Project Report
CSE 327

Tittle-Network Intrusion Category Detection

Group Members -

NAME	ID
SOJIBUL ISLAM SOJIB	1921331642
FAHIM FOYSAL APURBA	1921442642
MD. MEHEDI HASAN	1921421642

Acknowledgement

First and foremost, I would like to thank Allah (SWT) for granting me permission to work on this project and for providing me with a bountiful supply of resources.

I would like to express my gratitude to Md. Sazzad Hossain,my instructor for his advice and ongoing assistance in finishing this undertaking. He provided me with some helpful advice on how to improve the project and how to do effective research. My supervisor's patience and support have motivated me to be more imaginative in this assignment.Even though I don't often see him, his words never leave my ears. We are grateful for this wonderful chance.

Thank you also to all my fellow friends that help me a lot in completing this project. They always give advice and guidance to make sure that we will do excellent work. Lastly, I would like to thank all of the people who have contributed directly or indirectly towards the success of this project.

1 Introduction

1.1 Background

Detecting cyber attacks is a crucial aspect of maintaining the security and integrity of computer systems and networks. Over the years, various projects and initiatives have been developed to improve the detection capabilities and respond effectively to cyber threats. Here is a background on some key aspects of detecting cyber attacks projects: Intrusion Detection Systems (IDS): Intrusion Detection Systems are software or hardware-based solutions designed to monitor network traffic and identify any suspicious or malicious activities. IDS can be classified into two main types: network-based (NIDS) and host-based (HIDS). NIDS monitor network traffic, while HIDS analyze activities on individual hosts or devices. These systems use different techniques such as signature-based detection, anomaly detection, and behavioral analysis to identify potential cyber attacks. Security Information and Event Management (SIEM): SIEM systems collect and

analyze security events and logs from various sources within a network. They provide a centralized view of the security posture and enable real-time monitoring, threat detection, and incident response. SIEM solutions typically employ correlation rules, machine learning algorithms, and statistical analysis to identify patterns or anomalies that could indicate a cyber attack. Machine Learning and Artificial Intelligence (AI): Machine learning and AI techniques have gained prominence in cyber attack detection. These technologies can analyze large volumes of data, identify patterns, and detect anomalies that may signify a cyber attack. Machine learning models can be trained on historical data to recognize known attack patterns and improve detection accuracy over time. AI-based systems can also adapt and learn from new threats to enhance their detection capabilities. Threat Intelligence: Threat intelligence involves gathering information about potential cyber threats and attackers to understand their motives, methods, and indicators of compromise. Threat intelligence feeds provide real-time data on known malicious entities, attack techniques, and vulnerabilities. Integrating threat intelligence into detection systems enhances their ability to identify and respond to emerging cyber threats effectively. Collaborative Defense: Many organizations and security communities work together to share threat information and collaborate on cyber attack detection. Initiatives such as Information Sharing and Analysis Centers (ISACs), Computer Emergency Response Teams (CERTs), and industry-specific forums facilitate the exchange of threat intelligence, best practices, and incident response strategies. Collaboration helps in detecting attacks more efficiently by leveraging collective knowledge and resources. Big Data Analytics: With the ever-increasing volume of data generated in today's digital landscape, big data analytics plays a crucial role in cyber attack detection. Analyzing vast amounts of structured and unstructured data can uncover hidden patterns and indicators of compromise. Big data platforms enable real-time processing, storage, and analysis of diverse data sources, enabling organizations to detect and respond to cyber attacks more effectively. Threat Hunting: Threat hunting is a proactive approach to cyber attack detection that involves actively searching for threats within an organization's network or systems. It combines human expertise and automated tools to identify advanced and persistent threats that may evade traditional security measures. Threat hunters use various techniques, including log analysis, behavior analysis, and penetration testing, to detect and mitigate cyber attacks before they cause significant damage. These projects and approaches collectively contribute to the ongoing effort to enhance the detection of cyber attacks. As cyber threats evolve, organizations and security professionals continually refine their strategies and leverage new technologies to stay ahead of attackers and protect critical
systems and data.

1.2 Statement of the problem

It is difficult or almost impossible to develop an intrusion detection system with 100 percent success rate. Most systems today have a lot of security flaws. Not all kinds of intrusions are known. Also, hackers are figuring out new ways into the networks using machine learning techniques . Quick detection of these attacks will help to identify possible intruders. and limit damage effected. So, developing an efficient and accurate intrusion detection system will help to reduce network security threats.

1.3 Objectives

There are several intrusion detection systems in use today . Researchers are trying to develop systems which use machine learning techniques to identify the signature of attackers [6]. This proposed system aims to find a suitable novel technique to be used as a backend to such a system.

1.5 Proposed system

proposed system for a cyber attack detection project. Here's a high-level overview of the components and steps you can consider: Data Collection: Gather relevant data from various sources, such as network logs, system logs, firewall logs, intrusion detection system (IDS) alerts, and any other relevant sources. This data will serve as the input for your detection system. Data Preprocessing: Clean and preprocess the collected data to remove noise, normalize values, and handle missing data. This step ensures that the data is in a suitable format for analysis. Feature Extraction: Extract relevant features from the preprocessed data. These features can include network traffic patterns, login attempts, system resource usage, and any other indicators of potential cyber attacks. Domain knowledge and machine learning techniques can be used to identify informative features. Model Development: Train machine learning or deep learning models using the extracted features. This step involves selecting an appropriate algorithm, splitting the data into training and testing sets, and optimizing the model's parameters. Some common algorithms for cyber attack detection include anomaly detection methods (e.g., Isolation Forest, One-Class SVM) and supervised learning techniques (e.g., Random Forest, Support Vector Machines). Real-Time Monitoring: Deploy the trained model in a real-time monitoring system. This system continuously collects new data and applies the trained model to detect potential cyber attacks in real time. It should be capable of handling high volumes of data and providing prompt alerts when an attack is detected. Alert Generation: When a potential cyber attack is detected, generate alerts or notifications to inform the appropriate personnel or security teams. These alerts can include details about the type of attack, affected systems, and recommended actions to mitigate the threat. Incident Response: Develop an incident response plan to guide the actions that should be taken when an attack is detected.

This plan should include steps for containment, eradication, and recovery from the attack. Automate and integrate incident response procedures into the system to enable a swift response. Continuous Improvement: Monitor the performance of the detection system over time and collect feedback from security analysts. Use this feedback to refine and improve the system's accuracy, adjust detection thresholds, and incorporate new attack patterns or emerging threats. Remember that this is a high-level overview, and the specifics of each step will depend on the scope and requirements of your project. It's important to continuously evaluate and update your system to stay ahead of evolving cyber threats.

1.6 Benefits or Significance of the project

The project of network intrusion detection brings several benefits to organizations and individuals. Some of the key advantages include:

1. Enhanced Security: Network intrusion detection systems (NIDS) help identify and alert against potential threats and malicious activities in real-time. By monitoring network traffic and analyzing patterns, NIDS can detect unauthorized access attempts, malware, and other suspicious activities. This proactive approach enhances the overall security posture of the network.

2. Early Threat Detection: Network intrusion detection systems can detect and respond to threats at an early stage, minimizing the potential damage caused by intrusions. By identifying and addressing security incidents promptly, organizations can prevent data breaches, service disruptions, and financial losses.

3. Prevention of Data Loss: NIDS can help protect sensitive data by monitoring network traffic and detecting attempts to exfiltrate data. By identifying unauthorized data transfers or unusual network behavior, NIDS can prevent data breaches and safeguard valuable information.

4. Regulatory Compliance: Many industries have specific regulations and compliance requirements related to network security. Implementing a network intrusion detection system can help organisations meet these requirements and demonstrate their commitment to data protection and security.

5. Real-time Alerts and Incident Response: NIDS generate alerts and notifications when suspicious activities or potential intrusions are detected. This enables

organisations to take immediate action and respond to security incidents promptly. Timely incident response can help mitigate the impact of a security breach and minimise downtime.

6. Network Performance Optimization: Network intrusion detection systems provide valuable insights into network traffic and behaviour. By analysing this data, organizations can identify performance bottlenecks, optimize network configurations, and ensure efficient utilization of network resources.

Overall, network intrusion detection systems play a crucial role in maintaining the integrity and security of networks, protecting sensitive data, and mitigating the risks associated with cyber threats.

1.7 Scope of the project

The scope of a project on network intrusion detection encompasses various aspects related to implementing an effective system for detecting and preventing unauthorized access and malicious activities on a network.
The project aims to develop and deploy a network intrusion detection system (NIDS) that monitors network traffic, analyzes patterns and behaviors, and detects potential security threats in real-time. The scope includes:

1. System Design: The project involves designing the architecture and components of the network intrusion detection system. This includes determining the appropriate sensors, network probes, and monitoring tools to capture and analyze network traffic.

2. Data Collection and Analysis: The NIDS should be capable of collecting network data, such as packet headers, payload content, and session information, for analysis. The scope includes defining the methodologies and algorithms to identify patterns of normal behavior and detect anomalies or suspicious activities.

3. Rule-Based Detection: The project may include defining and implementing a set of rules or signatures to identify known attack patterns. These rules can be based on known vulnerabilities, malware signatures, or specific attack techniques. The scope involves developing and updating these rules to keep up with emerging threats.

4. Machine Learning and Anomaly Detection: To enhance the detection capabilities, the project may involve incorporating machine learning algorithms for anomaly detection. This includes training the system to recognize deviations from normal network behavior and identify previously unseen attack patterns.

5. Incident Response and Remediation: The project may include integrating the NIDS with incident response processes. This involves defining the actions to be taken upon receiving an alert, such as blocking suspicious IP addresses, isolating compromised systems, or initiating forensic analysis for further investigation.

6. Documentation and Training: The project may involve documenting the system design, configuration, and operational procedures. Additionally, it may include providing training and documentation to network administrators and security personnel on effectively managing and maintaining the network intrusion detection system.

By addressing these aspects, the project on network intrusion detection aims to implement a robust and efficient system that can detect and respond to potential security threats, ensuring the integrity and security of the network infrastructure.

1.8 Feasibility Assessment for a Network Intrusion Detection Project:

1. Economic Feasibility:
Cost-Benefit Analysis: Evaluate the costs associated with acquiring and implementing the necessary hardware, software, and personnel for the network intrusion detection system. Compare these costs against the potential benefits such as reduced data breach risks, improved regulatory compliance, and saved financial losses due to incidents. Assess whether the benefits justify the investment.

2. Technical Feasibility:
Technology Assessment: Assess the availability and suitability of the required technologies for the project, including hardware, software, and network infrastructure. Evaluate if the chosen technologies can effectively monitor, analyze, and detect network intrusions. Consider factors such as scalability, compatibility with existing systems, and performance capabilities.

3. Operational Feasibility:
User Acceptance: Assess the willingness and ability of end-users, such as network administrators and security teams, to adopt and utilize the network intrusion detection system effectively. Consider factors such as user-friendliness, training requirements, and potential resistance to change.

Organizational Impact: Evaluate the impact of the project on existing operational processes, workflows, and resources. Consider any potential disruptions, dependencies, or adjustments required for integrating the system into the organization's network infrastructure, security protocols, and incident response procedures.

Maintenance and Support: Assess the organization's capacity to provide ongoing maintenance, monitoring, and support for the network intrusion detection system. Consider the availability of resources and expertise required for system updates, rule/signature updates, and incident response activities.

4. Schedule Feasibility:

Project Planning: Develop a detailed project plan outlining the tasks, milestones, dependencies, and estimated durations for each phase of the project. Assess the feasibility of the proposed timeline, considering potential risks, contingencies, and resource constraints.

Resource Availability: Evaluate the availability and allocation of resources required for the project, including personnel, equipment, and software licenses. Ensure that the necessary resources can be acquired and managed effectively to meet the project's schedule.

Project Management: Assess the organization's project management capabilities and experience in handling similar projects. Determine if the project team has the necessary skills, tools, and processes to effectively manage and control the project's progress, ensuring timely completion.

2 Literature Review

A study by Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, Farhan Ahmad in 2020 explored Network intrusion detection system: A systematic study of machine learning and deep learning approaches. This study first clarifies the concept of IDS and then provides the taxonomy based on the notable ML and DL techniques adopted in designing network-based IDS (NIDS) systems.In this study they found accuracy as high as 95 percent[**1**].An application of machine learning to network intrusion detection by C. Sinclair,L. Pierce,S. Matzner describes the machine learning methodology and the applications employing machine learning to network intrusion detection.they have adapted existing machine learning applications to develop rules for a deployed IDS. The rule generation component of NEDAA is layered onto an expert system that enhances the ability of the IDS to filter anomalous connections.The main result of the presented material is the production of rules for compilation into the expert system[**2**].An article Machine Learning for Reliable Network Attack Detection in SCADA Systems by Rocio Lopez Perez∗, Florian Adamsky†, Ridha Soua†, and Thomas Engel† describes Machine Learning (ML) for intrusion detection in SCADA systems using a real data set collected from a gas pipeline system and provided by the Mississippi State University (MSU). The contribution of this paper is twofold: 1) The evaluation of four techniques for missing data estimation and two techniques for data normalization, 2) The performances of Support Vector Machine (SVM), and Random Forest (RF) are assessed in terms of accuracy, precision, recall and F1 score for intrusion detection. Two cases are differentiated: binary and categorical classifications.They provided a complete comparison between these algorithms along with the random hyper-parameter search results[**3**]. A study Internet of Things Cyber Attacks Detection using Machine Learning by Jadel Alsamiri , Khalid Alsubhi describes various machine learning algorithms that can be used to quickly and effectively detect IoT network attacks. A new dataset, Bot-IoT, is used to evaluate various detection algorithms. In the implementation phase, seven different machine learning algorithms were used, and most of them achieved high performance[**4**].Modern smart grid systems are heavily dependent on Information and Communication Technology, and this dependency makes them prone to cyber-attacks.Sequential Supervised Machine Learning Approach for Cyber Attack Detection in a Smart Grid System by Yasir Ali Farrukh ,Zeeshan Ahmad, Irfan Khan,Rajvikram Madurai Elavarasan describe two-layer hierarchical machine learning model having an accuracy of 95.44 % to improve the detection of cyberattacks. The first layer of the model is used to distinguish between the two modes of operation - normal state or cyberattack. The second layer is used to classify the state into different types of cyberattacks[**5**].With the development of the Internet, cyber-attacks are changing rapidly and the cyber security situation is not optimistic.A study “Machine Learning and Deep Learning Methods for Cybersecurity” by Yang Xin, Lingshuang Kong , Zhi Liu, , Yuling Chen , Yanmiao Li , Hongliang Zhu , Mingcheng Gao , Haixia Hou , Chunhua Wang ,the paper report describes machine learning (ML) and deep learning (DL) methods for network analysis of intrusion detection and provides a brief tutorial description of each ML / DL method. Papers representing each method were indexed, read, and summarized based on their temporal or thermal correlations.The paper, which has mostly focused on the last three years, introduces the latest applications of ML and DL in the field of intrusion detection[**6**].Artificial intelligence (AI), and in particular machine learning(ML), deep learning (DL) has seen huge pace in recent years and is now set to really start influencing all aspects of community and occupations in which people are engaged.”The Future of Cybersecurity: Major Role of Artificial Intelligence, Machine Learning, and Deep Learning in Cyberspace”by B. Geluvaraj, P. M. Satwik and T. A. Ashok Kumar describes the challenges and what is the role of AI, ML, and DL in avoiding cybercrime in future[**7**].

Methodology-

3.1 Dataset Collection

The UNSW15 dataset, also known as the UNSW-NB15 dataset, is a network intrusion detection dataset that is commonly used for evaluating intrusion detection systems. The dataset contains network traffic data captured from a real-world network environment.

The UNSW15 dataset has a total of 49 features, including both numerical and categorical features. These features provide information about various aspects of network traffic, such as source and destination IP addresses, port numbers, packet sizes, protocol types, and others.

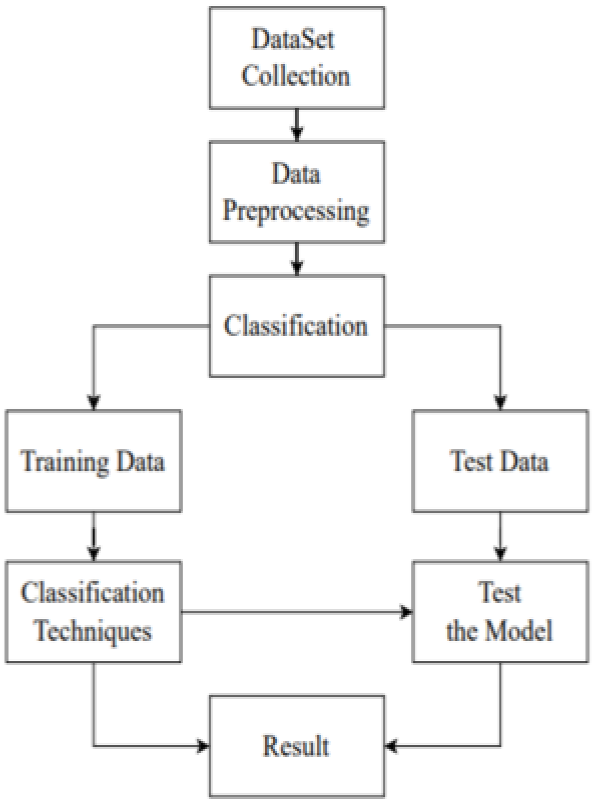
It's worth noting that the dataset may also include additional columns for the target variable, which indicates whether a network flow is classified as normal or an attack. The presence of the target variable depends on how the dataset is prepared for a specific task, such as intrusion detection or classification.

Overall, the UNSW15 dataset consists of 49 features that provide information about network traffic characteristics and can be used for analyzing and developing intrusion detection systems.

3.1.1 Data Processing

For betterment we generated the correlation matrix and we tried to found out the features with very low correlation with other features, near by 0. We finished with finding 15 features and we drop those features from the dataset. Then, using labelEncoder we convert all the categorical attributes to the numerical values so that the dataset was ready to use for algorithm. We keep the **“attack_cat”** attribute as our target attribute.

3.2 System Architecture for Predicting Category of Network Attack by using Multiple ML Algorithms



3.2.1 Naive Bayes Classifier:

Notably the Naive Bayes Classifier, take the traits given in Table 3.1 as input. The Naive Bayes algorithm is the most popular and simple machine learning classification approach, as described in [3]. It is a set of Bayes Theorem-based classification methods. We cannot refer to it as a single algorithm since it is a family of algorithms that all operate in accordance with the same principle. The Bayes theorem may be used to calculate the posterior probability P(c|x) from P(c), P(x), and P(x|c) using the following equation:

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

The posterior probability of the provided predictor (x, attributes) to the given class (c, target) is P(c|x) in this case.

P(c) stands for the prior probability of the class.

• The predictor's prior probability is denoted by P(x).

• The likelihood, or P(x|c), measures how likely a certain class of predictors is.

The Gaussian Naive Bayes Classifier, one of three varieties of Naive Bayes models, was employed. Classification algorithms like the Gaussian Naive Bayes make the assumption that features have a normal distribution.

3.2.2 Support vector classifier:

The support vector machine (SVM) is a further straightforward method that any machine learning expert should be familiar with. Because the support vector machine achieves excellent accuracy while consuming less processing power, many people like it. SVM is a sort of method that may be applied to both classification and regression. It is, nonetheless, heavily utilized in classification aims. Each data point in the SVM algorithm is represented as a point in n-dimensional space (where n is the number of features), with each feature's value being the value of a specific coordinate in the SVM algorithm. By finding the

hyper-plane that clearly separates the two classes, the classification is then completed.

3.2.3 Classifier KNeighbors:

One of the clustering methods used to group datasets according to nearest-neighbor is K-means clustering. Here, the data are grouped into k clusters depending on how similar they are to one another. Using k-means [20], we can additionally fill in the missing values in the data. The supervised machine learning technique known as the k-nearest neighbors (KNN) can be used to tackle classification and regression issues. Although it is simple to use and comprehend, a big disadvantage is that it becomes noticeably 13 slower as the amount of data in use increases. KNN works by calculating the distances between a query and each example in the data, choosing the K instances closest to the query, and then voting for the label with the highest frequency (in the case of regression).

3.2.4 Random Forest Algorithm:

The Random Forest algorithm is an ensemble learning method that combines multiple decision trees to make predictions. Each decision tree in the random forest is trained on a random subset of the training data and uses a random subset of the features. The final prediction of the random forest is determined by aggregating the predictions of all the individual trees, either through majority voting (for classification problems) or averaging (for regression problems).

The equation for the Random Forest classifier can be described as follows:

For a given input instance x, let F be the set of decision trees in the random forest, and let f_i(x) be the prediction of the i-th decision tree. Each decision tree provides a binary output (0 or 1) indicating the predicted class label.

The Random Forest classifier aggregates the predictions of all the decision trees by majority voting. The final prediction y_rf(x) for the input instance x is determined by:

y_rf(x) = argmax(sum(f_i(x))),

where argmax is the function that returns the class label with the highest sum of votes.

In summary, the Random Forest classifier combines the predictions of multiple decision trees by majority voting to make the final prediction for a given input instance.

3.4.5 Logistic Regression Algorithm:

Logistic Regression is a popular statistical algorithm used for binary classification problems. It models the relationship between a set of input features and a binary target variable using the logistic function, which maps the input to a probability value between 0 and 1.

Model Evaluation: After the model is trained, it can be evaluated on a separate test set to assess its performance using various metrics such as accuracy, precision, recall, and F1-sco

Logistic Regression can also be extended to handle multi-class classification problems using techniques like one-vs-rest or softmax regression.

4 Proposed System

In the proposed system the UNSW15 dataset is used to generate the training and testing samples .Then after back propagation parameters determine the algorithm used to train network to be used later in intrusion detection .The proposed system consisted of following steps:

1. Selecting subsets of samples for training and test phases from UNSW15 dataset.

2. Preprocessing of the selected subsets of samples

3. Training back propagation neural network algorithm using the samples of training set.

4. Test the BPNN using the testing samples.

5. Calculating the performance of trained module.

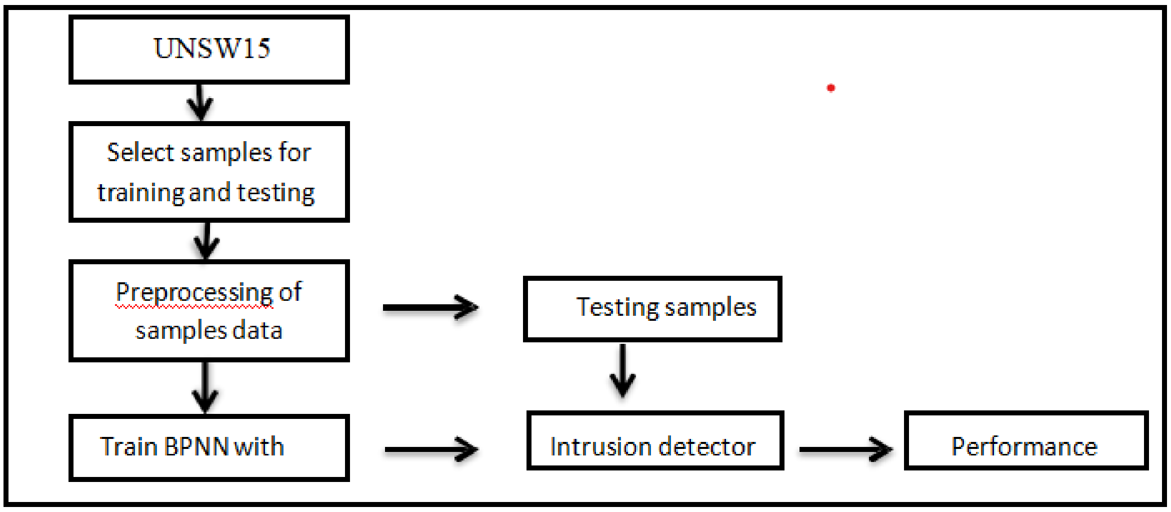

UNSW15
Select samples for training and testing
Preprocessing of samples data
Testing samples
Train BPNN with
Intrusion detector
Performance

Fig 4.1 - Block diagram of the proposed system

4.2 Functional Requirements

Functional Requirements The module needs to perform a variety of functions. We have grouped these functions into several categories covering collecting, processing, analyzing, reporting, and storing intrusion and vulnerability data, providing alerts, displaying information, controlling IDS resources, reacting to intrusions and vulnerabilities, and interacting with other modules. These functions provide lower level requirements than the processing requirements of the previous section. Collecting The NMA module collects intrusion detection and vulnerability data. For intrusion detection in both real time and non-real time, the NMA module needs to

• Collect suspicious traffic and ancillary information that describes or characterizes the traffic, identifying distinct network connections or associations (connectionless traffic) and including enough detail to assist criminal investigations and prosecutions

• Detect intrusions specific to a designated area of protection

• Detect denial of service attacks to include

• Automatically record events and incidents
• Monitor and scan networks
• Monitor and scan hosts
• Detect based on content; for example, a packet body
• Detect based on context
• Detect intrusions for multiple operating systems

Non Functional Requirements

Non-functional requirements are the requirements that do not directly show the specific functions of the system. They may specify system performance and maintainability and security.

System performance:

The user interface should be smooth and there should not be any crashes in the system.

Usability:

The system should be compatible with any PCs. It should work under any environment and also under any conditions.

Maintainability:

Preparing the software is not just the final. Maintenance is also an important thing. The maintenance cost should be less. Services should be available all the time without any interruptions.

Efficiency:

The output should be more accurate and should have a low false positive rate.

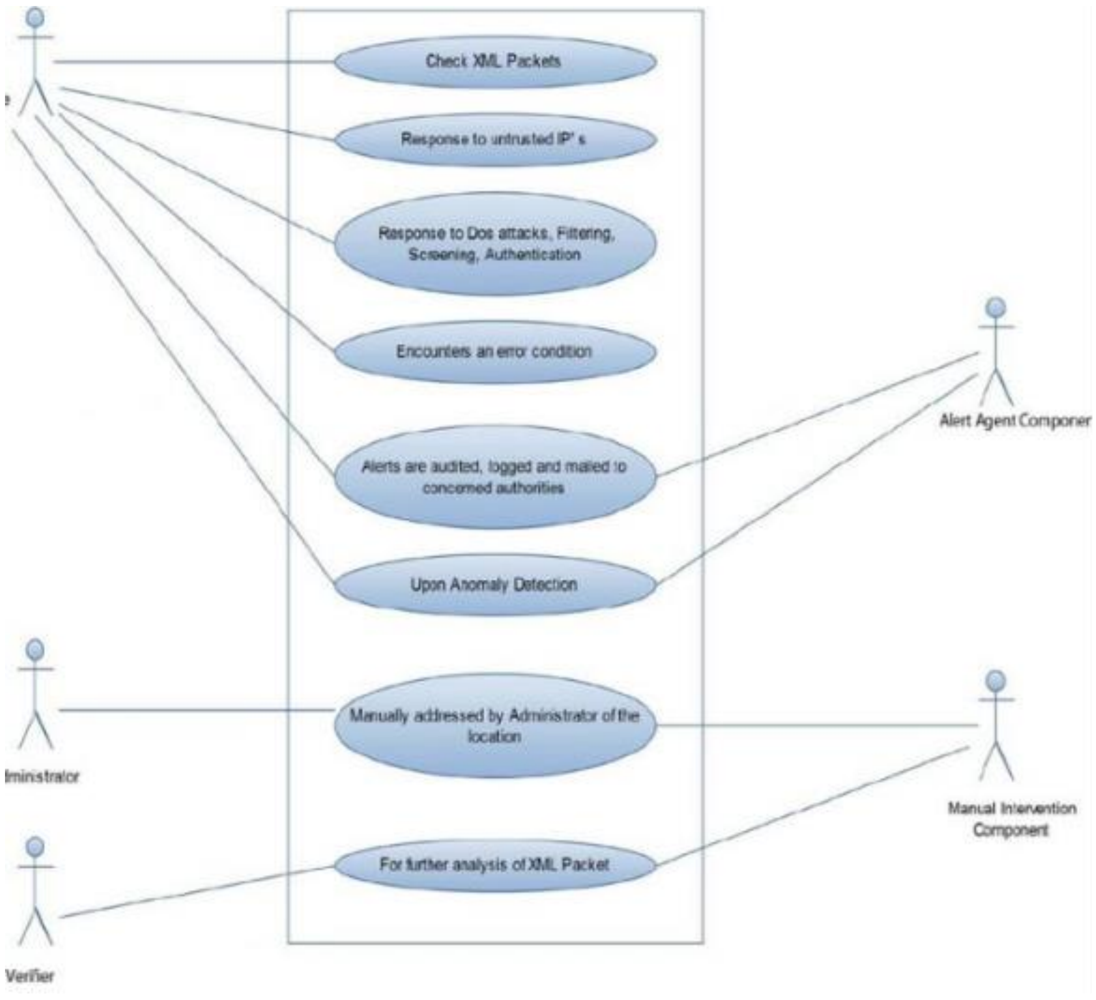

Check XML Packets
Response to untrusted IP' s
Response to Dos attacks, Filtering, Screening, Authentication
Encounters an error condition
Alerts are audited, logged and mailed to concerned authorities
Upon Anomaly Detection
Manually addressed by Administrator of the location
For further analysis of XML Packet
Alert Agent Componer
lministrator
Manual Intervention Component
Verifier

Fig- Use Case Diagram

Use Case Description: Network Attack Category Detection using Machine Learning

Introduction:

Network security is a critical aspect of any organization's infrastructure. Detecting and mitigating network attacks is crucial to ensure the confidentiality, integrity, and availability of data and resources. Machine learning can play a significant role in automating the detection of network attacks. In this use case, we aim to develop a simple machine learning project for network attack category detection.Problem Statement:

The goal of this project is to build a machine learning model that can classify network traffic into different attack categories. The model will analyze network packets or flow data and predict the type of attack present in the traffic

Dataset:

To train the machine learning model, a labeled dataset of network traffic is required. This dataset should include network packets or flow data along with their corresponding attack labels. The dataset can be obtained from public repositories, such as the KDD Cup 1999 dataset or the NSL-KDD dataset, which contain labeled network traffic for various attack types.

Data Preprocessing:

Before training the machine learning model, the dataset needs to be preprocessed. This step may involve removing irrelevant features, normalizing numerical values, handling missing data, and converting categorical variables into numerical representations.

Feature Extraction:

To capture the relevant information from the network traffic data, feature extraction techniques can be employed. Commonly used features include packet header information, protocol-specific attributes, and statistical features derived from the traffic data.

Model Training:

Once the dataset is preprocessed and features are extracted, the next step is to train a machine learning model. Various supervised learning algorithms can be considered, such as decision trees, random forests, support vector machines (SVM), or neural networks. The model will be trained on the labeled dataset, using a portion of the data for training and the rest for evaluation.

Model Evaluation:

To assess the performance of the trained model, evaluation metrics such as accuracy, precision, recall, and F1 score can be computed. Additionally, techniques like cross-validation can be applied to ensure the model's generalization capability.

Deployment:

Once the model is trained and evaluated, it can be deployed in a real-time network monitoring system. The deployed model will analyze incoming network traffic and predict the attack category associated with each network flow or packet.

Continuous Improvement:

To ensure the model's effectiveness in detecting new and evolving network attacks, continuous improvement and monitoring are crucial. The model can be periodically retrained with new labeled data to adapt to changing attack patterns.

Class Diagram-

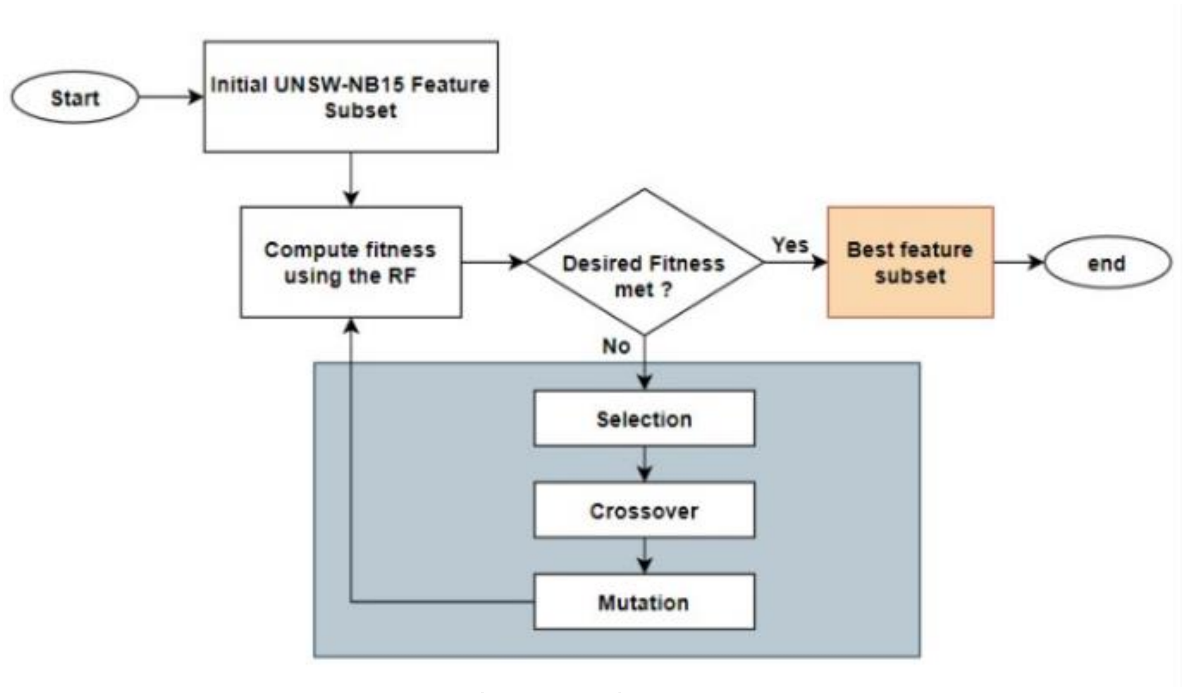


Fig- Class Diagram

Sequence Diagram-

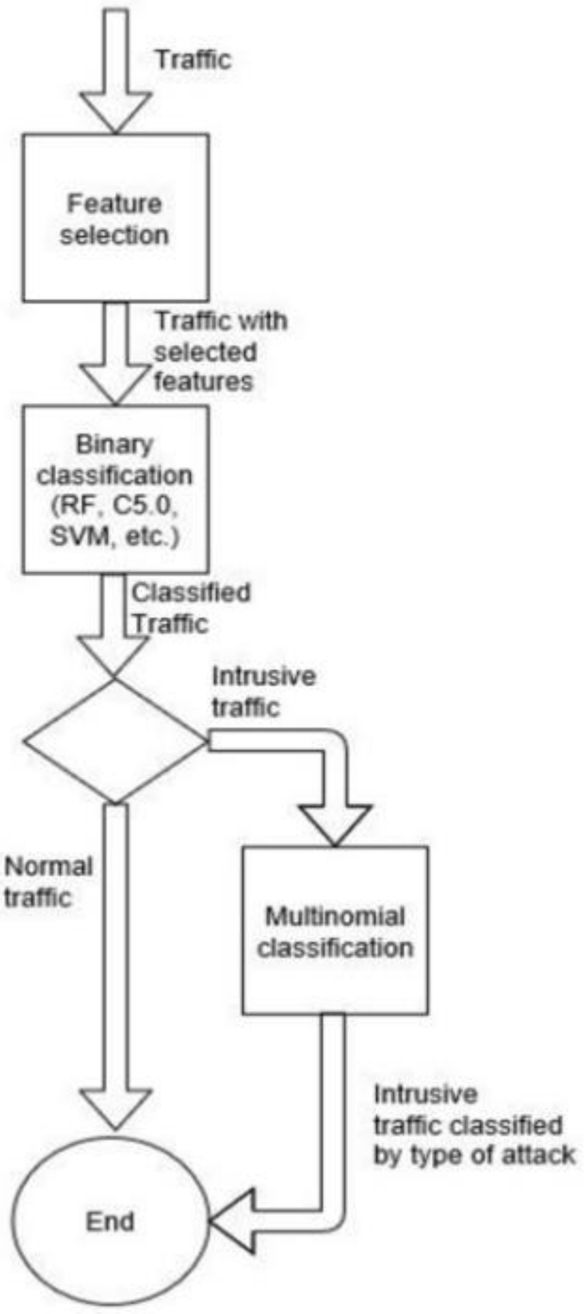

Traffic
Feature selection
Traffic with selected features
Binary classification (RF, C5.0, SVM, etc.)
Classified Traffic
Intrusive traffic
Normal traffic
Multinomial classification
Intrusive traffic classified by type of attack
End

Fig - Sequence Diagram

5 Design Contents

Basically it involves three major phases: (i) input, which captures the network traffic and transmits it to the next phase; (ii) analysis, which is the most significant phase of the system where actual computation is performed and it consists of several components; and (iii) output, where alerts are being generated to identify malicious activities. The following subsections explain each phase in detail.

4.2 System Architecture

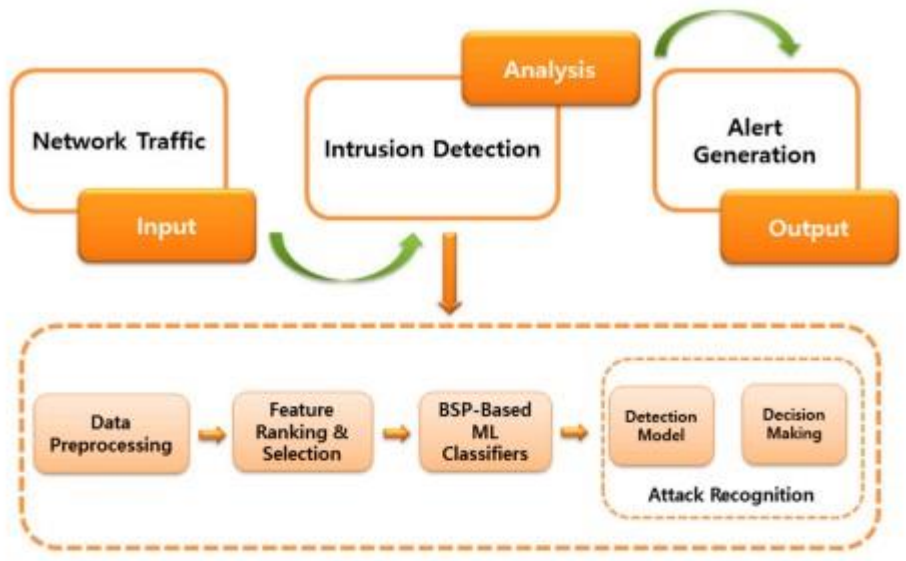


Fig 4.2.1 -Proposed System Architecture

The architecture of the proposed intrusion detection system. Input It is the basic but most significant part of any system that greatly affects its performance and operations. In the case of IDSs, these are network flows in real-time environments or may be recorded network traces often called workloads or datasets. The type, quality and the location where data is collected from are the determinate factors in the design and effectiveness of an IDS. We believe that the productivity of NIDS research is largely dependent on the quality of datasets being used in addition to computational techniques involved. Based on these principles, we decided to use the UNSW15 dataset as input to our proposed system.

4.3 Subsystem Decomposition

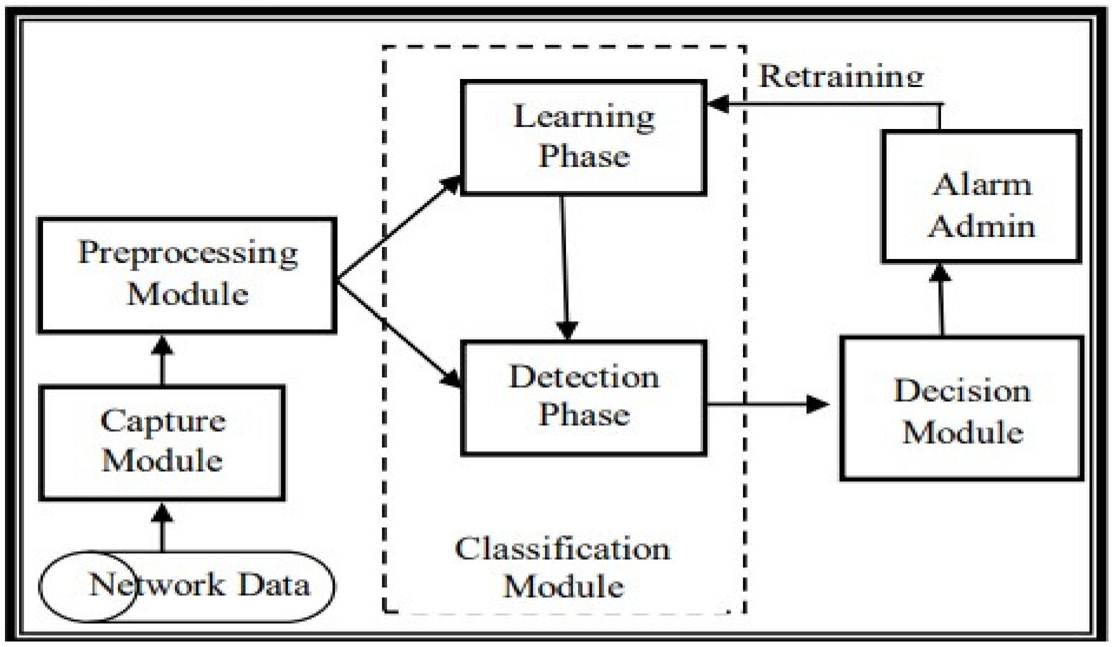


Fig 4.3.1 -Components of System

4.3.1 Module list

A. The Capture Module

Raw data of the network are captured and stored using the network adapter. It utilises the capabilities of the TCP dump capture utility for Windows to gather historical network packets.

B. The Preprocessing Module

The data must be of uniform representation to be processed by the classification module. The preprocessing module is responsible for reading, processing, and filtering the audit data to be used by the classification module. The preprocessing module handles Numerical Representation, Normalization and Features selection of raw input data. The preprocessing module consists of three phases:

1) Numerical Representation

2) Normalization

3)Dimension Reduction

C. The classification Module

The classification module has two phases of operation. The learning and the detection phase.

1) The Learning Phase-

In the learning phase, the classifier uses the preprocessed captured network user profiles as input training patterns. This phase continues until a satisfactory correct classification rate is obtained.

2) The Detection Phase-

Once the classifier is learned, its capability of generalization to correctly identify the different types of users should be utilized to detect intruder. This detection process can be viewed as a classification of input patterns to either normal or attack.

D. The Decision Module

The basic responsibility of the decision module is to transmit an alert to the system administrator informing him of coming attack. This gives the system administrator the ability to monitor the progress of the detection module. To evaluate our system we used two major indices of performance. We calculate the detection rate and the false alarm rate according to the following assumptions :

False Positive (FP): the total number of normal records that are classified as anomalous

False Negative (FN): the total number of anomalous records that are classified as normal

Total Normal (TN): the total number of normal records

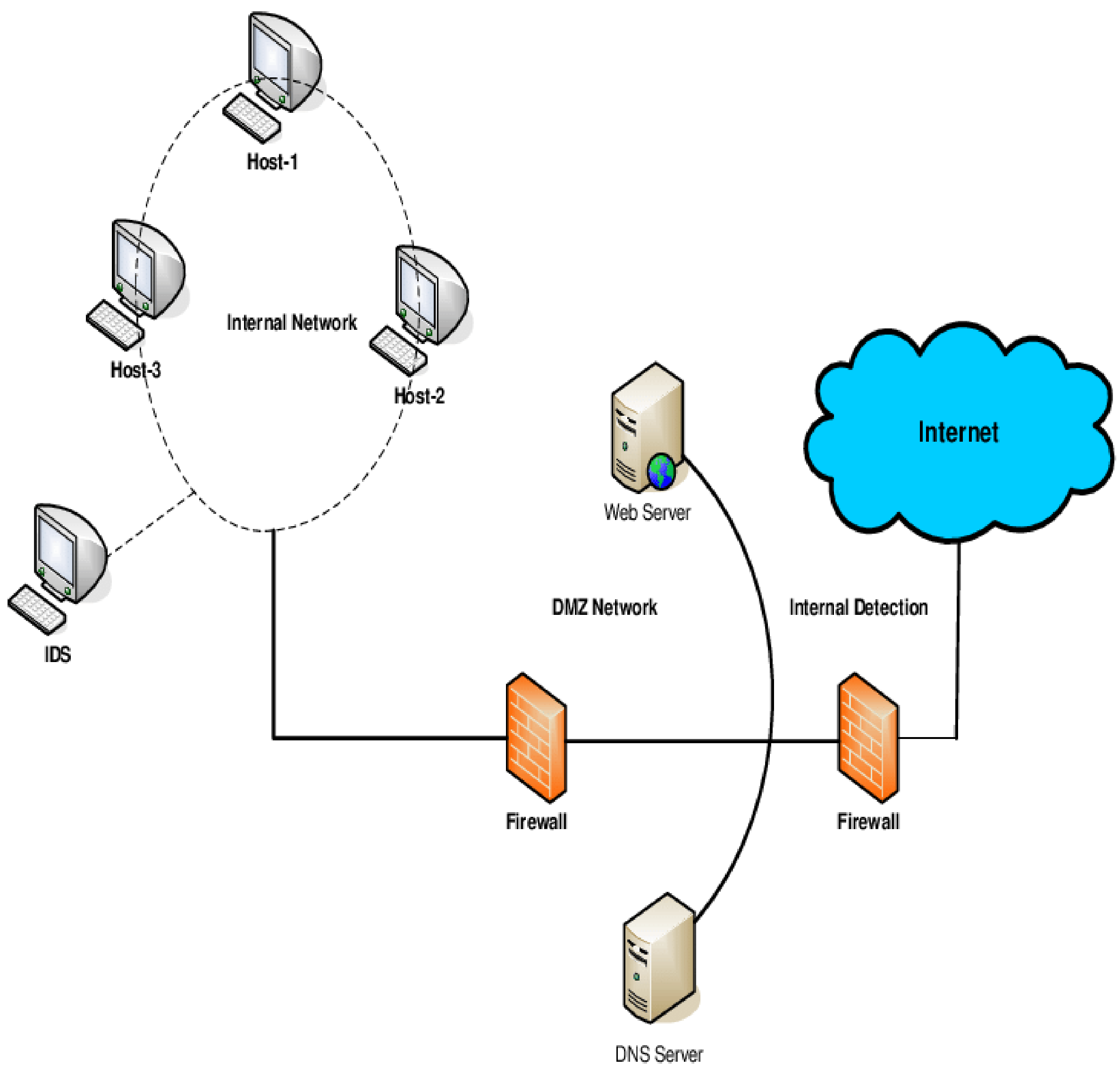
Total Attack (TA): the total number of attack records

Detection Rate = [(TA-FN) / TA]*100

False Alarm Rate = [FP/TN]*100

Correct Classification Rate = Number of Records Correctly Classified / Total Number of records in the used dataset

5.4 System Layout



Intrusion Detection System (IDS) are classified according to the audit data to tow main parts: Host-based or Network based. The hostbased IDS uses application logs in the analysis. Whereas the network-based operates by capturing and evaluating (investigate) the network packet received from network traffic. On the other hand, IDS schemes are categorised

5.5 User interface design

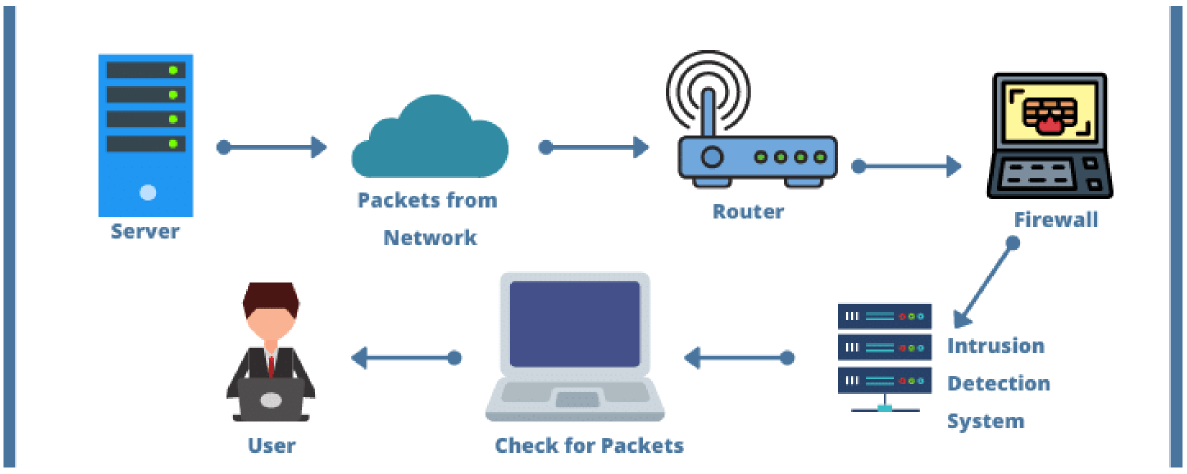


Fig 5.5.1 User Interface Design

Chapter Six - Implementation

6.1 Introduction

The Network Attack Detection App is an implementation designed to detect network attacks using the UNSW15 dataset. The dataset consists of network traffic data captured in a controlled environment, containing both normal and malicious activities. The app aims to analyze the network traffic and classify it as either normal or indicative of an attack.

The app focuses on a single input page and a single output page, which makes it suitable for analyzing specific network traffic patterns. By utilizing machine learning techniques and feature extraction methods, the app will train a model on the UNSW15 dataset to accurately identify network attacks in real-time.

Data Preprocessing: The first step in the implementation involves preprocessing the UNSW15 dataset. This includes cleaning the data, handling missing values, and performing feature engineering to extract relevant information from the network traffic.

Feature Extraction: Feature extraction plays a crucial role in identifying patterns and characteristics of network attacks. The app will extract relevant features such as packet size, protocol type, source and destination IP addresses, and more, to create a comprehensive representation of the network traffic.

Model Training: Once the features are extracted, a machine learning model will be trained using a suitable algorithm such as Random Forest, Support Vector Machine (SVM), or Neural Networks. The training process involves splitting the dataset into training and validation sets, training the model on the training set, and evaluating its performance on the validation set.

Real-Time Classification: After the model is trained, it can be deployed to classify real-time network traffic. The app will continuously monitor incoming network traffic, extract the necessary features, and feed them into the trained model. Based on the model's predictions, the app will determine whether the traffic is normal or indicative of a network attack.

Alert Generation: In case a network attack is detected, the app will generate an alert to notify the appropriate personnel or systems responsible for network security. The alert can be in the form of a system notification, email, or any other suitable means.

6.2 Algorithm Development

After applying 9 ML algorithm's we reached to the conclusion that Random Forest Classifier Algorithm Performed best of our project. It has given 91.65% accuracy in classification of attack on train dataset and 83.03% accuracy on test dataset. Here is the description, how we reached to that conclusion-

Data Preprocessing:

Load the UNSW15 dataset. Perform data cleaning, handling missing values, and removing irrelevant columns. Encode categorical features into numerical representations if necessary. Split the dataset into input features (X) and target labels (y).

Feature Extraction:

Select relevant features from the input page that can be indicative of network attacks. Extract features such as packet size, protocol type, source and destination IP addresses, and any other relevant information. Perform feature scaling or normalization if required.

Model Training:

Split the preprocessed dataset into training and validation sets. Select a suitable machine learning algorithm such as Random Forest, Support Vector Machine (SVM), or Neural Networks. Train the selected model using the training set and tune its hyperparameters if necessary. Evaluate the model's performance using appropriate metrics on the validation set.

Real-Time Classification:

Set up a real-time network traffic monitoring system. Continuously capture network traffic data from the input page. Preprocess the incoming traffic data using the same preprocessing steps as performed during training. Extract the relevant features from the incoming traffic. Feed the features into the trained model for classification. Obtain the predicted label (normal or attack) from the model.

Alert Generation:

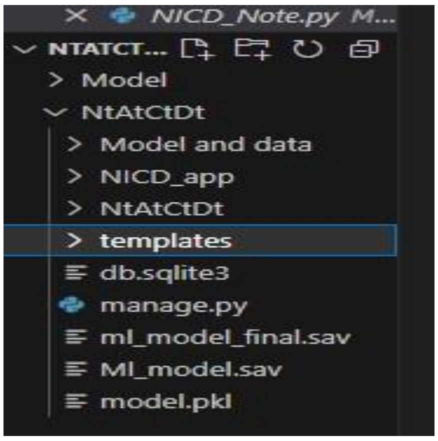
If the predicted label indicates an attack, generate an alert. Determine the appropriate method of alert generation, such as system notifications, emails, or logging to a central monitoring system. Include relevant information about the detected attack, such as timestamp, source/destination IP addresses, and the type of attack. Trigger the alert to notify the appropriate personnel or systems responsible for network security.

Continual Monitoring and Model Updates:

Continuously monitor the network traffic and update the model as needed Periodically retrain the model using newly collected labeled data to adapt to changing attack patterns. Perform regular evaluations to ensure the model's effectiveness and make necessary adjustments if required.

6.3 Coding (coding as annex):

Here is the coding below and explanation, We do this project using Django to make a small website which has a input taking page and output giving page.

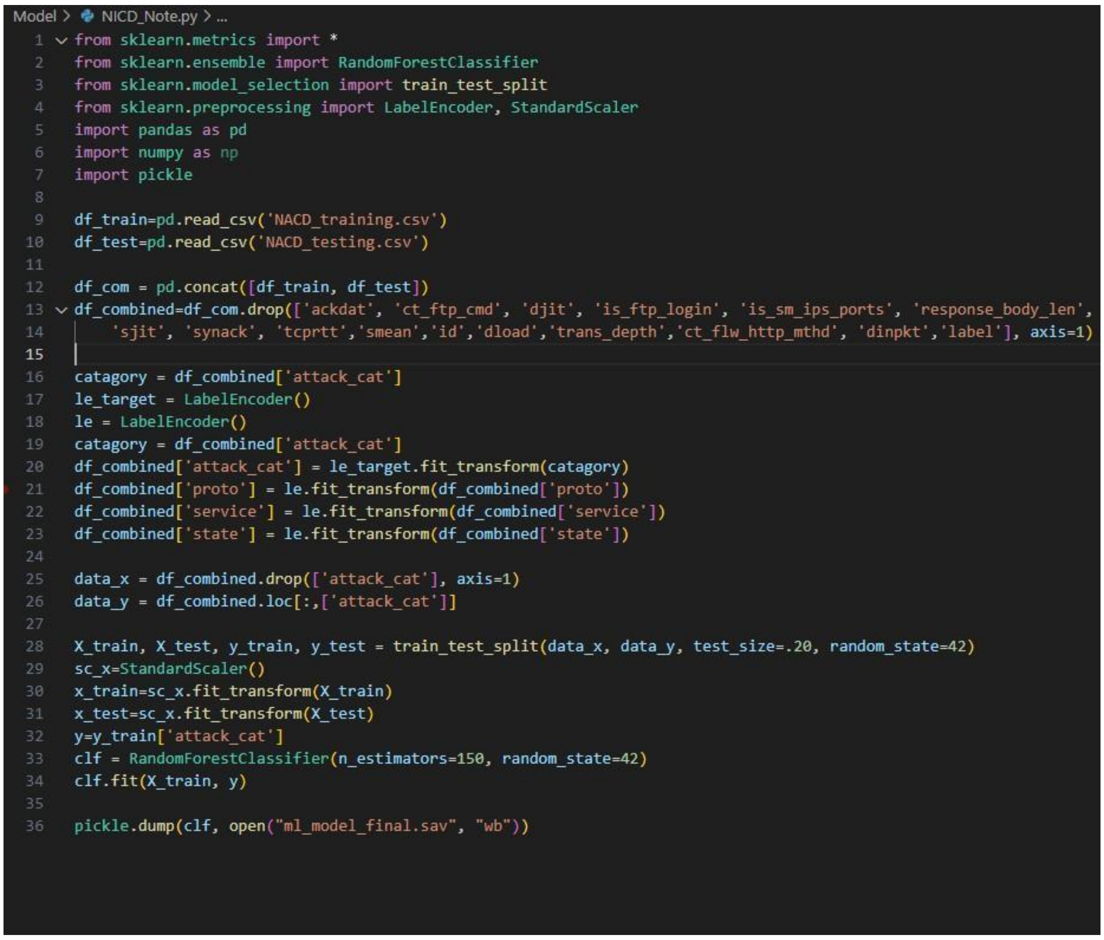


```
NICD_Note.py M...
NTATCT...
  > Model
  v NtAtCtDt
    > Model and data
    > NICD_app
    > NtAtCtDt
    > templates
    db.sqlite3
    manage.py
    ml_model_final.sav
    Ml_model.sav
    model.pkl
```

Here, is the folder structure which follows The MVC framework method.

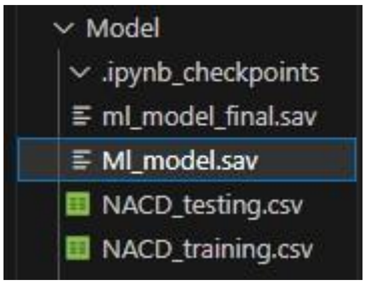
Here, we import the ML algorithm Random Forest Classifier in the project using pickle library.

Firstly, We have made a ml_model_final.SAV file which helps us to import the algorithm in the project directly.

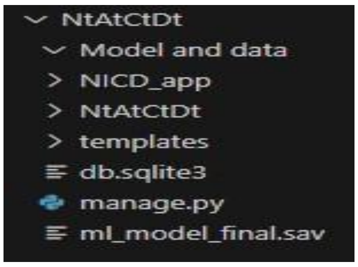

Model > NICD_Note.py > ...

```python
from sklearn.metrics import *
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import pandas as pd
import numpy as np
import pickle

df_train=pd.read_csv('NACD_training.csv')
df_test=pd.read_csv('NACD_testing.csv')

df_com = pd.concat([df_train, df_test])
df_combined=df_com.drop(['ackdat', 'ct_ftp_cmd', 'djit', 'is_ftp_login', 'is_sm_ips_ports', 'response_body_len',
    'sjit', 'synack', 'tcprtt','smean','id','dload','trans_depth','ct_flw_http_mthd', 'dinpkt','label'], axis=1)

catagory = df_combined['attack_cat']
le_target = LabelEncoder()
le = LabelEncoder()
catagory = df_combined['attack_cat']
df_combined['attack_cat'] = le_target.fit_transform(catagory)
df_combined['proto'] = le.fit_transform(df_combined['proto'])
df_combined['service'] = le.fit_transform(df_combined['service'])
df_combined['state'] = le.fit_transform(df_combined['state'])

data_x = df_combined.drop(['attack_cat'], axis=1)
data_y = df_combined.loc[:,['attack_cat']]

X_train, X_test, y_train, y_test = train_test_split(data_x, data_y, test_size=.20, random_state=42)
sc_x=StandardScaler()
x_train=sc_x.fit_transform(X_train)
x_test=sc_x.fit_transform(X_test)
y=y_train['attack_cat']
clf = RandomForestClassifier(n_estimators=150, random_state=42)
clf.fit(X_train, y)

pickle.dump(clf, open("ml_model_final.sav", "wb"))
```

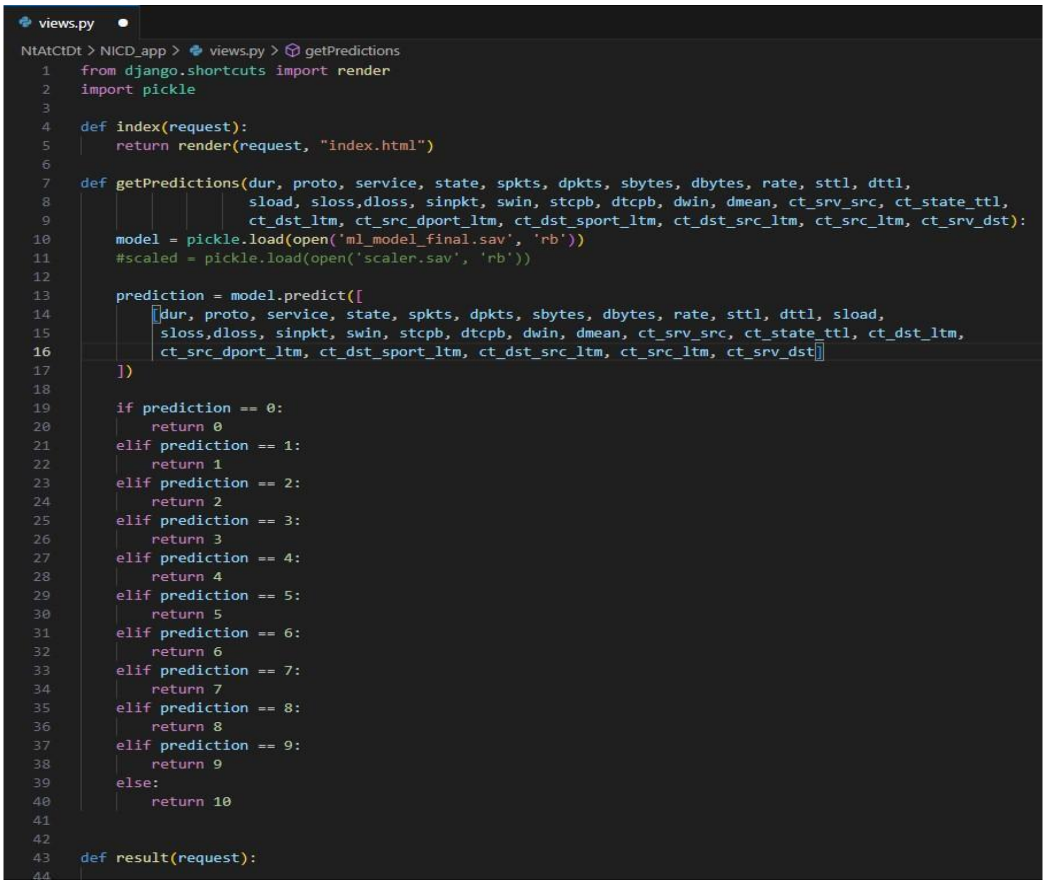
Here, is the just made ml_model_final.SAV file


```
Model
  .ipynb_checkpoints
  ml_model_final.sav
  Ml_model.sav
  NACD_testing.csv
  NACD_training.csv
```

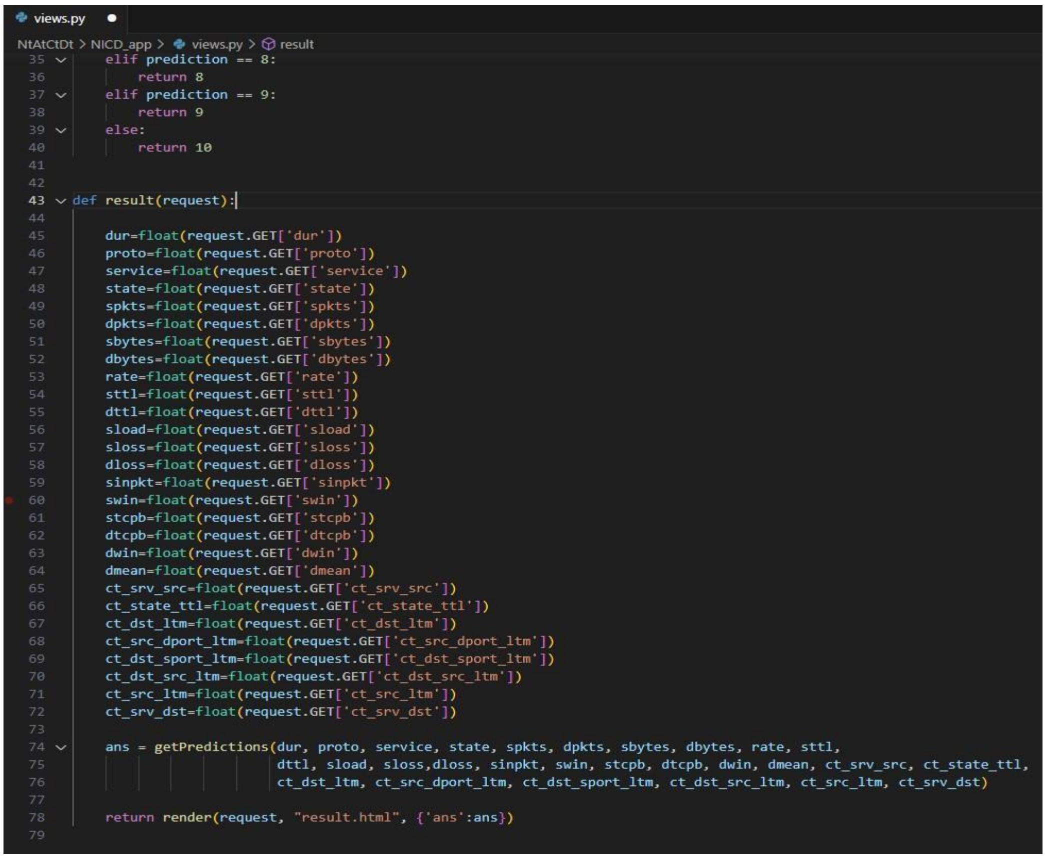
Then, we transfer the ml_model_final.SAV file to the actual project.


```
NtAtCtDt
  Model and data
  NICD_app
  NtAtCtDt
  templates
  db.sqlite3
  manage.py
  ml_model_final.sav
```

Here, is the view controller. Here, we made three functions. Among them index() we help the index.html page, which is our input page and the other two getpredictions() and The Rresult() function will help the output page which is result.html to return the result or, final output .

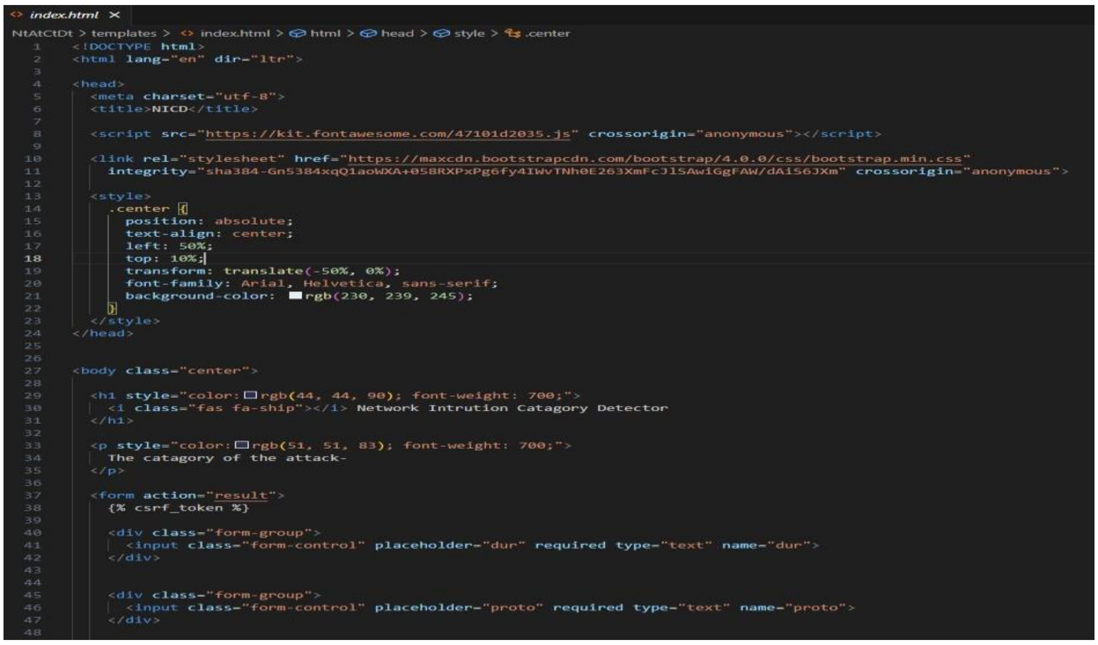


```python
from django.shortcuts import render
import pickle

def index(request):
    return render(request, "index.html")

def getPredictions(dur, proto, service, state, spkts, dpkts, sbytes, dbytes, rate, sttl, dttl,
                   sload, sloss,dloss, sinpkt, swin, stcpb, dtcpb, dwin, dmean, ct_srv_src, ct_state_ttl,
                   ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst):
    model = pickle.load(open('ml_model_final.sav', 'rb'))
    #scaled = pickle.load(open('scaler.sav', 'rb'))

    prediction = model.predict([
        [dur, proto, service, state, spkts, dpkts, sbytes, dbytes, rate, sttl, dttl, sload,
         sloss,dloss, sinpkt, swin, stcpb, dtcpb, dwin, dmean, ct_srv_src, ct_state_ttl, ct_dst_ltm,
         ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst]
    ])

    if prediction == 0:
        return 0
    elif prediction == 1:
        return 1
    elif prediction == 2:
        return 2
    elif prediction == 3:
        return 3
    elif prediction == 4:
        return 4
    elif prediction == 5:
        return 5
    elif prediction == 6:
        return 6
    elif prediction == 7:
        return 7
    elif prediction == 8:
        return 8
    elif prediction == 9:
        return 9
    else:
        return 10


def result(request):
```



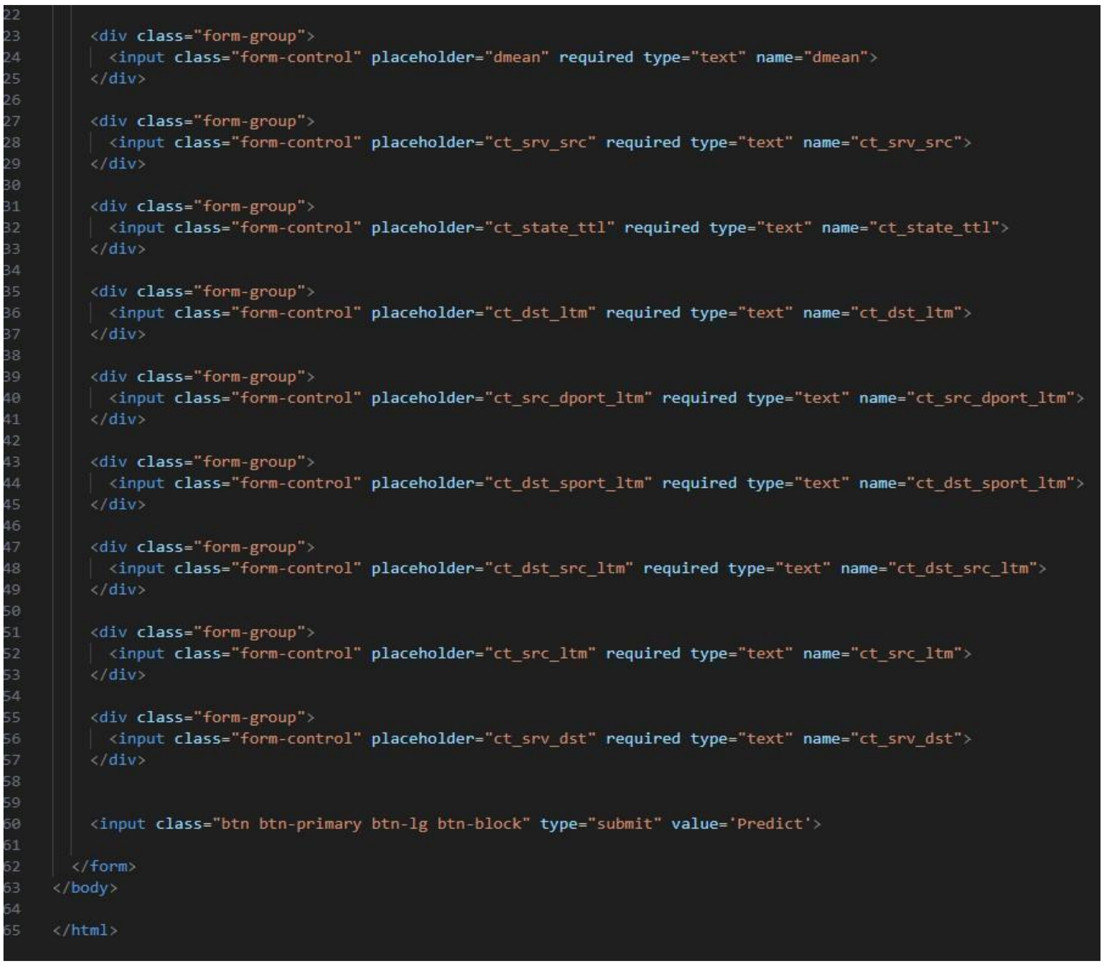
```python
    elif prediction == 8:
        return 8
    elif prediction == 9:
        return 9
    else:
        return 10


def result(request):

    dur=float(request.GET['dur'])
    proto=float(request.GET['proto'])
    service=float(request.GET['service'])
    state=float(request.GET['state'])
    spkts=float(request.GET['spkts'])
    dpkts=float(request.GET['dpkts'])
    sbytes=float(request.GET['sbytes'])
    dbytes=float(request.GET['dbytes'])
    rate=float(request.GET['rate'])
    sttl=float(request.GET['sttl'])
    dttl=float(request.GET['dttl'])
    sload=float(request.GET['sload'])
    sloss=float(request.GET['sloss'])
    dloss=float(request.GET['dloss'])
    sinpkt=float(request.GET['sinpkt'])
    swin=float(request.GET['swin'])
    stcpb=float(request.GET['stcpb'])
    dtcpb=float(request.GET['dtcpb'])
    dwin=float(request.GET['dwin'])
    dmean=float(request.GET['dmean'])
    ct_srv_src=float(request.GET['ct_srv_src'])
    ct_state_ttl=float(request.GET['ct_state_ttl'])
    ct_dst_ltm=float(request.GET['ct_dst_ltm'])
    ct_src_dport_ltm=float(request.GET['ct_src_dport_ltm'])
    ct_dst_sport_ltm=float(request.GET['ct_dst_sport_ltm'])
    ct_dst_src_ltm=float(request.GET['ct_dst_src_ltm'])
    ct_src_ltm=float(request.GET['ct_src_ltm'])
    ct_srv_dst=float(request.GET['ct_srv_dst'])

    ans = getPredictions(dur, proto, service, state, spkts, dpkts, sbytes, dbytes, rate, sttl,
                         dttl, sload, sloss,dloss, sinpkt, swin, stcpb, dtcpb, dwin, dmean, ct_srv_src, ct_state_ttl,
                         ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst)

    return render(request, "result.html", {'ans':ans})
```

This is the index.html page means our input page’s code. Here 28 input or feature value will be taken.



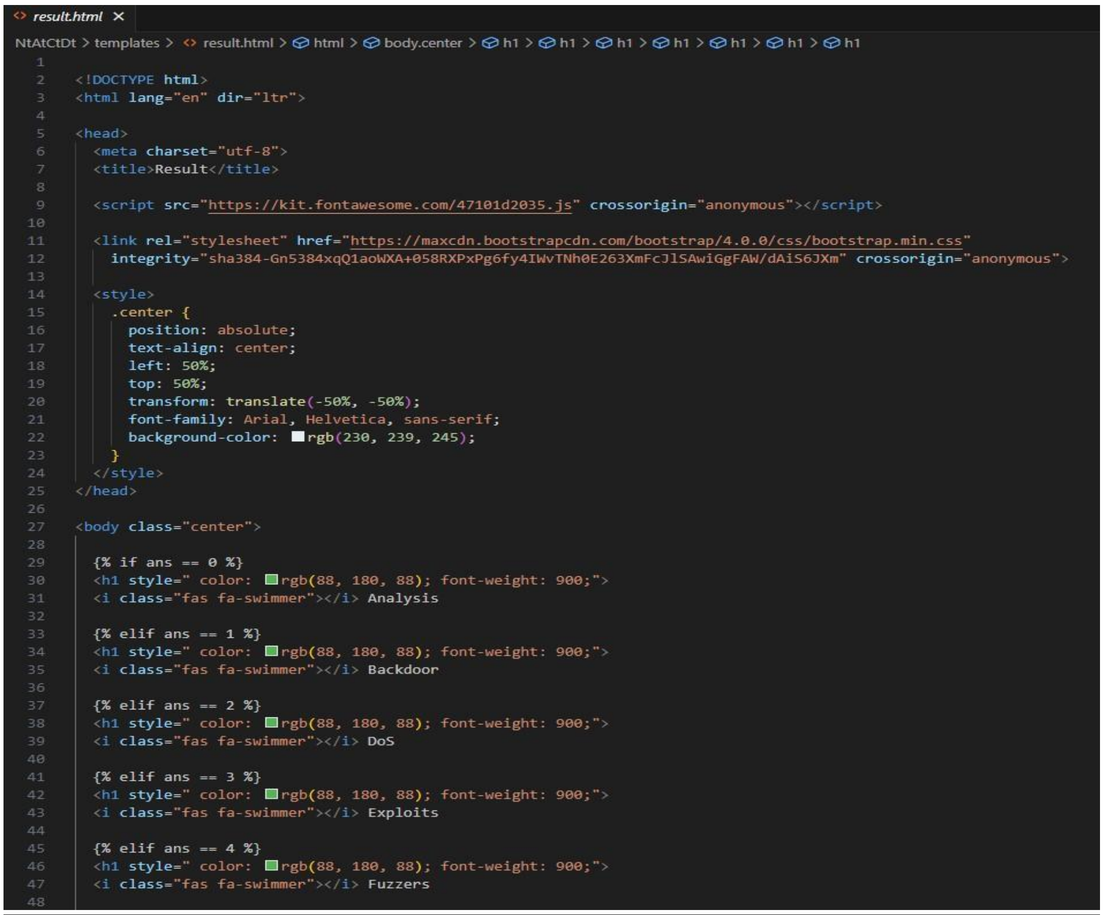
index.html
NtAtCtDt > templates > index.html > html > head > style > .center

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
  <meta charset="utf-8">
  <title>NICD</title>

  <script src="https://kit.fontawesome.com/47101d2035.js" crossorigin="anonymous"></script>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

  <style>
    .center {
      position: absolute;
      text-align: center;
      left: 50%;
      top: 10%;
      transform: translate(-50%, 0%);
      font-family: Arial, Helvetica, sans-serif;
      background-color: rgb(230, 239, 245);
    }
  </style>
</head>


<body class="center">

  <h1 style="color: rgb(44, 44, 90); font-weight: 700;">
    <i class="fas fa-ship"></i> Network Intrution Catagory Detector
  </h1>

  <p style="color: rgb(51, 51, 83); font-weight: 700;">
    The catagory of the attack-
  </p>

  <form action="result">
    {% csrf_token %}

    <div class="form-group">
      <input class="form-control" placeholder="dur" required type="text" name="dur">
    </div>


    <div class="form-group">
      <input class="form-control" placeholder="proto" required type="text" name="proto">
    </div>

```

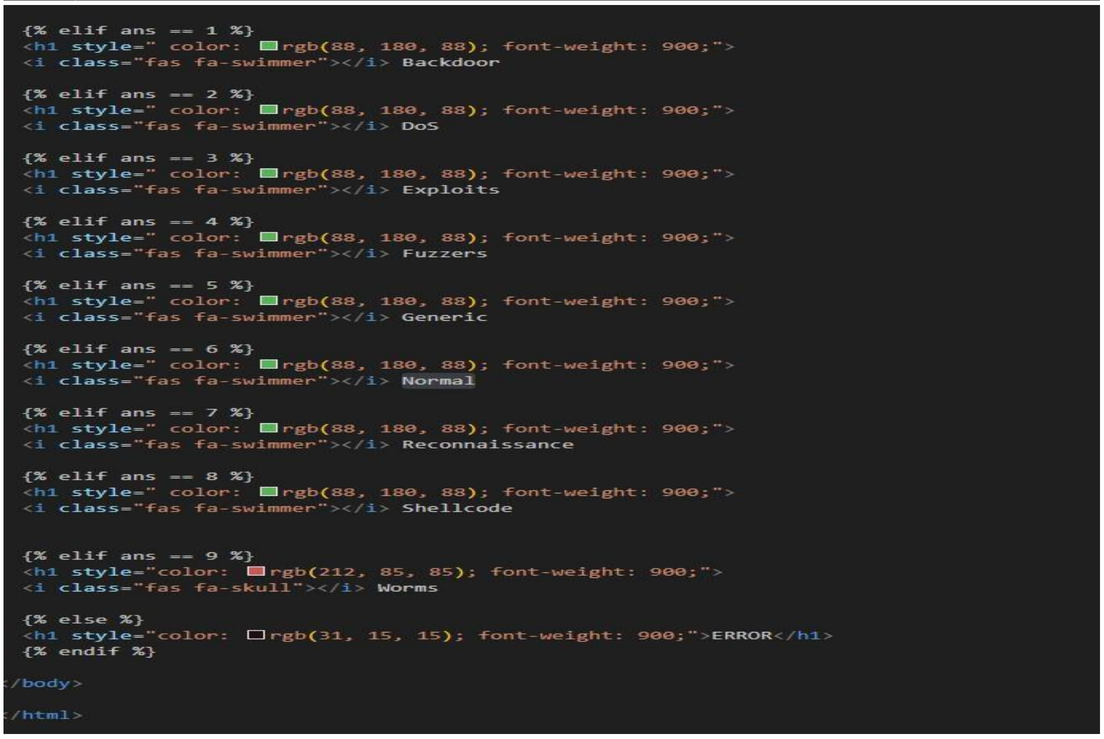


```

    <div class="form-group">
      <input class="form-control" placeholder="dmean" required type="text" name="dmean">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_srv_src" required type="text" name="ct_srv_src">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_state_ttl" required type="text" name="ct_state_ttl">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_dst_ltm" required type="text" name="ct_dst_ltm">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_src_dport_ltm" required type="text" name="ct_src_dport_ltm">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_dst_sport_ltm" required type="text" name="ct_dst_sport_ltm">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_dst_src_ltm" required type="text" name="ct_dst_src_ltm">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_src_ltm" required type="text" name="ct_src_ltm">
    </div>

    <div class="form-group">
      <input class="form-control" placeholder="ct_srv_dst" required type="text" name="ct_srv_dst">
    </div>


    <input class="btn btn-primary btn-lg btn-block" type="submit" value='Predict'>

  </form>
</body>

</html>
```

And This is our output page or, result.html.


result.html ×
NtAtCtDt > templates > result.html > html > body.center > h1 > h1 > h1 > h1 > h1 > h1 > h1

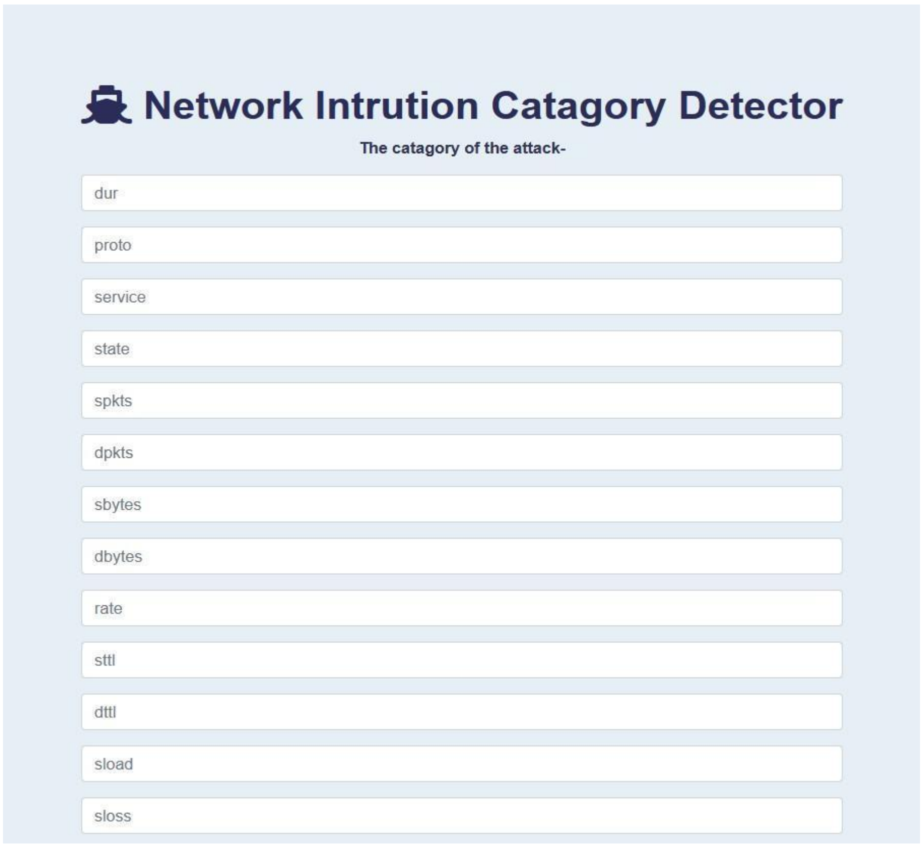
```html

<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>
  <meta charset="utf-8">
  <title>Result</title>

  <script src="https://kit.fontawesome.com/47101d2035.js" crossorigin="anonymous"></script>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

  <style>
    .center {
      position: absolute;
      text-align: center;
      left: 50%;
      top: 50%;
      transform: translate(-50%, -50%);
      font-family: Arial, Helvetica, sans-serif;
      background-color: rgb(230, 239, 245);
    }
  </style>
</head>

<body class="center">

  {% if ans == 0 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Analysis

  {% elif ans == 1 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Backdoor

  {% elif ans == 2 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> DoS

  {% elif ans == 3 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Exploits

  {% elif ans == 4 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Fuzzers

```



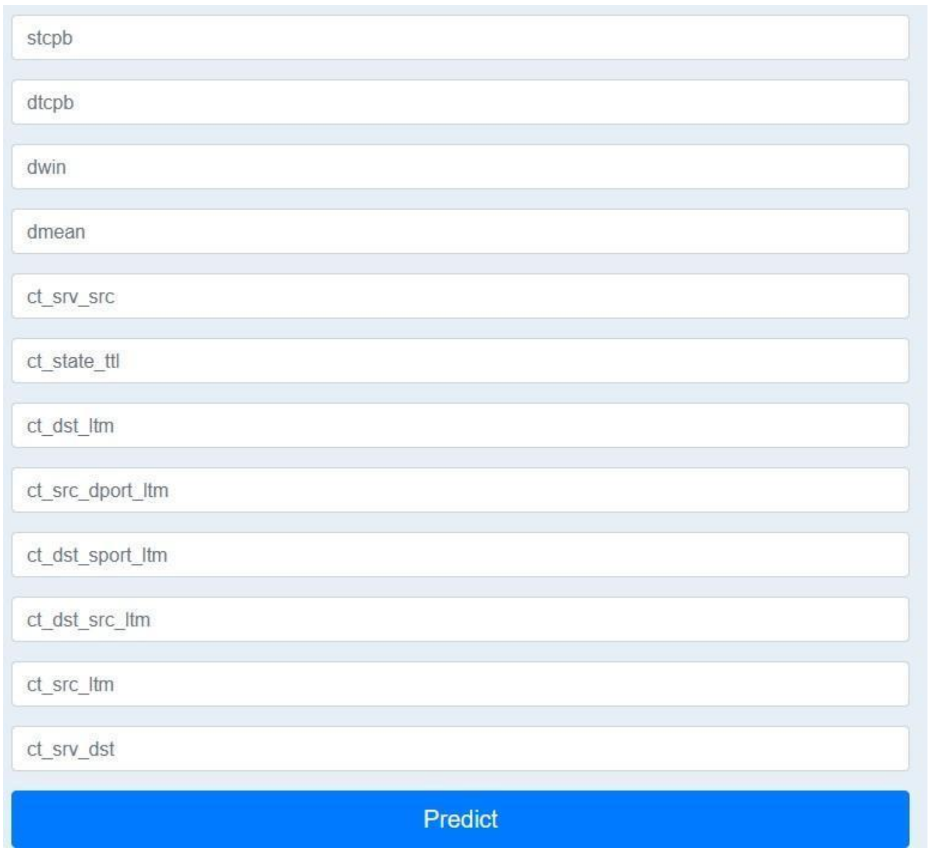
```html
  {% elif ans == 1 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Backdoor

  {% elif ans == 2 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> DoS

  {% elif ans == 3 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Exploits

  {% elif ans == 4 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Fuzzers

  {% elif ans == 5 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Generic

  {% elif ans == 6 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Normal

  {% elif ans == 7 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Reconnaissance

  {% elif ans == 8 %}
  <h1 style=" color: rgb(88, 180, 88); font-weight: 900;">
  <i class="fas fa-swimmer"></i> Shellcode


  {% elif ans == 9 %}
  <h1 style="color: rgb(212, 85, 85); font-weight: 900;">
  <i class="fas fa-skull"></i> Worms

  {% else %}
  <h1 style="color: rgb(31, 15, 15); font-weight: 900;">ERROR</h1>
  {% endif %}

/body>

/html>
```

Here, is the demonstration of the Input and output page means, real implementation.

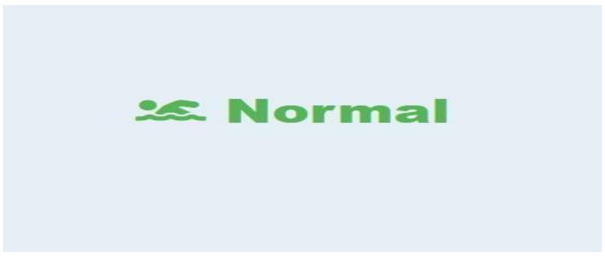
This is the input page which take 28 attributes values as input.

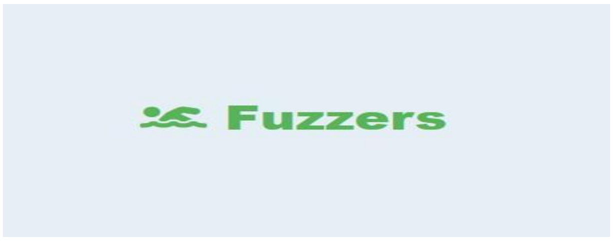

Network Intrution Catagory Detector

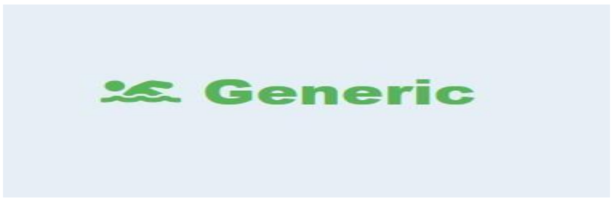
The catagory of the attack-



After checking for values for different attacks from the testing dataset it gives these outputs and these all are correct. So, it's working very efficiently. These are outputs.







DoS

Analysis

backdoor

6.4 Hardware and software Acquisition:

Hardware Requirements:

Server or computer: A dedicated server or computer is required to host the Network Attack Detection App. Sufficient processing power: The hardware should have enough CPU power to handle real-time network traffic analysis and model training. Sufficient memory: Sufficient RAM is necessary to accommodate the dataset, perform data processing, and train machine learning models. Network interface: The hardware should have a network interface card to capture network traffic data.

Software Requirements:

Operating system: Choose an operating system that is compatible with the selected machine learning framework and tools. Python: Install Python, a widely used programming language for data analysis and machine learning. Data processing libraries: Install necessary libraries such as Pandas, NumPy, and Scikit-learn to preprocess the dataset and extract features. Machine learning framework: Install a machine learning framework such as TensorFlow, or Scikit-learn to train and deploy machine learning models. Network traffic capturing tool: Install a tool such as Wireshark or Tcpdump to capture and analyze network traffic data. IDE or text editor: Choose an IDE or text editor such as PyCharm, Jupyter Notebook, or Visual Studio Code for coding and development. Alert generation mechanism: Implement an alert generation mechanism, which may involve system notifications, email services, or integration with a centralized monitoring system.

Dataset Acquisition:

Download the UNSW15 dataset from the official source or a trusted repository. Ensure that the dataset is compatible with the selected machine learning framework and can be loaded into the programming environment. If required, preprocess the dataset to handle missing values, clean the data, and remove irrelevant columns. Split the dataset into input features and target labels for training and evaluation.

Deployment and Scalability Considerations:

Choose a deployment strategy based on the anticipated workload and scalability requirements. Consider containerization technologies such as Docker to encapsulate the Network Attack Detection App and its dependencies for easy deployment and scalability. If the workload increases, you may need to consider distributed computing and cloud infrastructure to handle the increased traffic and model training.

Monitoring and Maintenance:

Implement monitoring mechanisms to track the performance of the Network Attack Detection App and identify any anomalies or issues. Set up a maintenance schedule to ensure regular updates, security patches, and monitoring of the hardware and software components. Continuously evaluate the performance of the machine learning model and consider retraining it periodically to adapt to changing attack patterns

6.5 Installation:

There is no need to installation, because, it’s a demo project for experiment.

6.6 Testing (Unit testing, Integration and system Testing):

Testing (Unit Testing, Integration, and System Testing) for Network Attack Detection App:

Unit Testing: Perform unit testing to validate the individual components of the Network Attack Detection App. Write test cases to cover different functionalities and scenarios. Test the data preprocessing functions to ensure correct handling of missing values, data cleaning, and feature engineering. Test the feature extraction functions to ensure the correct extraction of relevant features from the input page. Test the machine learning model training and evaluation functions to verify the accuracy of the model's predictions. Use assertions or testing frameworks (e.g., unittest, pytest) to automate the testing process and verify expected outcomes.

Integration Testing: Conduct integration testing to ensure the proper functioning of the integrated components within the Network Attack Detection App. Test the integration between the data preprocessing, feature extraction, and model training modules to ensure seamless data flow and compatibility. Verify the integration between the real-time traffic monitoring mechanism and the feature extraction and classification modules. Check the integration between the alert generation mechanism and the model's predictions for accurate and timely alert generation.

System Testing: Perform system testing to evaluate the overall performance, functionality, and reliability of the Network Attack Detection App.

Test the app's ability to handle real-time network traffic and classify it accurately. Conduct performance testing to assess the app's responsiveness and scalability under different loads and traffic conditions. Verify the alert generation mechanism to ensure that it triggers alerts correctly for detected network attacks. Test the app's compatibility with different operating systems, environments, and network setups. Evaluate the app's security measures to protect against false positives and false negatives in attack detection.

Test Automation and Regression Testing: Automate the testing process by writing test scripts or utilizing testing frameworks to streamline the testing procedure. Set up a regression testing suite to ensure that modifications or updates to the app do not introduce new bugs or affect existing functionality. Continuously run the test suite to verify the stability and correctness of the Network Attack Detection App.

Test Reporting and Documentation: Document the test cases, test results, and any issues or bugs encountered during testing. Create a test report summarizing the testing process, outcomes, and any recommendations or improvements for the app. Maintain proper documentation for the testing procedures, including test cases, test scripts, and test data.

By conducting unit testing, integration testing, and system testing, you can ensure that the Network Attack Detection App performs as expected, accurately detects network attacks, and generates timely alerts. Test automation and regression testing help maintain the stability and reliability of the app, and proper documentation facilitates easy understanding and future maintenance.

6.7 Maintenance

By conducting unit testing, integration testing, and system testing, you can ensure that the Network Attack Detection App performs as expected, accurately detects network attacks, and generates timely alerts. Test automation and regression testing help maintain the stability and reliability of the app, and proper documentation facilitates easy understanding and future maintenance.

7 Conclusions and Recommendations

This paper provides an extensive review of the network intrusion detection mechanisms based on the ML and DL methods to provide the new researchers with the updated knowledge, recent trends, and progress of the field. A systematic approach is adopted for the selection of the relevant articles in the field of AI-based NIDS. Firstly, the concept of IDS and its different classification schemes is elaborated extensively based on the reviewed articles. Then the methodology of each article is discussed and the strengths and weaknesses of each are highlighted in terms of the intrusion detection capability and complexity of the model.The empirical analysis from this research suggests that our proposed approach using various machine learning methodologies using UNSW15 dataset performs reasonably well in all categories of majority attacks. The low false positive rate obtained in classifying attacks as well as normal instances makes our approach more interesting. We also perform a cost sensitive classification and found the models achieve low misclassification cost. Finally, it can be noted that no system is absolutely secure with a given set of best possible algorithms, while protecting our resources from network attacks. This makes the computer security is always an active and challenging area of research. The empirical analysis from this research suggests that our proposed approach using various machine learning methodologies using UNSW15 dataset performs reasonably well in all categories of majority attacks. To move further in this direction, we propose to evaluate more machine learning algorithms to detect minority attacks efficiently with acceptable false positive rate and low cost of misclassification in future.

References

[1] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, “Network

intrusion detection system: A systematic study of machine learning and Deep Learning

Approaches,” Transactions on Emerging Telecommunications Technologies, vol. 32, no. 1,

2020. doi:10.1002/ett.4150

[2] C. Sinclair, L. Pierce, and S. Matzner, “An application of machine learning to network

intrusion detection,” Proceedings 15th Annual Computer Security Applications Conference

(ACSAC’99). doi:10.1109/csac.1999.816048

[3] R. Lopez Perez, F. Adamsky, R. Soua, and T. Engel, “Machine learning for reliable

network attack detection in SCADA systems,” 2018 17th IEEE International Conference

On Trust, Security And Privacy In Computing And Communications/ 12th IEEE

International Conference On Big Data Science And Engineering (TrustCom/BigDataSE),

2018. doi:10.1109/trustcom/bigdatase.2018.00094

[4] J. Alsamiri and K. Alsubhi, “Internet of things cyber attacks detection using machine

learning," International Journal of Advanced Computer Science and Applications, vol. 10,

no. 12, 2019. doi:10.14569/ijacsa.2019.0101280

[5] Y. A. Farrukh, Z. Ahmad, I. Khan, and R. M. Elavarasan, "A sequential supervised

machine learning approach for cyber attack detection in a smart grid system," 2021 North

American Power Symposium (NAPS), 2021. doi:10.1109/naps52732.2021.9654767

[6] Y. Xin et al., "Machine learning and deep learning methods for cybersecurity," IEEE

Access, vol. 6, pp. 35365–35381, 2018. doi:10.1109/access.2018.2836950

[7] B. Geluvaraj, P. M. Satwik, and T. A. Ashok Kumar, "The future of cybersecurity:

Major role of Artificial Intelligence, Machine Learning, and Deep Learning in cyberspace,"

International Conference on Computer Networks and Communication Technologies, pp.

739–747, 2018. doi:10.1007/978-981-10-8681-6_67