Feature Analysis and Random Forest Classification for Network Intrusion Detection

Devika G, Manasa S P, Sudeep K, Shariya Habiba K S, Shivanand Mirji
Department of Computer Science and Engineering, Government Engineering College, Kushalnagar Karnataka, India

Abstract: **The increasing frequency and sophistication of cyberattacks have highlighted the critical need for intelligent and adaptive Intrusion Detection Systems (IDS). This project presents the design and implementation of a machine learning-based IDS utilizing the Random Forest algorithm to detect various types of network intrusions with high accuracy and efficiency. The system is trained and tested using benchmark datasets such as NSL-KDD, enabling it to learn and classify patterns of normal and malicious activities. Data preprocessing steps, including feature selection, normalization, and data balancing, were applied to improve model performance. The Random Forest classifier was chosen for its robustness, interpretability, and ability to handle complex and high-dimensional data. Experimental results demonstrate that the proposed system achieves high detection rates and low false positive rates, outperforming several traditional and contemporary machine learning approaches. This project contributes to the development of scalable and effective intrusion detection solutions suitable for modern network environments.**

Keywords: **Intrusion Detection System (IDS), Machine Learning, Random Forest Algorithm, Cybersecurity, Network Security, KDD Cup 99 Dataset**

I. INTRODUCTION

Intrusions in computer systems are a serious and ongoing threat. From the beginning of computing to today's connected digital world, cyberattacks have continued to grow in number and complexity. Even though many security measures have been developed over the years, attackers keep finding new and smarter ways to break into systems.

Today, computers are a key part of almost every area of life and business. That's why protecting them from malicious activities is more important than ever. One of the key tools used for this purpose is the Intrusion Detection System (IDS). An IDS helps monitor network activity and alerts users if it finds something suspicious or harmful.

Our project focuses on improving how accurately these systems can detect threats. To do this, we need to analyze large amounts of data, which is not easy to handle manually. So, we use machine learning, specifically the Random Forest algorithm, to process this data and improve detection.

The Random Forest algorithm is good at recognizing patterns in complex data. In our system, it is trained using network data that shows both normal and attack behaviors. After training, the system can detect different types of intrusions when new data is given to it.

II. LITERATURE REVIEW

[1] Amrith Bhatnang introduced a federated learning-based IDS model, emphasizing privacy preservation by training local models on decentralized data and aggregating them into a global model. This approach showed effective anomaly detection capabilities while maintaining data confidentiality, highlighting federated learning as a secure and scalable alternative to centralized systems.

[2] Yoheswari S developed an optimized IDS using Support Vector Machine (SVM) algorithms. The model incorporated feature selection, dimensionality reduction, and hyperparameter tuning through Grid Search and Particle Swarm Optimization (PSO). This resulted in improved accuracy, reduced false positives, and enhanced computational efficiency, making it suitable for dynamic cybersecurity environments.

[3] Naga Durga Srinivas Nidamanuri proposed an IDS using genetic algorithms and a modified J48 decision tree, leveraging protocol standardization techniques. This method reduced search space and improved detection speed and consistency, particularly useful in environments requiring fast response times and IPv6 support.

[4] Vincent Zibi Mohale and Ibidun Christiana Obagbuwu presented a systematic review of Explainable Artificial Intelligence (XAI) in IDS. Their analysis revealed that techniques like SHAP and LIME improve model transparency and interpretability but can compromise detection accuracy. They stressed the need for hybrid models and standard evaluation metrics to balance explainability with performance.

[5] Basheer Ullah et al. (2025) applied Deep Neural Networks (DNN) to the NSL-KDD dataset and reported a 92% accuracy rate. They also noted that combining DNN with algorithms like Random Forest and Decision Trees could further enhance detection rates, suggesting a promising direction in hybrid modeling.

[6] Arun Kumar Silivery and Ram Mohan Rao Kovvur (2023) investigated Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) with various optimizers, finding that the Adamax optimizer yielded the best results. Their model showed high accuracy, strong detection rates, and low false alarms, proving the effectiveness of deep learning in multi-attack classification.

[7] Sai Srinivas Vellela et al. (2024) developed an ensemble learning-based IDS using the DARPA dataset. Their approach involved extracting features from FTP traffic and using multiple learning models to boost detection accuracy. Among the models tested, the Multi-Layer Perceptron (MLP) with 30 neurons showed notable performance improvements.

III. METHODOLOGY

1. System Architecture

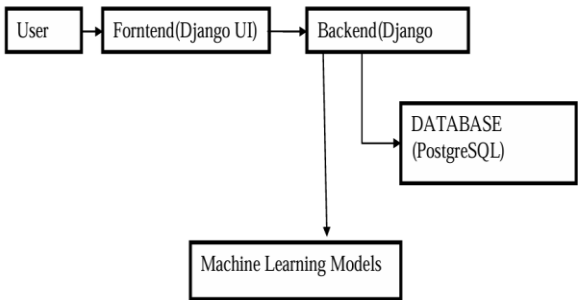

Fig3.1: System architecture

The proposed Intrusion Detection System (IDS) is designed to efficiently detect and classify network intrusions using machine learning techniques, specifically the Random Forest algorithm. This system architecture integrates a trained classification model with a robust web-based user interface, ensuring both accuracy in detection and usability for end-users.

1. Dataset Used:
   - Utilizes the KDD CUP 99 dataset, a standard benchmark in IDS research, for training the machine learning model on different types of network attacks.
2. Core Components of the System:
   - The system is composed of three major modules:
     - Frontend Interface
     - Backend Application
     - Machine Learning Module
3. Frontend Interface:

Frontend development involves the part of a website or web application that users directly interact with. This is also known as the client-side of web development. It focuses on the visual layout, structure, design, and user experience.

A frontend developer's goal is to ensure the interface is visually appealing, functional, and responsive across all devices. The key technologies used in frontend development are:

- Core Technologies in Frontend Development

1. HTML (HyperText Markup Language):
- Provides the basic structure of web pages.
- Acts as the skeleton for all web content.

2. CSS (Cascading Style Sheets):
- Responsible for styling the HTML content.
- Includes layout, colors, fonts, and overall visual aesthetics.

- Extended by:
  - CSS Libraries: E.g., Pure CSS
  - CSS Frameworks: Bootstrap, Bulma, Foundation, Materialize CSS, Semantic UI, Tailwind CSS
- CSS Preprocessors: SASS, LESS – These provide advanced features like variables, nesting, and functions.

3. JavaScript:
- Adds interactivity and dynamic behavior to web applications.
- Enables features such as animations, form validations, and content updates without refreshing the page.

- Includes:
  - ES6 (ECMAScript 6): The modern version of JavaScript that introduces new features like arrow functions, classes, and modules.

- TypeScript: A superset of JavaScript that adds static typing for better code management and error checking.
- JavaScript Libraries: jQuery, D3.js, Lodash, Underscore.js, Fabric.js, p5.js, etc.
- JavaScript Frameworks: AngularJS, ReactJS, Vue.js – These frameworks help in building complex, component-based applications.

➢ Supporting Tools
- Tools like npm (Node Package Manager) and yarn are used to manage project dependencies and libraries, making development more efficient and scalable.

4. Backend Application (Django Framework):

Backend development, also known as server-side development, plays a crucial role in modern web applications by handling the logic, database operations, and performance-related functionality that underpins the user experience.

Backend development is powered by a wide array of programming languages and frameworks. Each ecosystem offers tools and best practices tailored for different needs.

1. PHP

PHP is one of the oldest and most widely adopted server-side scripting languages. It powers many content management systems (CMS) and frameworks.

- Package Manager: Composer
- Testing Tool: PHPUnit
- Framework: Laravel
- Popular CMS: WordPress, Joomla, Drupal, Magento

2. Node.js

Node.js is a JavaScript runtime built on Chrome's V8 engine, allowing developers to use JavaScript for both frontend and backend development.

- Package Managers: npm, yarn
- Framework: Express.js

3. Python

Python is recognized for its clean syntax and robust libraries, making it suitable for both simple applications and complex, data-driven systems.

- Frameworks: Django (full-featured), Flask (lightweight)
- Package Manager: pip

4. Ruby

Ruby offers elegant syntax and is most famously paired with Ruby on Rails, a full-stack framework.

- Framework: Ruby on Rails

5. Java

Java is widely used in enterprise environments due to its robustness, scalability, and extensive ecosystem.

- Framework: Spring

6. Golang (Go)

Go is known for its simplicity, speed, and efficiency, making it ideal for microservices and high-performance applications.

7. C#

C# is a language developed by Microsoft, commonly used in desktop and web applications on Windows platforms.

- Framework: .NET

➢ Data Storage and Databases

Data persistence is a cornerstone of backend systems. Storage can be categorized into:

1. Relational Databases (SQL)

Relational databases store structured data and use structured query language (SQL) for data manipulation.

- Examples: PostgreSQL, MariaDB, MySQL

2. NoSQL Databases

NoSQL databases are designed for flexible, scalable storage of unstructured or semi-structured data.

- Example: MongoDB

5. Machine Learning Module:
   - Contains a pre-trained Random Forest model, loaded via joblib.
   - Uses pandas and numpy for:
     - Data preprocessing (cleaning, formatting).
     - Ensuring compatibility with the model's input requirements.
   - Outputs a classification label indicating the type of traffic (e.g., Normal, DOS, Probe, R2L, U2R).
6. System Workflow (Data Flow):
   - User logs in through the web interface.
   - Submits network traffic features through structured forms.
   - Data is sent to the Django backend for processing.
   - Backend passes the data to the Random Forest model.
   - Model returns a prediction, which is displayed on the frontend.

2. Entity relationship model

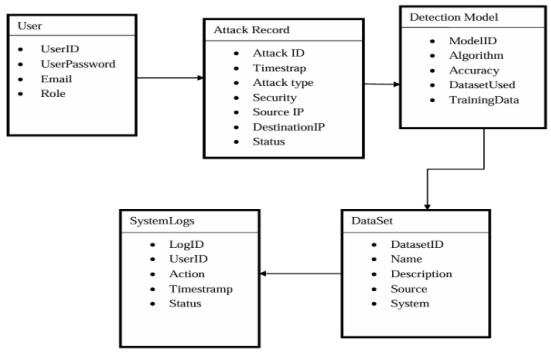

Fig 3.2: Entity Relationship model

The entity relationship model in the image shows the relationships between different components of the Intrusion Detection System (IDS).

- Detection Model: This entity stores information about the detection models used by the system, including the model ID, algorithm, accuracy, dataset used, and training data.
- Attack Record: This entity records details of detected attacks, such as attack ID, timestamp, attack type, security, source IP, destination IP, and status.
- User: This entity stores user information, including user ID, password, email, and role.
- DataSet: This entity contains information about the datasets used for training the models, such as dataset ID, name, description, and source.
- SystemLogs: This entity logs system activities, including log ID, user ID, action, timestamp, and status.

IV. IMPLEMENTATION

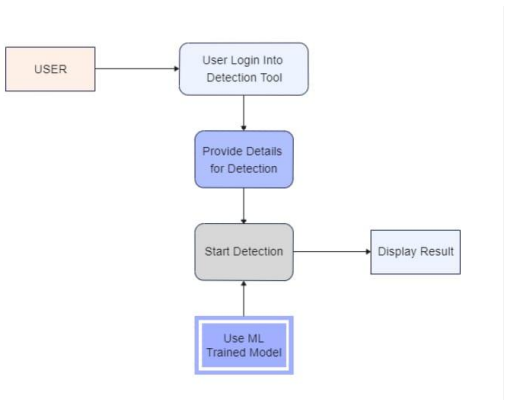

Fig 4.1 System Implementation Flowchart

The proposed IDS features a role-based, web-based interface that enables real-time intrusion detection through a streamlined and interactive user experience. The process begins with secure user authentication, after which users input network parameters required for attack detection. The system guides users through multi-step data entry forms, and upon submission, a pre-trained machine learning model (Random Forest) analyzes the input. Detection results are immediately shown in a popup alert, identifying the type of attack.

1. UI Design

The interface consists of five main HTML templates:

- index.html: Login page
- dashboard.html: Displays dataset insights and starts detection
- home.html, home_2.html, home_3.html: Collect user inputs in stages

The workflow allows users to explore the KDD dataset, fill in detection-related inputs across multiple screens, and trigger the model to generate a detection output, displayed as an alert.

2. Django Framework

The system is built using Django, a high-level Python web framework following the Model-View-Template (MVT) architecture. Django handles authentication, form validation, and integration with the ML model, while allowing each module (login, detection, logging) to function independently, ensuring scalability and modularity. logging) to function independently, ensuring scalability and modularity.

3. Technologies Incorporated

Frontend Development: Uses HTML, CSS, Bootstrap, and Jinja2 to build a responsive, user-friendly interface.

Backend Development: Powered by Django, it handles data processing, authentication, and interaction with the machine learning model.

This full-stack design offers a robust, scalable, and intuitive IDS platform, suitable for real-time network threat detection.

4. Machine Learning Technique & Algorithm

- Machine learning is a type of artificial intelligence.
- It helps computers learn from data and make decisions or predictions on their own.
- These computer programs get better the more data they use.
- Machine learning finds patterns in big sets of data to help make smart guesses.

Random Forest:

- Random Forest is a machine learning method that uses many decision trees.
- A decision tree is like a flowchart that helps make decisions.

5. Classification of Cyber-Attacks Detected by IDS

➢ Denial of Service (DoS)

A Denial of Service attack is aimed at disrupting the normal functionality of a network, service, or system by overwhelming it with excessive traffic or resource requests.

- Objective: To render a service or system unavailable to legitimate users.
- Example: Sending thousands of requests per second to a server, causing it to crash or slow down.
- Analogy: Similar to a scenario where too many students try to enter a small classroom at once, resulting in chaos and inaccessibility.

➢ Probe Attacks

Probe attacks are reconnaissance attempts where attackers scan networks to identify vulnerabilities or open ports that can later be exploited.

- Objective: To gather information about the target system for future exploitation.
- Example: Port scanning tools like Nmap are used to detect active devices and services.
- Analogy: Comparable to a burglar walking through a neighborhood and checking which doors are unlocked.

➢ Remote to Local (R2L)

In R2L attacks, the attacker gains unauthorized access to a system over a network, typically by exploiting weak credentials or application vulnerabilities.

- Objective: To gain user-level access on a target machine from a remote location.
- Example: A hacker guessing or stealing login credentials via phishing.
- Analogy: Like someone trying to guess your email password from afar to log in as you.

➢ User to Root (U2R)

User to Root attacks involve an intruder who already has limited access to a system (as a regular user) escalating their privileges to gain full control, or root-level access.

- Objective: To exploit system vulnerabilities and gain administrative control.
- Example: Buffer overflow exploits or privilege escalation bugs.
- Analogy: Similar to a student gaining access to the school's administrative system and assigning themselves as principal.

V. RESULT AND DISCUSSION

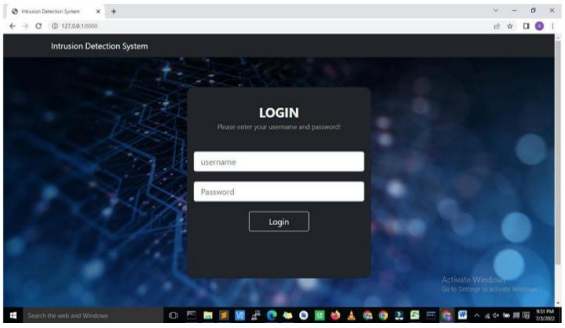

Fig 5.1 Index Page

The index page of an Intrusion Detection System (IDS) is the first screen users see when they open the system. It asks users to enter a username and password to log in. This helps make sure that only authorized people can use the system. The page is easy to use, with clearly marked boxes for entering the login details and a "Login" button to continue. This process helps protect the system from unauthorized access and keeps track of who is using it.

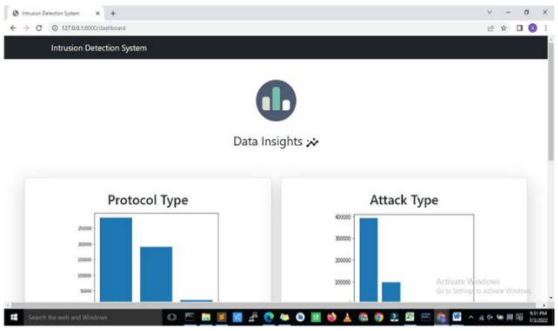

Fig 5.2 Dashboard Page

After logging in, users are taken to the Intrusion Detection System's dashboard, which gives a quick and clear overview of network activity. In the center, a "Data Insights" section highlights important information. On the left side, a bar chart shows the types of communication protocols being used in the network. On the right, another chart displays different types of attacks the system has detected. These visuals help users easily understand what is happening on the network and identify any suspicious activity that may need attention.

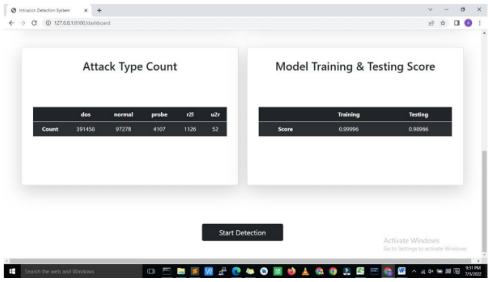

Fig 5.3 Dashboard Page with Start Detection Button

This dashboard gives a detailed look at how well the Intrusion Detection System is working. A table called "Attack Type Count" shows how many times different types of threats—like DoS, probe, and others—have been detected. Next to it, the "Model Training & Testing Score" section displays how accurately the system's machine learning model performs. The high scores (almost 100%) show that the model can detect threats reliably. The "Start Detection" button lets users begin real-time monitoring of network traffic using the trained model. system's performance data with the start of live attack detection.

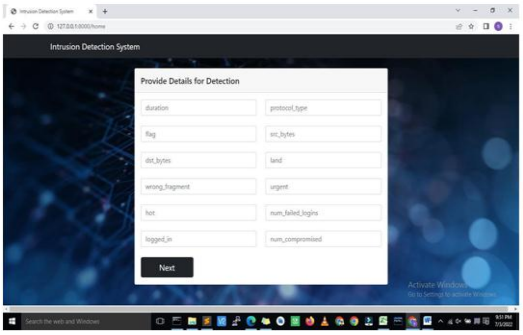

Fig 5.4 Details Page for detection

This section of the system asks users to enter details about a network connection to check for possible threats. Some fields describe how the connection works, while others focus on data from the network packets or user activity. These inputs help the system better understand the traffic and look for signs of an attack. After entering the details, the user clicks "Next" to continue with the detection process.

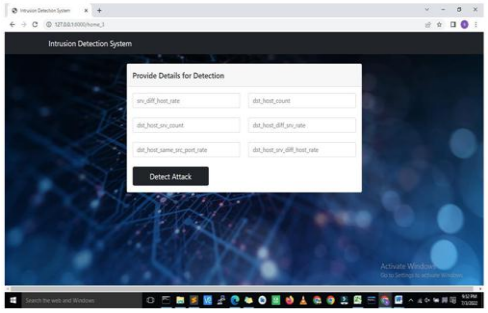

Fig 5.5 Details page for detection with start button

In this step, users enter more specific network details to help the system check for threats. These include how often different hosts are contacted and how consistent the traffic is. After filling in the information, clicking the "Detect Attack" button starts the process. This helps the system find suspicious activity more accurately.

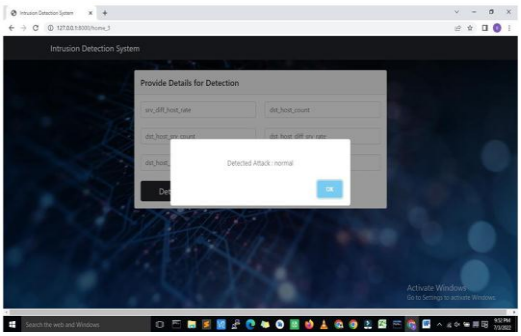

Fig 5.6 Result of Detection

After clicking "Detect Attack," a pop-up message appears showing "Detected Attack: normal." This means the system checked the network details and found no signs of an attack. The user can click "Ok" to close the pop-up and continue using the system. This message confirms that the network activity is safe based on the system's analysis.

VI. CONCLUSION

This paper introduces a user-friendly and accurate Intrusion Detection System (IDS) capable of detecting various types of network attacks. The system features a responsive interface, making it easy to use for individuals with different levels of technical expertise. It employs the Random Forest algorithm, which is well-known for its effectiveness in handling complex datasets and delivering high accuracy. By training the model on the KDD CUP 99 dataset—a widely recognized benchmark in the field—the IDS is able to recognize and respond to a broad range of attack types, including DoS, Probe, R2L, and U2R.

The high accuracy of the system ensures that network administrators receive timely and reliable alerts, improving their ability to respond to threats efficiently. The tool not only boosts security but also provides valuable insights in an easy-to-understand format. These improvements would strengthen the IDS's capabilities and adaptability, making it an even more effective cybersecurity solution in a constantly evolving digital environment.

VII. REFERENCE

[1] Bhatnang, A. (2025). Federated learning-based intrusion detection system for the Internet of Things using unsupervised and supervised deep learning models. *Cyber Security and Applications*, 3, 100068.

[2] Yoheswari, S. (2024). Optimized intrusion detection model for identifying known and innovative cyber attacks using support vector machine (SVM) algorithms. *Journal of Science Technology and Research (JSTAR)*, 5(1), 398–404.

[3] Nidamanuri, N. D. S. (2024). Intrusion detection using genetic algorithms and protocol-aware decision trees. *Journal of Network and Computer Applications*, 110, 98–110.

[4] Mohale, V. Z., & Obagbuwu, I. C. (2024). A review of explainable AI techniques in intrusion detection systems. *ACM Computing Surveys*, 56(4), Article 78.

[5] Ullah, B., Ahmed, S., & Khan, M. A. (2025). Hybrid deep learning models for intrusion detection: A case study on NSL-KDD. *IEEE Access*, 13, 23015–23028.

[6] Silivery, A. K., & Kovvur, R. M. R. (2023). Enhancing intrusion detection using LSTM-RNN with optimized training. *Journal of Information Security and Applications*, 72, 103404.

[7] Vellela, S. S., Reddy, K. R., & Patnaik, S. (2024). Ensemble learning-based IDS for FTP traffic analysis using DARPA dataset. *Computers & Security*, 134, 103219.