanitization measures.
* Use secure coding practices and conduct regular code reviews.
* Deploy web application firewalls (WAFs) to detect and block shellcode attempts.

**8. Generic Attacks:**

A broad category encompassing various types of attacks not specifically classified.

Mitigation:

* Implement defense-in-depth security measures, including firewalls, IDS/IPS, access controls, and secure configuration practices.
* Regularly update and patch software, operating systems, and applications.
* Conduct security awareness training for employees to recognize and report potential threats.
* Develop and test an incident response plan to effectively respond to and mitigate attacks.

**5.3. Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) using Snort**

**5.3.1. INSTALLATION**

Install Snort for network intrusion detection and prevention on Ubuntu

( Burp Suite).

**sudo apt-get install snort -y**

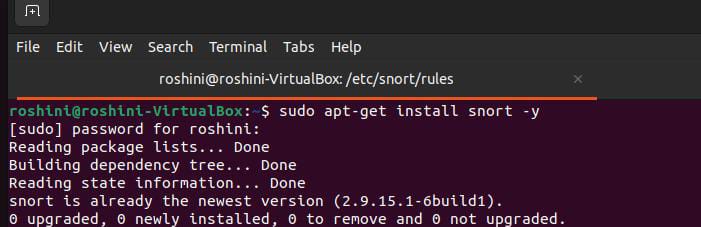


Fig 5.3.1.1

**5.3.2. CONFIGURATION**

Configuring Snort on Ubuntu for Effective Intrusion Detection Using Nano Text Editor.

.

**cd /etc/snort/rules**

**nano local.rules**

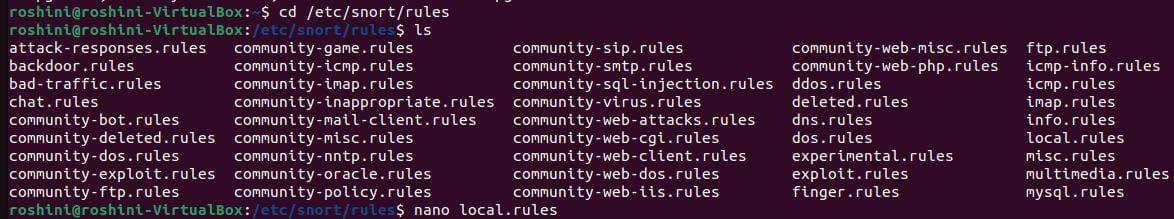


Fig 5.3.2.1

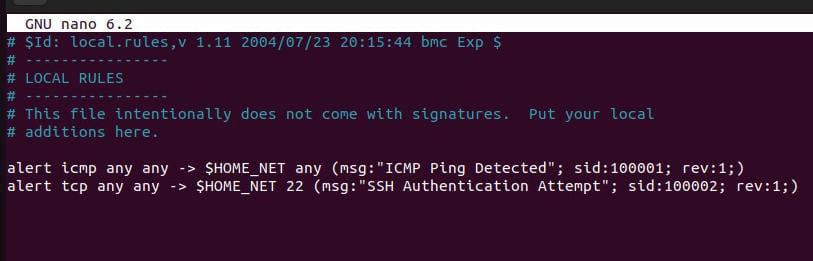
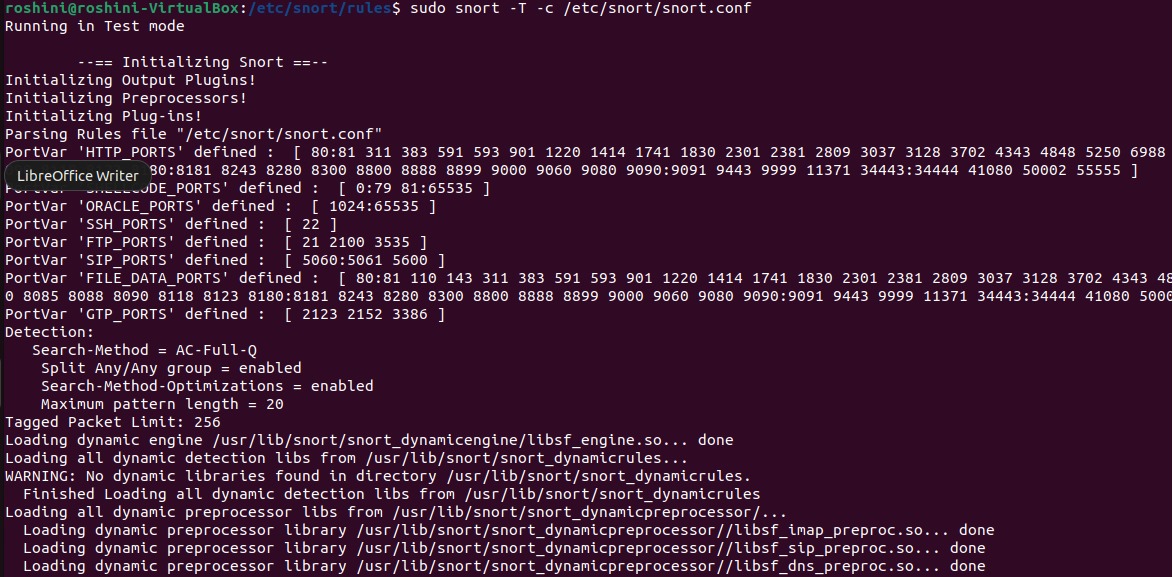


Fig 5.3.2.2

**sudo snort -T -c /etc/snort/snort.conf**



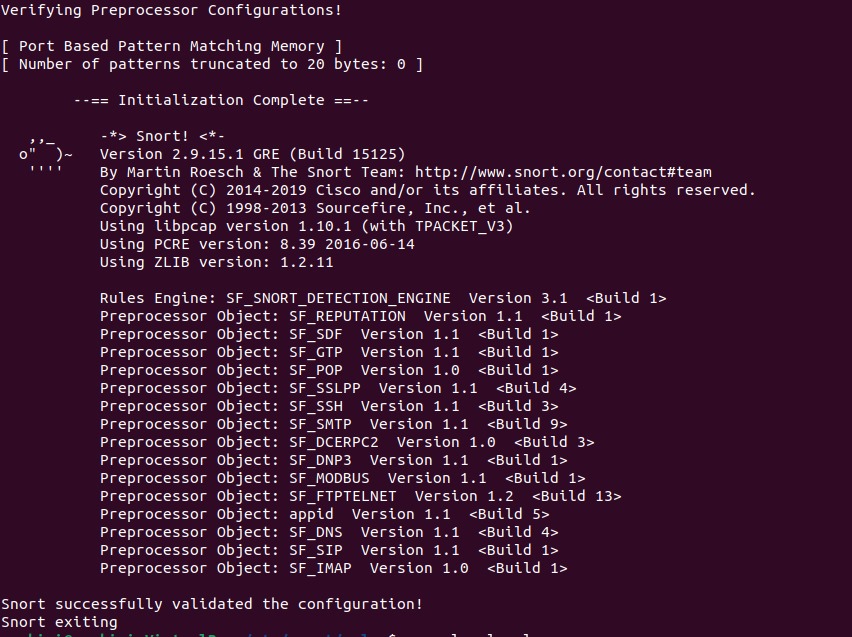
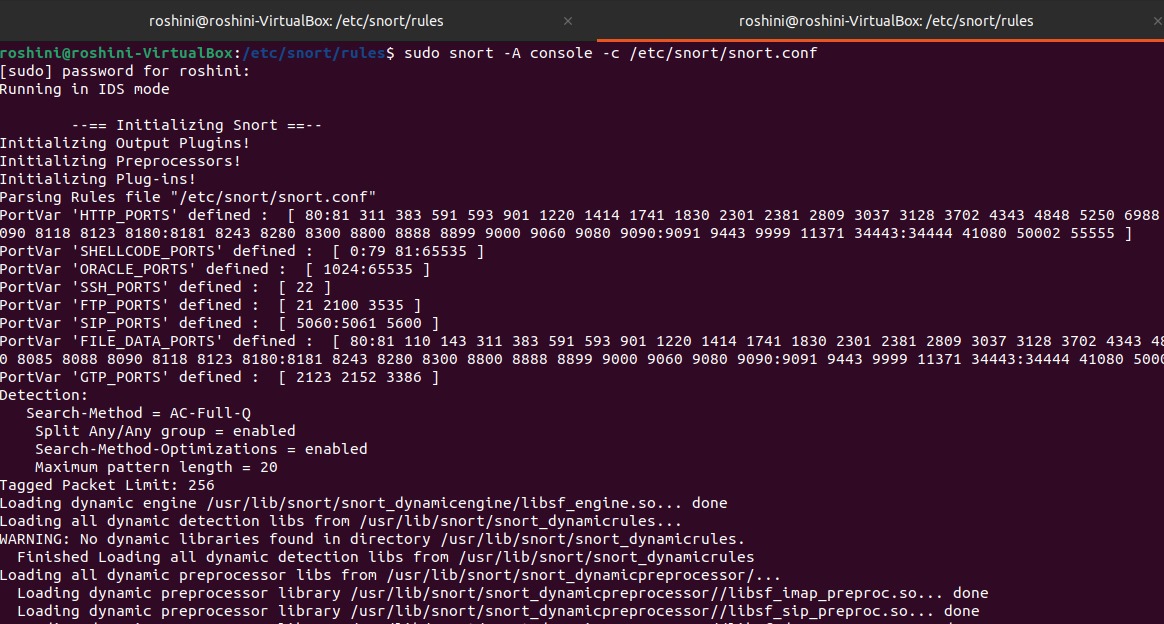


Fig 5.3.2.3

**5.3.3.**  **DETECTING THREATS** **USING SNORT**

Beginning Threat Detection with Snort on Ubuntu to safeguard against potential security breaches.

**sudo snort -A console -c /etc/snort/snort.conf**



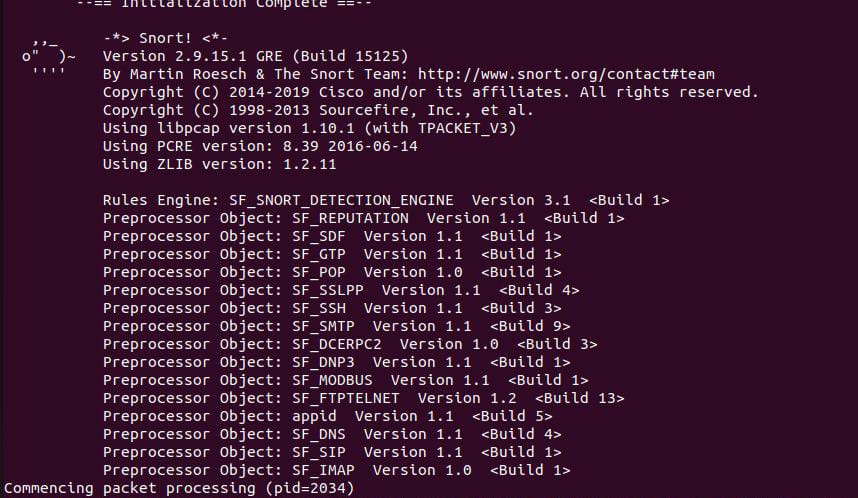


Fig 5.3.3.1

**5.3.4.** **THREAT SIMULATION**

* Sending Attacks from Kali Linux to Ubuntu

**nmap -Pn -p 22 10.0.2.15**

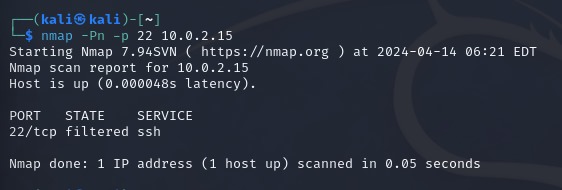


Fig 5.3.4.1

* Sending Attacks from Windows to Ubuntu

**Ping 10.0.2.15**

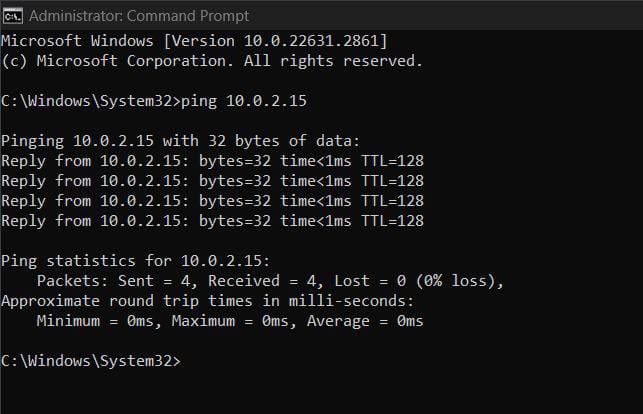


Fig 5.3.4.2

**5.3.5. THREAT DETECTED**

Identification of Attacks from Kali Linux and Windows on Ubuntu

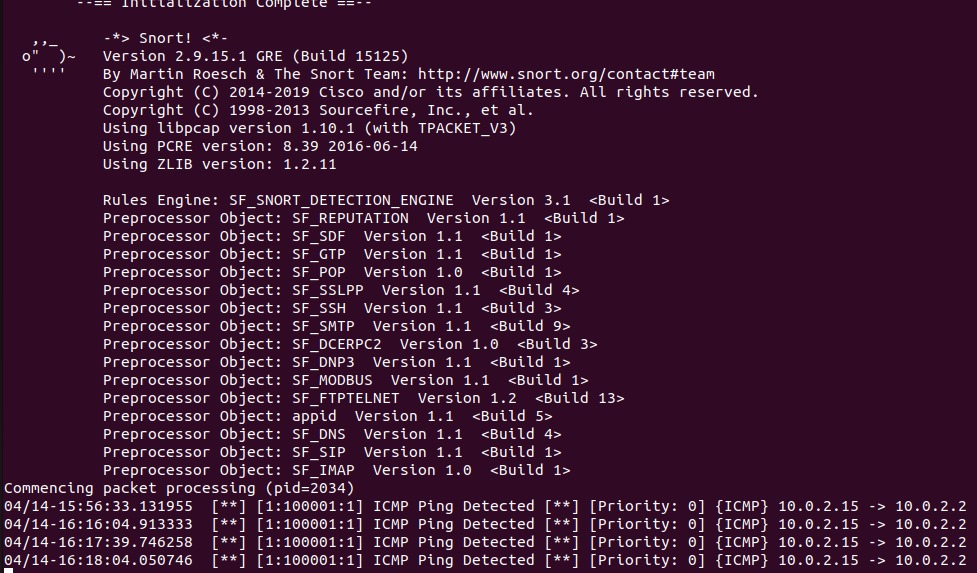


Fig 5.3.5.1

**5.3.6. REPEAT THREAT DETECTION**

* Uncovering Attacks from Windows on Ubuntu

**ftp 10.0.2.15**

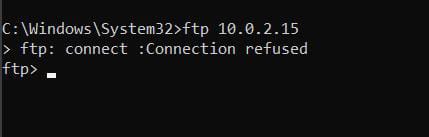


Fig 5.3.6.1

* Uncovering Attacks from Kali Linux on Ubuntu

**ftp 10.0.2.15**

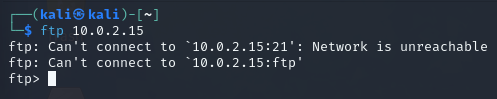
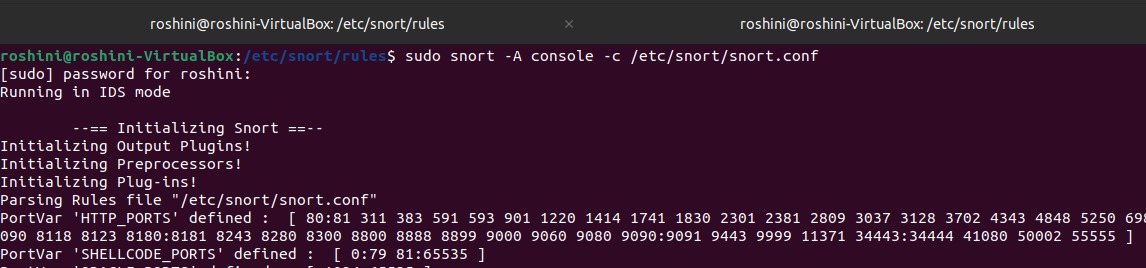
****

Fig 5.3.6.2

**5.3.7. THREAT PREVENTION**

Ubuntu Successfully Defends Against Attacks from Kali Linux and Windows

**sudo snort -A console -c /etc/snort/snort.conf**



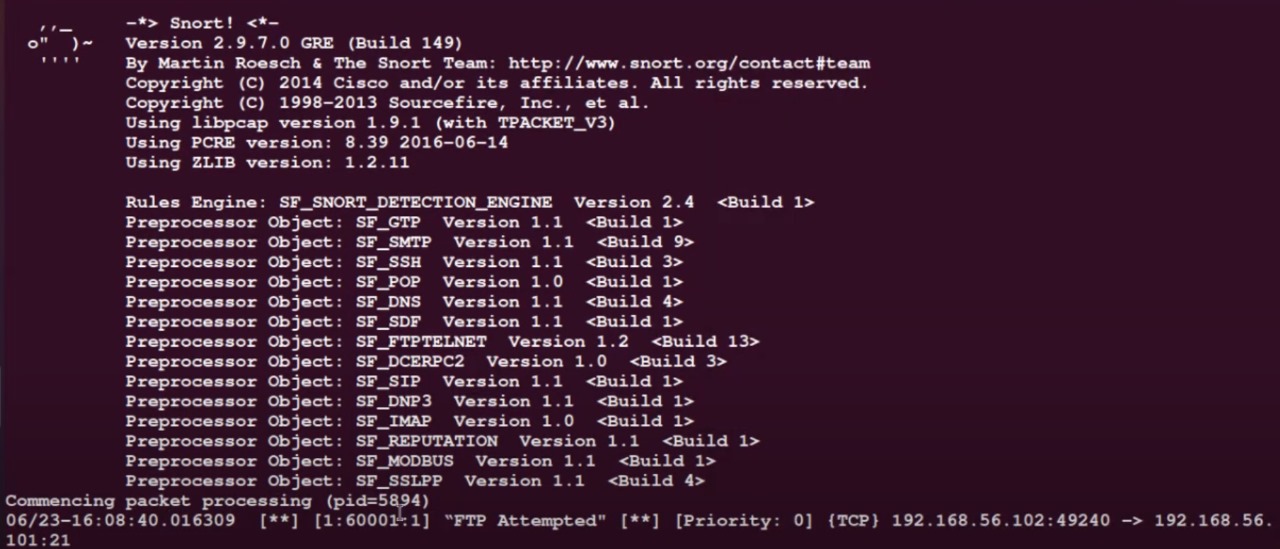


Fig 5.3.7.1

**5.4 RESOURCE DEMAND**

**```**

**comparison\_melted <- pivot\_longer(comparison, cols =** **c(Execution\_Time, Memory\_Usage, Accuracy), names\_to = "Metric", values\_to = "Value")**

**```**

The code performs a thorough examination of the resource needs for several machine learning algorithms, including Gradient Boosting, Random Forest, Decision Tree, and Naive Bayes. It assesses both execution time and memory consumption to quantify their resource usage. The code compares how well they perform in terms of accuracy, execution time, and memory usage to provide information about how effective and efficient they are in comparison. It uses graphic aids, such as bar plots and radar charts, to provide this comparison in a way that is visually engaging and helps the reader better understand the findings. The code concludes by determining which algorithm has the largest area under the curve, which provides a clear indication of the algorithm that performs the best out of all the ones that were assessed.

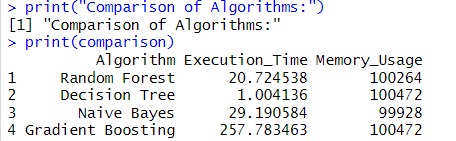


Fig 5.4.1

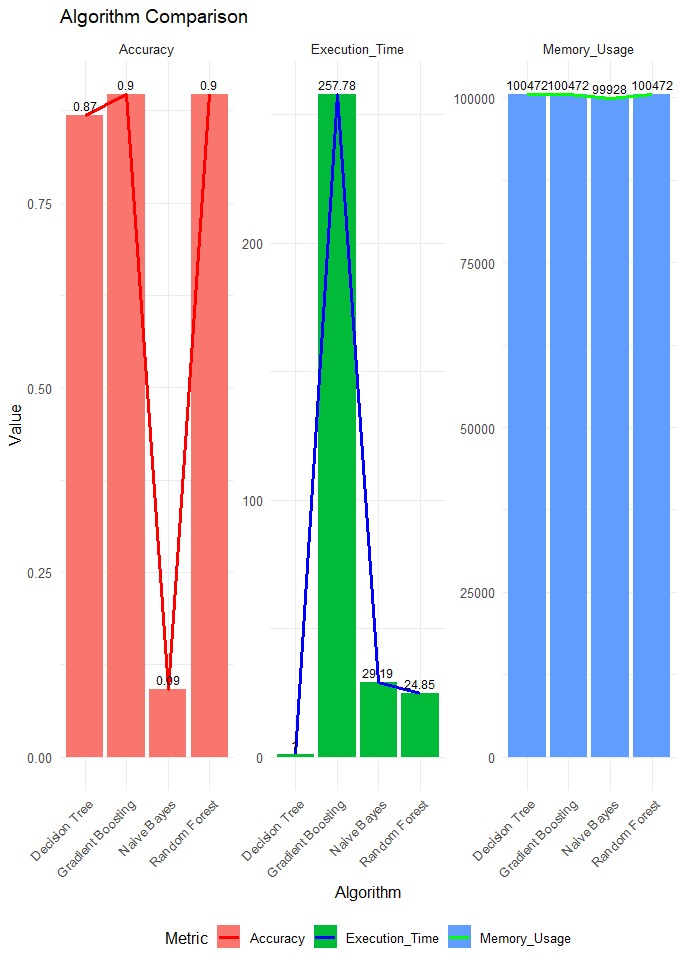


Fig 5.4.2

**CHAPTER 6**

**RESULT**

**CONCLUSION:**

In conclusion, this project has effectively employed a diverse range of machine learning algorithms such as Random Forest, Decision Tree, Naive Bayes, and Gradient Boosting for intrusion detection. Furthermore, it has proposed robust mitigation strategies to tackle identified threats and successfully implemented real-time intrusion detection and prevention using Snort IDS/IPS tools. Additionally, the project has addressed resource demand challenges, optimizing system performance for enhanced security measures. These efforts collectively contribute to a comprehensive approach towards network security, ensuring resilience against evolving cyber threats and safeguarding critical assets.

**FUTURE SCOPE:**

* **Automated Response System:** Implement automated response system that can swiftly adapt to detect intrusion, enabling quick real-time actions to solve the malicious activities.
* **Continuous Monitoring and Evaluation:** Develop a extensive framework for live monitoring and evaluation of network security measures, thus execute flexible security strategies that can respond to evolving threats.
* **Adversarial Attack Detection:** Mainly Concentrated on researching and implementing innovative methods to detect and prevent attacks that aims to deceive intrusion detection system, ensuring resilience against sophisticated threats.
* **Threat Hunting and Response Orchestration:** Enhance the capabilities for proactive threat hunting to identify and address potential security breaches, while also incorporating mechanism to automate the incident response workflow for faster and more effective threat containment.
* **Blockchain Integration:** Examine how blockchain technology can be integrated to provide safe and impenetrable data transfer, improving the security and integrity of network connections.

**CHAPTER 7**

**APPENDICES**

**APPENDIX-1: CODE**

#Loading the dataset

data <- read.csv("C:\\Users\\pavi5\\Downloads\\itrain.csv")

head(data)

View(data)

names(data) #print colnames

summary(data)

# Selecting relevant columns from the dataset

selected\_columns <- c(

"proto", "service", "spkts", "dpkts", "sbytes", "dbytes", "rate", "sload",

"dload", "sloss", "dloss", "sinpkt", "dinpkt", "sjit", "djit", "swin",

"stcpb", "dtcpb", "dwin", "tcprtt", "synack", "ackdat", "smean", "dmean",

"trans\_depth", "response\_body\_len", "ct\_src\_dport\_ltm", "ct\_dst\_sport\_ltm",

"is\_ftp\_login", "ct\_ftp\_cmd", "ct\_flw\_http\_mthd", "is\_sm\_ips\_ports",

"attack\_cat", "label"

)

# Extracting selected columns

selected\_data <- data[, selected\_columns]

# Check for missing values

missing\_values <- colSums(is.na(selected\_data))

print(missing\_values)

View(selected\_data)

summary(selected\_data)

# Install and load necessary libraries

install.packages(c("randomForest", "caret", "ranger", "tidyverse", "rpart", "keras", "e1071"))

library(ranger, randomForest, caret, keras, rpart, e1071, class, tidyverse, ggplot2)

# Convert 'attack\_cat' variable to factor type before splitting the data

selected\_data$attack\_cat <- as.factor(selected\_data$attack\_cat)

# Split the Data into Training and Testing Sets

set.seed(42) # Set seed for reproducibility

split\_index <- createDataPartition(selected\_data$attack\_cat, p = 0.7, list = FALSE)

train\_data <- selected\_data[split\_index, ]

test\_data <- selected\_data[-split\_index, ]

# Parallel Random Forest

rf\_conf\_matrix\_parallel <- confusionMatrix(rf\_predictions\_parallel$predictions, test\_data$attack\_cat)

print(rf\_conf\_matrix\_parallel)

# Parallel Decision Tree

dt\_conf\_matrix\_parallel <- confusionMatrix(dt\_predictions\_parallel$predictions, test\_data$attack\_cat)

print(dt\_conf\_matrix\_parallel)

# Parallel Naive Bayes (using e1071 package)

nb\_conf\_matrix\_parallel <- confusionMatrix(nb\_predictions\_parallel, test\_data$attack\_cat)

print(nb\_conf\_matrix\_parallel)

# Gradient Boosting Machines

gbm\_conf\_matrix\_parallel <- confusionMatrix(gbm\_predictions\_parallel$predictions, test\_data$attack\_cat)

print(gbm\_conf\_matrix\_parallel)

# k-Nearest Neighbors (kNN) Classifier

knn\_model <- knn(train = train\_data\_scaled, test = test\_data\_scaled, cl = train\_data$attack\_cat, k = 5)

print(paste("Random Forest Classifier Accuracy:", metrics\_rf["accuracy"]))

print(paste("Decision Tree Classifier Accuracy:", metrics\_dt["accuracy"]))

print(paste("Naive Bayes Classifier Accuracy:", metrics\_nb["accuracy"]))

print(paste("Gradient Boosting Classifier Accuracy:", metrics\_gbm["accuracy"]))

print(paste("kNN Classifier Accuracy:", knn\_accuracy))

print(paste("kNN Classifier Accuracy:", knn\_accuracy))

# Function to compute precision, recall, and F1-score

compute\_metrics <- function(conf\_matrix) {

TP <- conf\_matrix$table[2, 2]

FP <- conf\_matrix$table[1, 2]

FN <- conf\_matrix$table[2, 1]

return(c(accuracy = accuracy, precision = precision, recall = recall, f1\_score = f1\_score))

}

# Compute metrics for each classifier

metrics\_rf <- compute\_metrics(rf\_conf\_matrix\_parallel)

metrics\_dt <- compute\_metrics(dt\_conf\_matrix\_parallel)

metrics\_nb <- compute\_metrics(nb\_conf\_matrix\_parallel)

metrics\_gbm <- compute\_metrics(gbm\_conf\_matrix\_parallel)

# Print metrics

cat("Random Forest Metrics:\n")

print(metrics\_rf)

cat("\nDecision Tree Metrics:\n")

print(metrics\_dt)

cat("\nNaive Bayes Metrics:\n")

print(metrics\_nb)

cat("\nGradient Boosting Metrics:\n")

print(metrics\_gbm)

# Create a data frame for visualization

results <- data.frame(

Classifier = c("Random Forest", "Decision Tree", "Naive Bayes", "Gradient Boosting"),

Accuracy = c(metrics\_rf["accuracy"], metrics\_dt["accuracy"], metrics\_nb["accuracy"], metrics\_gbm["accuracy"]),

Precision = c(metrics\_rf["precision"], metrics\_dt["precision"], metrics\_nb["precision"], metrics\_gbm["precision"]),

Recall = c(metrics\_rf["recall"], metrics\_dt["recall"], metrics\_nb["recall"], metrics\_gbm["recall"]),

F1\_Score = c(metrics\_rf["f1\_score"], metrics\_dt["f1\_score"], metrics\_nb["f1\_score"], metrics\_gbm["f1\_score"])

)

# Plot using ggplot

ggplot(results\_melted, aes(x = Classifier, y = Value, fill = Metric)) +

geom\_bar(stat = "identity", position = "dodge") +

labs(title = "Classifier Performance Metrics", y = "Value", fill = "Metric") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

**APPENDIX - 1: SCREENSHOT**

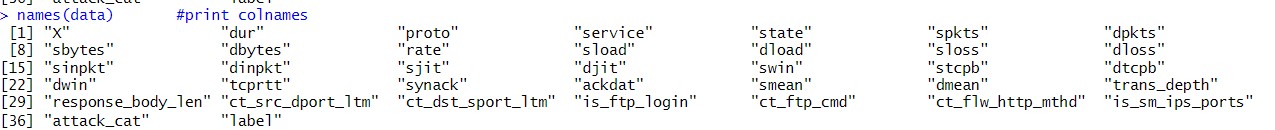


Fig 7.1.1

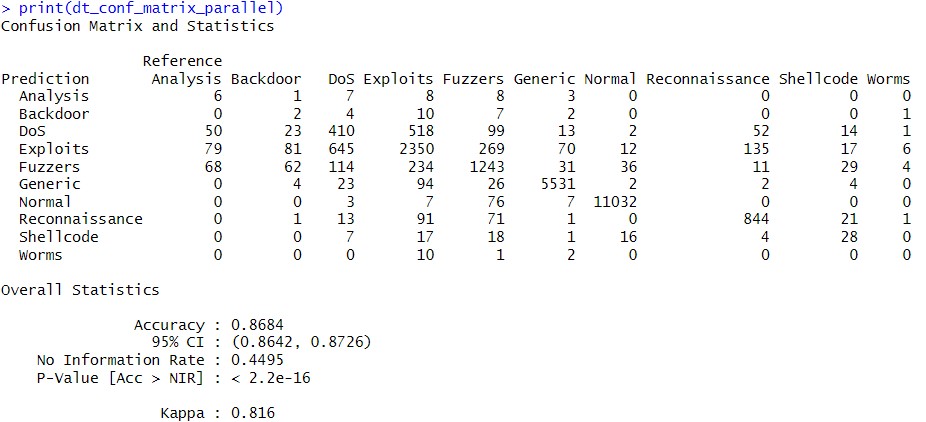


Fig 7.1.2

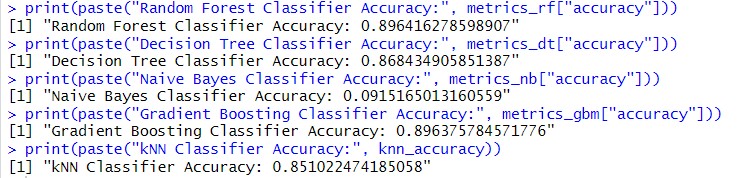


Fig 7.1.3

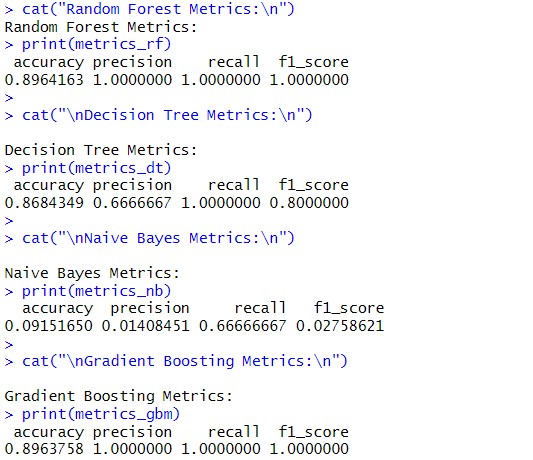


Fig 7.1.4

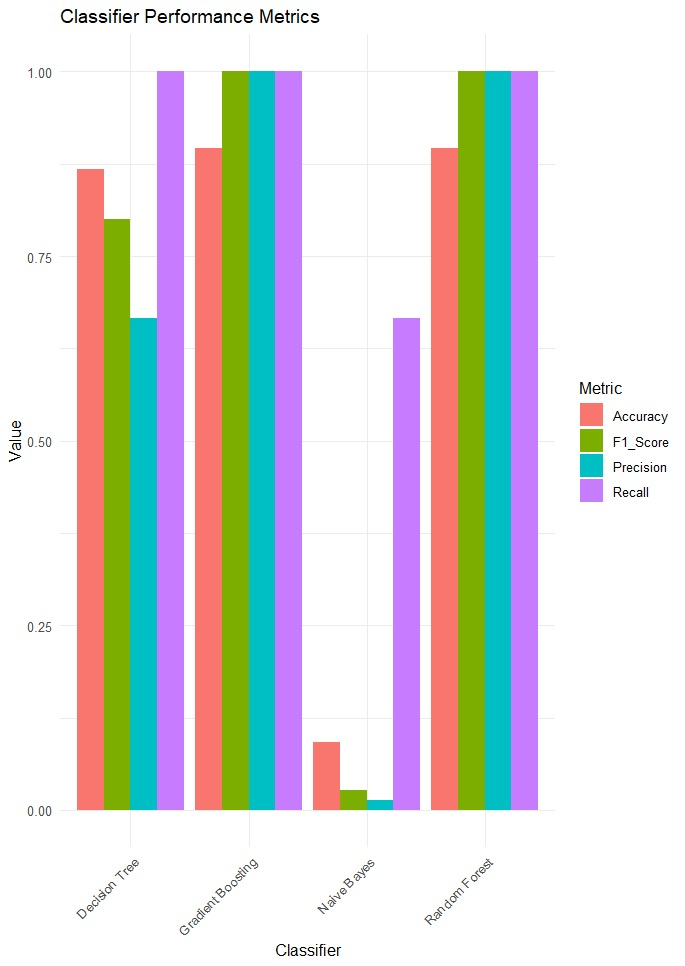
****

Fig 7.1.5

**REFERENCES**

**Journal References:**

1. Hu, Z., Knowledge Graph Based Large Scale Network Security Threat Detection Techniques. *Applied Mathematics and Nonlinear Sciences*, *9*(1).

2. Akhtar, M.S. and Feng, T., 2022. Malware analysis and detection using machine learning algorithms. *Symmetry*, *14*(11), p.2304.

3. Lee, J., Kim, J., Kim, I. and Han, K., 2019. Cyber threat detection based on artificial neural networks using event profiles. *Ieee Access*, *7*, pp.165607-165626.

4. Ashiku, L. and Dagli, C., 2021. Network intrusion detection system using deep learning. *Procedia Computer Science*, *185*, pp.239-247.

5. Thiziers, A.H., Tiémoman, K. and Kabir, T.T.Q., 2023. Malware Detection Using Deep Learning. *Open Journal of Applied Sciences*, *13*(12), pp.2480-2491.

**Web References:**

1. Basics Of Networking:

<https://www.geeksforgeeks.org/basics-computer-networking/>

2. Types of Attacks:  
<https://youtu.be/_zq4qTc9Jmg?si=upijfEn9CdxyeCe4>

3. Classification in Machine Learning (akkio):

<https://www.akkio.com/post/5-types-of-machine-learning-classification-algorithms>

4. Intrusion Detection Dataset (Kaggle): <https://www.kaggle.com/search?q=intrusion+detection+dataset>

**WORKLOG**

|  |  |  |
| --- | --- | --- |
| **Day** | **Date** | **Task Done** |
| Day 1 | 19/02/2024 | Examined the project key aspects, including contributors, tasks undertaken, and relevant trends. |
| Day 2 | 20/02/2024 | Skimmed through basics of Network Threat. |
| Day 3 | 22/02/2024 | Referred some Research papers and Journals. |
| Day 4 | 27/02/2024 | Referred some Research papers and Journals. |
| Day 5 | 29/02/2024 | Identified the primary limitation from logged the research papers and articulated the problem statement along with the objective. |
| Day 6 | 05/03/2024 | With the gathered information we created methodology. |
| Day 7 | 07/03/2024 | Explored ML algorithms (Naive Bayes,DT) and malware fundamentals |
| Day 8 | 12/03/2024 | Dataset Selection and Exploration |
| Day 9 | 14/03/2024 | Preprocess the dataset by handling missing values. Split the dataset into training and testing sets |
| Day 10 | 19/03/2024 | Identify potential threats or anomalies within the dataset using statistical analysis or domain knowledge. |
| Day 11 | 21/03/2024 | Apply anomaly detection techniques such as random forests, one-class SVM, or detect anomalies within the data. |
| Day 12 | 26/03/2024 | Trained the machine learning model on the training dataset. |
| Day 13 | 28/03/2024 | Evaluate the trained model's performance on the testing dataset using metrics such as accuracy. |
| Day 14 | 02/04/2024 | Use the trained model to classify data instances as normal or anomalous. Calculate the percentage of normal and anomaly data within the dataset. |
| Day 15 | 04/04/2024 | Identified the various types of anomalies and Developed strategies to counter each anomaly. |
| Day 16 | 08/04/2024 | Formulated Mitigation plans for enhanced security. Refer the videos for additional insight. |
| Day 17 | 10/04/2024 | Recognised anomaly pattern for targeted defence and utilized video resources to deepen understanding. |
| Day 18 | 11/04/2024 | Used Snort Tool on Ubuntu for threat detection and prevention. |
| Day 19 | 12/04/2024 | Detected and countered threats originating from kali linux and windows |
| Day 20 | 13/04/2024 | Implement proactive measure to safeguard against potential security breaches. |
| Day 21 | 15/04/2024 | A custom function in R was used to continuously measure CPU and memory usage in order to monitor the resource demand for the machine learning model. |