A Project Report on

Detection Of Cyber Attacks in a Network Using Machine Learning Techniques

Submitted in partial fulfillment for award of

Bachelor of Technology

Degree

in

Computer Science and Engineering

By

R. Venkatesh (Y20ACS547) **SK. Shifa Anjum (Y20ACS568)**

T. Narendra (Y20ACS577) **S. Sai Jagadeesh (Y20ACS569)**

Under the guidance of

Mr. J. Madhan Kumar, Asst. Prof.

Designation

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2023-2024

Department of

Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **Detection of Cyber Attacks in a Network using Machine Learning Techniques** that is being submitted by R. Venkatesh (Y20ACS547), SK. Shifa Anjum (Y20ACS568), T. Narendra (Y20ACS577), S. Sai Jagadeesh (Y20ACS569) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide	**Signature of the HOD**
Mr. J. Madhan Kumar	**Dr. M. Rajesh Babu**
Asst. Professor	**Assoc. Prof. & Head**

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

R. Venkatesh (Y20ACS547)

SK. Shifa Anjum (Y20ACS568)

T. Narendra (Y20ACS577)

S. Sai Jagadeesh (Y20ACS569)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide [**Name of the guide**, designation of guide], Department of CSE, for his/her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

R. Venkatesh (Y20ACS547)

SK. Shifa Anjum (Y20ACS568)

T. Narendra (Y20ACS577)

S. Sai Jagadeesh (Y20ACS569)

Abstract

In today's interconnected digital landscape, the threat of cyber attacks looms large, necessitating robust defense mechanisms. This project proposes a novel approach leveraging machine learning (ML) techniques for the detection of cyber attacks within network infrastructures. By harnessing the power of ML algorithms, including supervised, unsupervised, and deep learning methods, the system aims to identify anomalous patterns indicative of malicious activity. The dataset comprises network traffic data, enriched with labeled instances of known attacks for supervised learning, and unlabeled instances for unsupervised techniques. Through feature engineering, dimensionality reduction, and model optimization, the system can effectively discern between normal and malicious network behavior in real-time. The evaluation of the proposed methodology demonstrates promising results in terms of accuracy, precision, recall, and F1-score, highlighting its potential to enhance cybersecurity defenses and mitigate the impact of cyber threats.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

In today's interconnected world, the security of computer networks is of paramount importance. With the increasing prevalence of cyber threats and attacks, organizations and individuals alike face the challenge of safeguarding their networks from malicious actors. Traditional methods of network security, such as firewalls and intrusion detection systems (IDS), are no longer sufficient to defend against the evolving tactics employed by cybercriminals. As a result, there is a growing need for advanced techniques capable of detecting and mitigating network attacks in real-time.

Machine learning (ML) has emerged as a powerful tool in the field of network security, offering the potential to identify anomalous behavior and detect malicious activities with high accuracy. By leveraging large datasets and sophisticated algorithms, ML models can learn to distinguish between normal network traffic and suspicious patterns indicative of an attack. This capability makes ML-based intrusion detection systems (IDS) an attractive option for enhancing the security posture of organizations and minimizing the impact of cyber threats.

In this project, we focus on the task of detecting network attacks using machine learning techniques. Our objective is to develop a robust and effective IDS capable of accurately identifying various types of attacks in network traffic data. To achieve this goal, we leverage the CICIDS2017 dataset, a widely used benchmark dataset in the field of network security. This dataset contains a diverse set of network traffic flows, including both benign and malicious activities, making it well-suited for training and evaluating ML models for intrusion detection.

In the following sections of this report, we provide a detailed overview of our methodology, implementation, and experimental results. We describe the preprocessing steps applied to the CICIDS2017 dataset, the machine learning algorithms used for training our models, and the performance metrics used to evaluate their effectiveness. Additionally, we discuss the insights gained from our experiments, the challenges encountered during the project, and potential avenues for future research.

Overall, this project contributes to the ongoing efforts to enhance network security through the application of machine learning techniques. By developing an effective IDS capable of detecting network attacks, we aim to empower organizations with the tools they need to defend against cyber threats and safeguard their critical assets.

1.1 XGBoost:

XGBoost (Extreme Gradient Boosting) is a popular machine learning algorithm known for its efficiency and effectiveness in supervised learning tasks, particularly in classification and regression problems

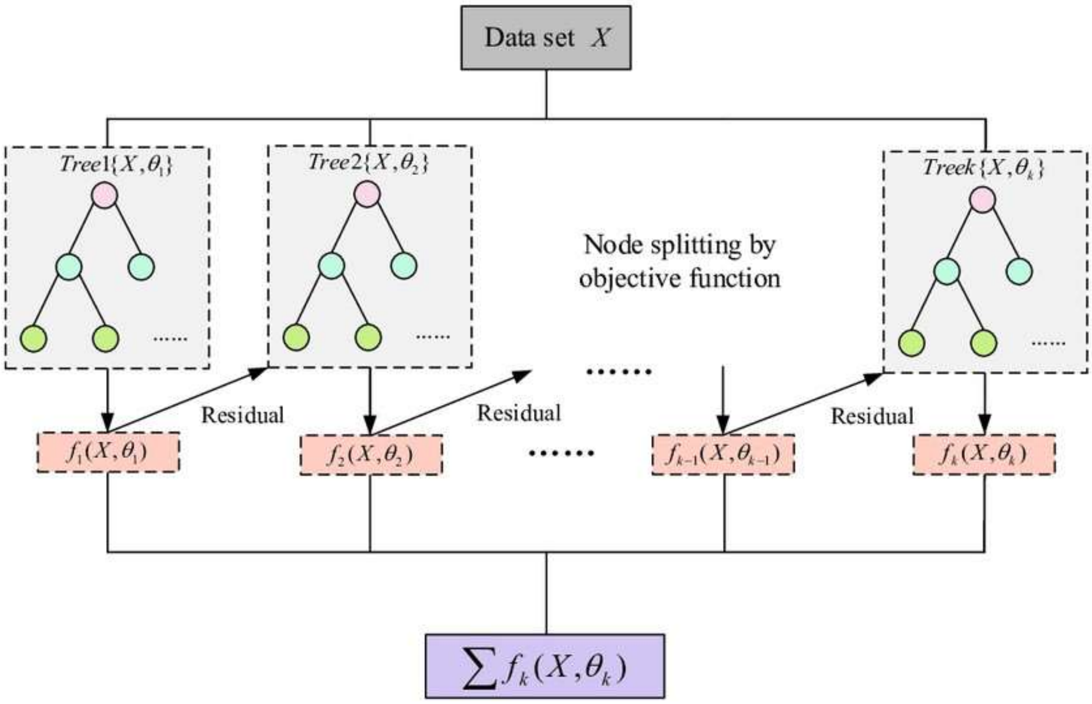


Figure 1.1 XGBoost

1.2 Decision Trees:

Decision trees are fundamental supervised learning models used for both classification and regression tasks. They are intuitive to understand and interpret, making them popular in various domains. Despite their simplicity, decision trees can be powerful models when used appropriately, especially in situations where interpretability and transparency are essential.

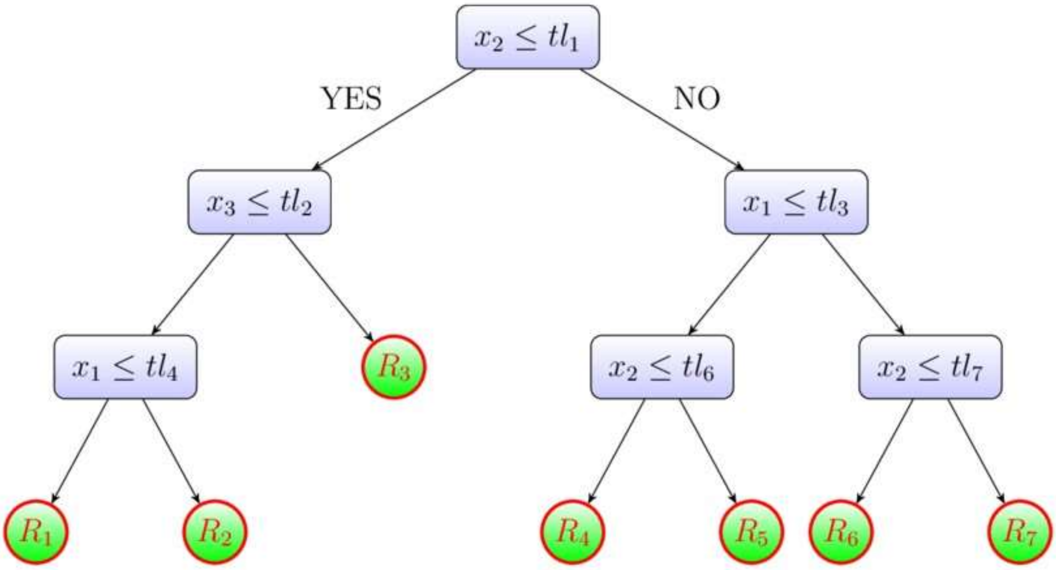


Figure 1.2 Decision Tree

1.3 Random Forest:

Random Forest is a powerful ensemble learning technique widely used in machine learning for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputting the mode (in classification) or mean prediction (in regression) of the individual trees.

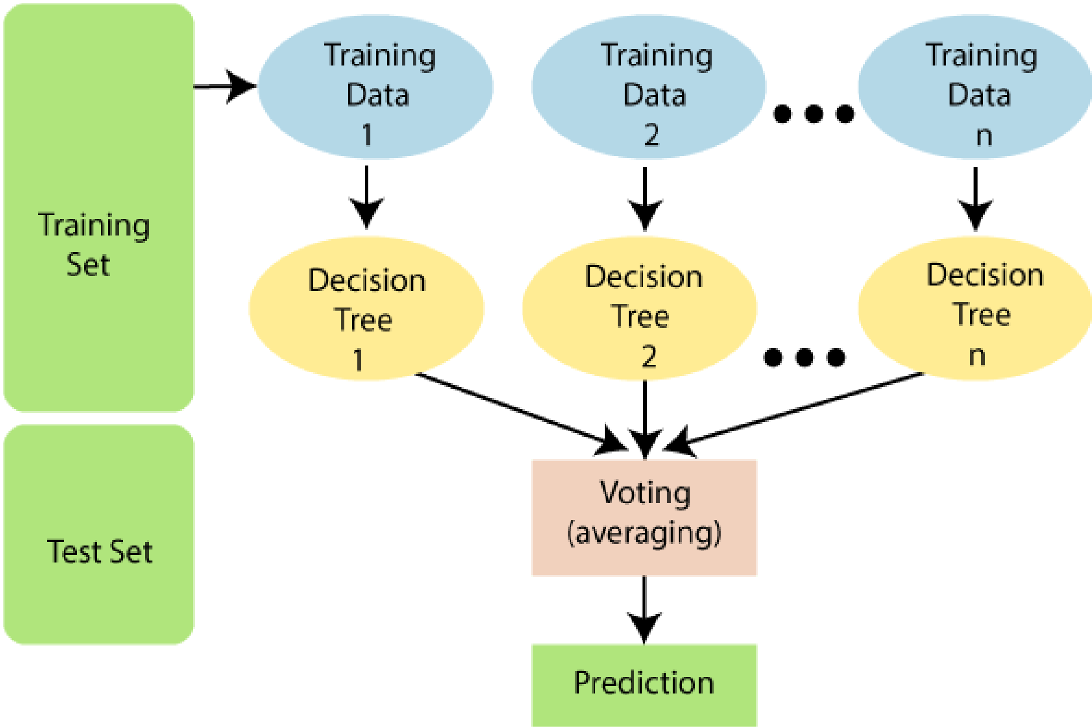


Figure 1.3 Random Forest

1.4 Multi Layer Perceptron

A Multi-Layer Perceptron (MLP) is a type of artificial neural network comprised of multiple layers of interconnected nodes (perceptrons), including an input layer, one or more hidden layers, and an output layer. Each node applies a weighted sum of inputs, followed by an activation function, to produce an output. Through a process of forward propagation, information flows from the input layer through the hidden layers to the output layer, where predictions or classifications are made. MLPs are trained using techniques like backpropagation and gradient descent, adjusting the weights of connections iteratively to minimize the difference between predicted and actual outputs. With their ability to model complex relationships and nonlinearities, MLPs are widely used in various machine learning tasks, including classification, regression, and pattern recognition.

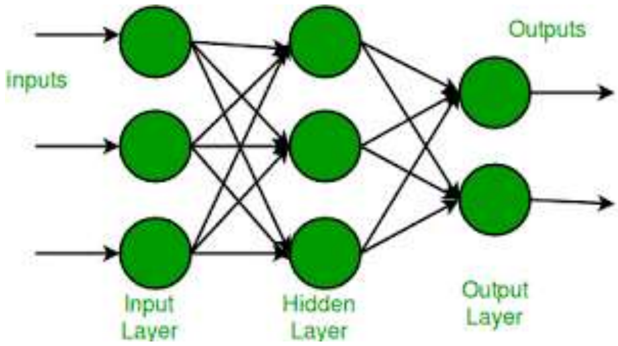


Figure 1.4 Multi Layer Perceptron

1.5 Gradient Boost

Gradient Boosting is a powerful ensemble learning technique used for both regression and classification tasks. It sequentially trains a series of weak learners, typically decision trees, with each subsequent tree focusing on the errors made by its predecessors. It works by fitting the new model to the residual errors of the previous model, effectively reducing the error in predictions with each iteration. This process continues until a predefined number of trees are built or until no further improvements can be made. By combining multiple weak learners into a strong learner, Gradient Boosting often yields highly accurate predictions and is widely utilized in various machine learning applications.

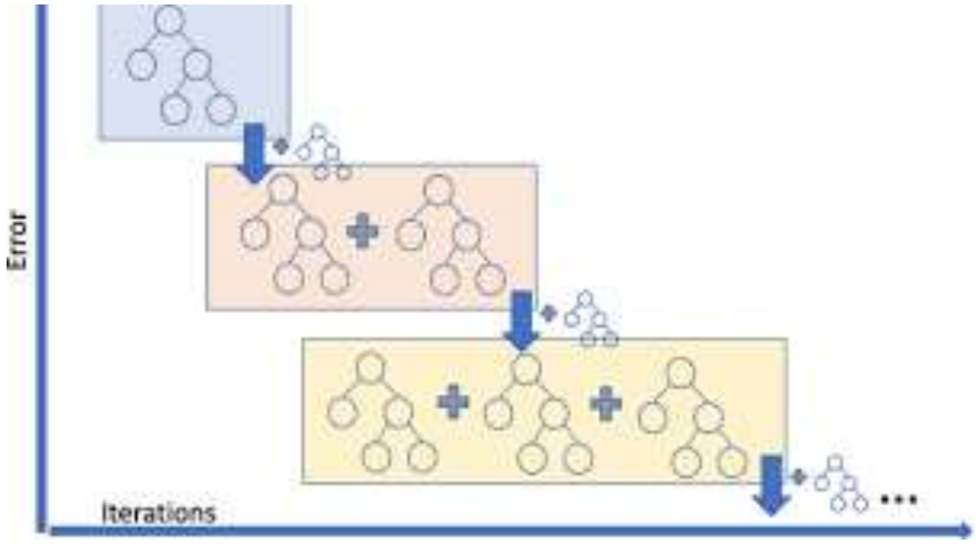


Figure 1.5 Gradient Boost

2 Literature Survey

Attacks on computer networks are devastating and can affect the functioning of the entire system by reading, damaging, and stealing the data [11]. Attacks are preceded by pre-intrusion activities like port scanning and IP Spoofing.

The primary functions of NIDS are packet sniffing, identifying attack signatures, identifying attacks, and reporting attack details. Attacks are identified by capturing features from source and destination IP addresses, ports, protocol details, header details, etc. Based on the nature of attacks, attacks can be classified as passive and active [12]. The passive attack may be system-based or network-based, where the attacker silently monitors the network and try to learn confidential data. Passive attacks are challenging to monitor. Active attackers break all security measures and get into the networks by exploiting security loopholes, masquerading as a trusted system, or stealing passwords.

Fuzzers attack inputs a massive amount of random data to the system to make it fail and find bugs [27]. It can identify software and system vulnerabilities and loopholes in networks and operating systems.

Penetrates the web application with port scanning, spam emails, and web scripts [22]. Machine learning models can identify port scanning by defeating IP Spoofing, altering port scan frequency, and changing the sequence in which ports are scanned. Spam emails are dangerous as they spread malicious code, run phishing scams and make money.

Machine learning models use content-based email filtering, which identifies some keywords that can produce high variance between spam and legitimate emails [6]. Malicious HTML code penetrations have many consequences, like disclosure of cookies, thereby altering the victim's page content.

Backdoor attacks compromise security mechanisms and access computer and their data [22]. This attack targets the privacy and availability of computing resources to users [25].

DoS attacks make network resources unavailable to the user by suspending service [22]. Verisign reports a massive increase in frequency and complexity of DoS attacks which demand strong NIDs using machine learning and deep learning models.

The attacker exploits the vulnerability of software or operating system, takes control of computer resources or network data, and results in system crashes or malfunctions. Zero-day exploits take advantage of software vulnerability about which vendors are unaware.

Generic attacks work against block ciphers without considering the internal structure of block ciphers [22]. Since the length of the key and blocks are limited, all block ciphers are under the threat of generic attacks. Generic attacks are detected by choosing appropriate external parameters. Different generic attacks on block ciphers are exhaustive key search, dictionary attack, rainbow table attack, etc. [7].

Reconnaissance attacks gather all possible information about the target system before launching the actual attack, and it acts as the preparation tool for the actual attack. The three main types of reconnaissance attacks are social, public, and software reconnaissance. During this attack, information is gathered by packet sniffing, port scanning, sweeping the ping, and queries regarding internet information [28]

Shellcode is a small piece of code used as the payload in the exploitation of software vulnerability. It runs a command interpreter that interactively enters commands to be executed on the vulnerable systems and reads back the output [3]. Shellcode attacks can be detected using run-time heuristics representing machine-level operations.

Worms replicate and spread to other computing resources by exploiting their security failures. Early warning and less reaction time for counteractions are two expected features of the worm detection system. It considers payload content and format, packet headers, network traffic, and monitoring host behavior for worm detection [19]

Almutairi et al., proposed a four-component NIDS consisting of an Intrusion Detection System, frequent signature database, updating agent, and complimentary signature database [1]. IDS extracts signature from network packets, compare them with signature databases, and trigger an alert if a match occurs. This four-component system ensures early and accurate detection of attacks with fewer false positives. Attacks with infrequent signatures are also caught with the signatures kept in the complementary database. False alarm minimization is the main issue to be addressed in the signature-based detection and can be solved using signature enhancement, state-full signatures, and vulnerability signatures [14].

Moustafa et al., performed statistical analysis of the observations and features using the Kolmogorov-Smirnov test, Multivariate skewness, and Multivariate kurtosis. Supervised feature correlation with Gain Ratio and unsupervised correlation with

Pearson’s correlation coefficient was also performed to measure the relevance between features. Finally, the UNSW-NB15 dataset complexities are evaluated with existing classifiers with metrics accuracy and false alarm rate. The decision tree classifier performed well with an accuracy of 85.56% and 15.78% false alarm rate [23].

Meftah et al., proposed anomaly-based NIDS with machine learning techniques. Random forest with 10-fold cross validation to assign the index of feature significance in reducing impurity in the whole forest. The top features of UNSW-NB15 Dataset are ct dst src ltm, ct srv dst, ct dst sport ltm, ct src dport ltm, ct srv src. Support vector machine with an accuracy of 82.11% outperformed Logistic Regression and Gradient Boost Machine in binary classification model for attack detection. For identifying the type of attack, the multi-classification model with Decision Tree C5.0, outperformed Naive Bayes and Support vector machine [20].

Peng et al., proposed Deep Neural Network(DNN)k with five hidden layers to identify attacks (Normal, DoS, Probe Categories, R2L, U2R) with NSL-KDD Dataset and compared the performance with Machine Learning models (Support Vector Machines, Random Forest, Linear Regression Models). DNN produced satisfactory results for identifying Normal, Dos, and Prob categories. SVM performed well in detecting Normal and four attacks. Random forest and linear regression also performed well in identifying network attacks [24].

The previous research on UNSW-NB15 dataset includes learning of machine learning and deep learning models on selected features, which decreases the performance of the model since the cardinality of the feature set is only 47 which is not all huge and the relevance of each feature is very significant. Regarding the deep neural network, works of literature are very limited and those works have addressed only a limited number of attacks. In this research four classical, three ensemble machine learning models, and deep multi-layer perceptron models are designed to identify network attacks.

3 PROPOSED SYSTEM

3.1 EXISTING SYSTEM:

3.1.1 Traditional Approaches:

- Reliance on rule-based and signature-based systems for network attack detection.

3.1.2 Limitations:

- Vulnerable to zero-day attacks due to predefined patterns.
- Struggles to adapt to the evolving landscape of cyber threats.

3.1.3 Challenges:

- Difficulty in keeping pace with rapidly changing attack methodologies.
- Limited dynamic and proactive response to emerging security risks.

3.2 PROPOSED SYSTEM:

- The proposed system uses machine learning models to detect any attacks in the network.
- Various classification models are to be trained and tested to find out the best performing model which can adapt to any attack that can be possible on the network.

- When an attack is detected, the system blocks the user who appears to be performing an attack and notifies the administrator about the attack for further review.

4 Requirements

4.1 Hardware Requirements

- X64/X86 based processor
- Ethernet/Wi-Fi capable computer

4.2 Software Requirements

- Python: pandas, numpy, scikit-learn==1.2.2, joblib==1.2.0, xgboost, scapy, psutil
- Any Desktop OS

5 DESIGN

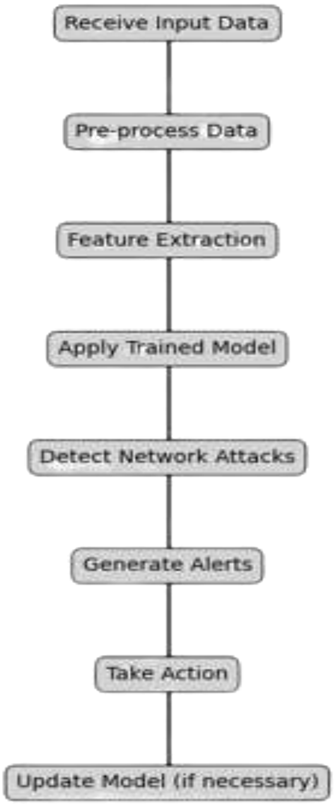


Figure 5.1 System Design

In the context of detecting cyber attacks in a network using machine learning, the first step is to receive input data. There are several ways to receive input data such as capturing network traffic data from various sources such as routers, switches, or network monitoring tools.

The next step is to pre-process data which is done through various processes like data collection, data cleaning by gathering data from various sources such as network traffic logs, system logs, firewall logs, intrusion detection system (IDS) alerts, and any other relevant sources.

The next step is feature extraction which can be done through transforming raw network data into a structured set of features that capture important characteristics of network traffic and potential cyber attacks. These features serve as input to machine learning models for training and detection of anomalous behavior in the network.

The next step is to apply trained model which helps to detect and mitigate cyber attacks in their networks, thereby enhancing their overall cybersecurity posture.

The next step is to detect network attacks by evaluating the performance of the trained models using validation and test datasets. Metrics such as accuracy, precision, recall, F1-score and confusion matrix are used to assess the model's ability to correctly classify normal and malicious network traffic.

The next step is to generate alerts. Generating alerts in the context of detecting cyber attacks in a network involves automatically notifying system administrators or security personnel when suspicious or malicious activity is detected.

The next step is to take action. Taking action in the context of detecting cyber attacks in a network using machine learning involves responding to alerts and implementing measures to mitigate the impact of detected threats.

The final step is to update the model only if it is necessary. It is essential for maintaining its effectiveness and adapting to evolving threats and to ensure that their detection capabilities remain robust and adaptive to evolving cyber threats in the network environment.

6 Code Implementation

6.1 Importing Dependencies:

6.1.1 Description:

This block imports all the necessary modules and packages required for the functioning of the script.

6.1.2 Details:

- `scapy.sendrecv`: Imports the `sniff` function from Scapy for packet sniffing.
- `traceback`: Used for printing exception tracebacks for debugging purposes.
- `PacketInfo` and `Flow` classes are imported from `PacketInfo.py` and `Flow.py` files, respectively, located in the `flow` directory. These classes handle packet and flow information.
- `joblib`, `pandas`, `json`, `os`, `platform`, and `xgboost` are standard Python libraries for various functionalities such as joblib for model loading, pandas for data manipulation, json for JSON file handling, os for system operations, platform for identifying the operating system, and xgboost for XGBoost model support.

6.2 Blocking Function:

6.2.1 Description:

Defines a function to block traffic from a specified IP address based on the operating system.

6.2.2 Details:

- Checks the current operating system using `platform.system()`.

- Uses conditional statements to execute different commands for blocking traffic based on the operating system.
- For Linux, it uses `iptables` commands to block traffic.
- For Windows, it utilizes `netsh advfirewall` commands to add rules to the Windows Firewall.
- For macOS, it employs `pfctl` commands to manipulate the Packet Filter firewall.

6.3 Loading Models and Labels:

6.3.1 Description:

Loads machine learning models and labels required for predicting network attacks.

6.3.2 Details:

- Loads the labels from a JSON file named `labels.json` and stores them in a dictionary format using the `format_label_dict` function.
- Loads the column names from a text file named `cols.txt` and splits them by newline character to get a list of column names.
- Loads the trained machine learning models using `joblib.load()` function. Models include Decision Tree (`dt`), Random Forest (`rf`), XGBoost (`xgb`), Gradient Boosting (`gbc`), and Multilayer Perceptron (`mlp`). Optionally, a scaler model can be loaded if required (commented out).

6.4 Feature Prediction:

6.4.1 Description:

Defines a function to predict network attacks using machine learning models.

6.4.2 Details:

- Constructs a DataFrame from the input features (packet information) using the column names loaded earlier.
- Predicts the attack probability using each loaded machine learning model.
- Aggregates the predictions and selects the maximum value.
- If an attack is predicted (i.e., the maximum prediction value is not zero), it retrieves the attack source IP address from the features and blocks traffic from that IP address using the `block_user` function.

6.5 Packet Handling:

6.5.1 Description:

Defines a function to handle incoming network packets.

6.5.2 Details:

- Instantiates a `PacketInfo` object to extract packet information and sets its attributes.
- Checks if the packet belongs to an existing flow based on its forward or backward flow ID.
- Updates the existing flow or creates a new flow based on the packet information.
- Handles exceptions and prints traceback information if any error occurs during packet processing.

6.6 Flow Management:

6.6.1 Description:

Initializes flow management parameters and defines the main function for packet sniffing and attack detection.

6.6.2 Details:

- `FlowTimeout` specifies the timeout threshold (in seconds) for an active flow. If a flow remains inactive for longer than this threshold, it is considered terminated.
- `current_flows` is a dictionary to store currently active flows.

6.7 Sniffing and Detection Loop:

6.7.1 Description:

Defines the main function for packet sniffing and attack detection.

6.7.2 Details:

- Calls the `sniff` function from Scapy to capture network packets.
- Processes each packet using the `newPacket` function for extracting features and managing flows.
- Periodically checks for terminated flows and predicts potential attacks using the `predict` function.

6.8 Conclusion:

The `sniffer.py` script integrates packet sniffing, feature extraction, flow management, machine learning-based attack detection, and traffic blocking functionalities to enhance network security. It continuously monitors network traffic, identifies potential attacks using trained machine learning models, and takes proactive measures to block malicious traffic sources. This comprehensive explanation provides insights into how each section of the `sniffer.py` script contributes to its overall functionality in monitoring and securing network traffic.

7 Results and Analysis

7.1 Results

7.1.1 Experimental Setup

We conducted experiments using the CICIDS2017 dataset to train multiple machine learning models for network intrusion detection. The models evaluated include Decision Trees (DT), Multi-Layer Perceptron (MLP), Random Forest (RF), XGBoost (XGB), Gradient Boosting (GradBoost), and an ensemble model.

7.1.2 Performance Metrics

We assessed the performance of each model using standard metrics including accuracy, precision, recall, and F1-score.

7.1.2.1 Accuracy:

Accuracy measures the proportion of correctly classified instances out of the total instances. It is calculated as the ratio of the number of correct predictions to the total number of predictions. Accuracy can be a misleading metric when dealing with imbalanced datasets.

7.1.2.2 Precision:

Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It indicates the model's ability to avoid false positives. Precision is calculated as the ratio of true positives to the sum of true positives and false positives.

7.1.2.3 Recall:

Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It indicates the model's ability to capture all positive instances. Recall is calculated as the ratio of true positives to the sum of true positives and false negatives.

7.1.2.4 F1-score:

The F1-score is the harmonic mean of precision and recall. It provides a single score that balances both precision and recall. F1-score is useful when you want to seek a balance between precision and recall, especially in scenarios where there is an uneven class distribution.

7.1.3 Results Analysis

The results of our experiments are summarized in the tables below:

7-1 Evaluation Metrics

Model	Accuracy	Precision	Recall	F1 Score
DT	0.9986	0.9986	0.9986	0.9986
MLP	0.8841	0.8640	0.8841	0.8726
RF	0.9985	0.9985	0.9985	0.9985
XGB	0.9987	0.9987	0.9987	0.9987
GradBoost	0.9858	0.9960	0.9858	0.9907
Ensemble	0.9987	0.9986	0.9987	0.9986

From the performance metrics, we observe that Decision Trees, Random Forest, XGBoost, and the ensemble model achieve high accuracy, precision, recall, and F1-

score values, indicating their effectiveness in detecting network intrusions. However, the Multi-Layer Perceptron model exhibits lower performance compared to other models.

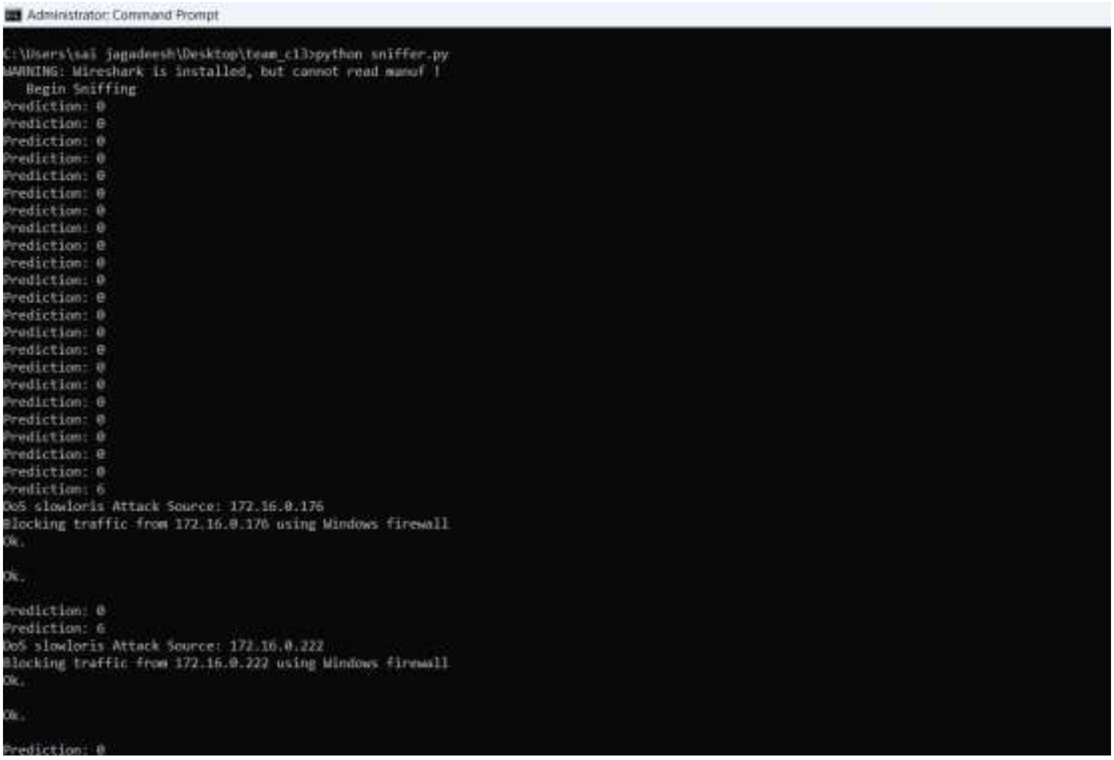
7-2 Training and Testing Time

Model	Training Time (s)	Testing Time (s)
DT	174.73	0.21
MLP	780.72	2.82
RF	71.49	0.94
XGB	447.64	7.28
GradBoost	43.94	3.38
Ensemble	0.00	18.52

Regarding computational efficiency, there are significant variations in training and testing times across different models. The Decision Trees and Random Forest models demonstrate relatively shorter training times, while the Multi-Layer Perceptron model requires the longest training time. During testing, the Decision Trees model exhibits the shortest inference time, followed by Random Forest, Gradient Boosting, XGBoost, and Multi-Layer Perceptron models. Surprisingly, the ensemble model shows a longer testing time compared to individual models, suggesting potential overhead from model aggregation.

7.1.4 Executing the Code

Running the code from the command prompt using administrator privileges to scan the network for any attacks



```
Administrator: Command Prompt
C:\Users\sai jagadeesh\Desktop\team_c13>python sniffer.py
WARNING: Wireshark is installed, but cannot read manuf !
   Begin Sniffing
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 0
Prediction: 6
DoS slowloris Attack Source: 172.16.0.176
Blocking traffic from 172.16.0.176 using Windows firewall
Ok.

Ok.

Prediction: 0
Prediction: 6
DoS slowloris Attack Source: 172.16.0.222
Blocking traffic from 172.16.0.222 using Windows firewall
Ok.

Ok.

Prediction: 0
```

Figure 7.1 Running the code

```
C:\Users\sai jagadeesh\Desktop\team_c13>netsh advfirewall firewall show rule name="Block 172.16.0.176"

Rule Name:                            Block 172.16.0.176
----------------------------------------------------------------------
Enabled:                              Yes
Direction:                            In
Profiles:                             Domain,Private,Public
Grouping:
LocalIP:                              Any
RemoteIP:                             172.16.0.176/32
Protocol:                             Any
Edge traversal:                       No
Action:                               Block
```

Figure 7.2 Checking Firewall Rules

7.2 Future Work

Using machine learning techniques for cyber attack detection in a network is a promising approach. You can explore supervised learning algorithms like K-Nearest Neighbors, Deep Neural Networks etc., trained on labeled datasets of normal and attack traffic. Unsupervised techniques like clustering or anomaly detection can also be effective for identifying unusual patterns that might indicate an attack. Additionally, consider leveraging deep learning methods for more complex data representations and feature extraction. Regular updating and fine-tuning of your models will be crucial to adapt to evolving cyber threats.

8 Bibliography

[1] Almutairi, A.H., Abdelmajeed, N.T., 2017. Innovative signature based intrusion detection system: Parallel processing and minimized database, in: 2017 International Conference on the Frontiers and Advances in Data Science (FADS), IEEE. pp. 114–119.

[2] Ammar, A., et al., 2015. A decision tree classifier for intrusion detection priority tagging. Journal of Computer and Communications 3, 52.

[3] Arce, I., 2004. The shellcode generation. IEEE security & privacy 2, 72–76.

[4] Belgiu, M., Dragut¸, L., 2016. Random forest in remote sensing: A review of applications and future directions. ISPRS journal of photogram- ˘ metry and remote sensing 114, 24–31.

[5] Cunningham, P., Delany, S.J., 2021. k-nearest neighbour classifiers-a tutorial. ACM Computing Surveys (CSUR) 54, 1–25.

[6] Dada, E.G., Bassi, J.S., Chiroma, H., Adetunmbi, A.O., Ajibuwa, O.E., et al., 2019. Machine learning for email spam filtering: review, approaches and open research problems. Heliyon 5, e01802.

[7] De Canniere, C., Biryukov, A., Preneel, B., 2006. An introduction to block cipher cryptanalysis. Proceedings of the IEEE 94, 346–356.

[8] Dhanya, K., Dheesha, O., Gireesh Kumar, T., Vinod, P., 2020. Detection of obfuscated mobile malware with machine learning and deep learning models, in: Symposium on Machine Learning and Metaheuristics Algorithms, and Applications, Springer. pp. 221–231.

[9] Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences 55, 119–139.

[10] Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics , 1189–1232.

[11] Gandhi, M., Srivatsa, S., 2008. Detecting and preventing attacks using network intrusion detection systems. International Journal of Computer Science and Security 2, 49–60.

[12] Garuba, M., Liu, C., Fraites, D., 2008. Intrusion techniques: Comparative study of network intrusion detection systems, in: Fifth International Conference on Information Technology: New Generations (itng 2008), IEEE. pp. 592–598.

[13] Gascon, H., Orfila, A., Blasco, J., 2011. Analysis of update delays in signature-based network intrusion detection systems. Computers & Security 30, 613–624.

[14] Hubballi, N., Suryanarayanan, V., 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. Computer Communications 49, 1–17.

[15] Jing, D., Chen, H.B., 2019. Svm based network intrusion detection for the unsw-nb15 dataset, in: 2019 IEEE 13th international conference on ASIC (ASICON), IEEE. pp. 1–4.

[16] Kumar, V., Sangwan, O.P., 2012. Signature based intrusion detection system using snort. International Journal of Computer Applications & Information Technology 1, 35–41.

[17] Lee, B., Amaresh, S., Green, C., Engels, D., 2018. Comparative study of deep learning models for network intrusion detection. SMU Data Science Review 1, 8.

[18] Lee, C.H., Su, Y.Y., Lin, Y.C., Lee, S.J., 2017. Machine learning based network intrusion detection, in: 2017 2nd IEEE International conference on computational intelligence and applications (ICCIA), IEEE. pp. 79–83.

[19] Li, P., Salour, M., Su, X., 2008. A survey of internet worm detection and containment. IEEE Communications Surveys & Tutorials 10, 20–35.

[20] Meftah, S., Rachidi, T., Assem, N., 2019. Network based intrusion detection using the unsw-nb15 dataset. International Journal of Computing and Digital Systems 8, 478–487.

[21] Moustafa, N., Slay, J., 2015a. The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems, in: 2015 4th international

workshop on building analysis datasets and gathering experience returns for security (BADGERS), IEEE. pp. 25–31.

[22] Moustafa, N., Slay, J., 2015b. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 military communications and information systems conference (MilCIS), IEEE. pp. 1–6.

[23] Moustafa, N., Slay, J., 2016. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. Information Security Journal: A Global Perspective 25, 18–31.

[24] Peng, Y., Su, J., Shi, X., Zhao, B., 2019. Evaluating deep learning based network intrusion detection system in adversarial environment, in: 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), IEEE. pp. 61–66.

[25] Rieger, P., Nguyen, T.D., Miettinen, M., Sadeghi, A.R., 2022. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. arXiv preprint arXiv:2201.00763 .

[26] Sugunan, K., Gireesh Kumar, T., Dhanya, K., 2018. Static and dynamic analysis for android malware detection, in: Advances in Big Data and Cloud Computing. Springer, pp. 147–155.

[27] Thanh, H.N., Van Lang, T., 2020. Evaluating effectiveness of ensemble classifiers when detecting fuzzers attacks on the unsw-nb15 dataset. Journal of Computer Science and Cybernetics 36, 173–185.

[28] Uma, M., Padmavathi, G., 2013. A survey on various cyber attacks and their classification. Int. J. Netw. Secur. 15, 390–396.

[29] Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. Ieee Access 7, 41525–41550.

[30] Vinayakumar, R., Soman, K., Poornachandran, P., 2017. Applying convolutional neural network for network intrusion detection, in: 2017 International Conference on

Advances in Computing, Communications and Informatics (ICACCI), IEEE. pp. 1222–1228.

[31] Wang, W., Jian, S., Tan, Y., Wu, Q., Huang, C., 2022. Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions. Computers & Security 112, 102537.

[32] Yang, H., Cheng, L., Chuah, M.C., 2019. Deep-learning-based network intrusion detection for scada systems, in: 2019 IEEE Conference on Communications and Network Security (CNS), IEEE. pp. 1–7.

[33] Yang, Y., Li, J., Yang, Y., 2015. The research of the fast svm classifier method, in: 2015 12th international computer conference on wavelet active media technology and information processing (ICCWAMTIP), IEEE. pp. 121–124