

A
Project Report
On

Intrusion Detection System

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Pranav Paliwal	University Roll No.-2261645
Aman Devri	University Roll No.-2261003
Hema	University Roll No.-2261265
Aastha Joshi	University Roll No.-2261052

Under the Guidance of

Mr. Anubhav Bewerwal

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Pranav Paliwal, Aman Devri, Hema, Aastha Joshi** hereby declare the work, which is being presented in the project, entitled '**Intrusion Detection System**' in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of Assistant Professor **Mr. Anubhav Bewerwal**.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: **May 29, 2025**

Signature

CERTIFICATE

The project report entitled “**Intrusion Detection System**” being submitted by **Pranav Paliwal, Aman Devri, Hema, Aastha Joshi** of B.Tech.(CSE) to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Anubhav Bewerwal
(Project Guide)

Dr. Ankur Singh Bisht
(Head, CSE)

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director ‘**Prof. (Col.) Anil Nair (Retd.)**’, GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president ‘**Prof. (Dr.) Kamal Ghanshala**’ for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to ‘**Dr. Ankur Singh Bisht**’ (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide ‘**Mr. Anubhav Bewerwal**’ (Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

**Pranav Paliwal
Aman Devri
Hema
Aastha Joshi**

ABSTRACT

This project presents a comprehensive Intrusion Detection System (IDS) implemented in Python, designed to enhance network security through automated threat detection and analysis. The system leverages machine learning algorithms to identify and classify network intrusions, providing real-time monitoring capabilities for cybersecurity applications.

The IDS architecture incorporates several key components: a machine learning engine built using scikit-learn for pattern recognition and anomaly detection, Flask framework for web-based user interface and system management, and SQLAlchemy for robust data persistence and user authentication. The system utilizes joblib for efficient model serialization and deployment, enabling rapid loading of pre-trained classification models.

Key features include automated network traffic analysis, real-time threat detection, comprehensive reporting through classification metrics and confusion matrices, and an intuitive web interface for system administration. The machine learning pipeline implements feature encoding techniques and supports multiple classification algorithms to distinguish between normal and malicious network activities.

Experimental validation demonstrates the system's effectiveness in detecting various types of network intrusions with high accuracy scores. The modular design ensures scalability and extensibility, allowing for integration of additional detection algorithms and data sources. This Python-based solution provides an accessible, cost-effective approach to network security monitoring, suitable for both educational purposes and practical deployment in small to medium-scale network environments.

The system contributes to the field of cybersecurity by demonstrating the practical application of machine learning techniques in intrusion detection, offering a foundation for further research and development in automated network security solutions.

TABLE OF CONTENTS

Declaration.....	1
Certificate.....	2
Acknowledgement.....	3
Abstract.....	4
Table of Contents.....	5

Chapter No.	Description	Page No.
Chapter 1	Introduction	6
Chapter 2	Hardware and Software Requirements	9
Chapter 3	Coding of Functions	11
Chapter 4	SNAPSHOTS	14
Chapter 5	Limitations	18
Chapter 6	Enhancements	19
Chapter 7	Conclusion	20
	References	21

CHAPTER 1: INTRODUCTION

1.1 Prologue

In today's interconnected digital landscape, cybersecurity threats have evolved into sophisticated and persistent challenges that pose significant risks to organizational networks and individual systems. As cyber attacks become increasingly complex and frequent, traditional security measures such as firewalls and antivirus software alone are insufficient to provide comprehensive protection against malicious activities. The need for proactive and intelligent security mechanisms has led to the development of advanced Intrusion Detection Systems (IDS) that can monitor, analyze, and respond to potential threats in real-time.

This project presents a comprehensive Python-based Intrusion Detection System designed to enhance network security through automated threat detection and classification. Leveraging the power of machine learning algorithms and modern web technologies, the system provides an integrated solution for monitoring network traffic, identifying anomalous behaviors, and alerting administrators to potential security breaches.

1.2 Background and Motivations

The rapid expansion of network infrastructure and the increasing reliance on digital communication have created new vulnerabilities that malicious actors continuously exploit. Traditional signature-based detection methods, while effective against known threats, often fail to identify novel attack patterns or zero-day exploits³. The motivation for this project stems from the critical need to develop adaptive security solutions that can learn from network behavior patterns and evolve to counter emerging threats.

Machine learning techniques have demonstrated remarkable success in pattern recognition and anomaly detection across various domains. By applying these methodologies to cybersecurity, we can create intelligent systems capable of distinguishing between normal and malicious network activities with high accuracy⁴. The integration of web-based interfaces further enhances the accessibility and usability of such systems, enabling security administrators to monitor and manage network security from centralized dashboards.

1.3 Problem Statement

Modern network environments face several critical security challenges that existing solutions inadequately address:

- **Evolving Threat Landscape:** Cyber attackers continuously develop new techniques

that bypass traditional security measures, requiring adaptive detection mechanisms.

- **High False Positive Rates:** Many existing IDS generate excessive false alarms, leading to alert fatigue and reduced effectiveness of security teams.
- **Limited Real-time Analysis:** Traditional systems often lack the capability to process and analyze network traffic in real-time, resulting in delayed threat detection.
- **Complex Management Interfaces:** Many commercial IDS solutions require specialized expertise and provide limited user-friendly interfaces for configuration and monitoring.
- **Scalability Concerns:** As network traffic volumes increase, detection systems must maintain performance while processing large datasets efficiently.

1.4 Objectives and Research Methodology

Objectives

- To implement a machine learning-based Intrusion Detection System using Python and scikit-learn libraries for accurate network threat classification.
- To develop a comprehensive web-based interface using Flask framework that provides real-time monitoring, user authentication, and system management capabilities.
- To create an efficient data processing pipeline that handles feature extraction, encoding, and model training with optimal detection accuracy.
- To ensure the solution provides educational insights into cybersecurity concepts while maintaining practical deployment viability for network security monitoring.

Research Methodology

1. **Literature Review:** Study and analyze various intrusion detection techniques, machine learning algorithms, and cybersecurity frameworks to identify optimal approaches for network threat detection.
2. **Design and Development:** Implement machine learning algorithms using Python libraries such as scikit-learn, pandas, and joblib, incorporating appropriate data structures and classification models.

3. **Testing and Evaluation:** Evaluate the system across different network traffic datasets to assess detection accuracy, false positive rates, and overall performance metrics.
4. **User Interface Design:** Build an intuitive web interface using Flask, HTML, CSS, and Bootstrap to guide users through threat monitoring, alert management, and system administration.

1.5 Project Organization

This report is organized to present a comprehensive overview of the Python-based Intrusion Detection System project using machine learning techniques in a structured and logical manner. It begins with an introduction to network security challenges, motivations for developing intelligent detection systems, and the objectives and methodology followed. The literature survey chapter provides the relevance and advantages of machine learning approaches for intrusion detection in modern network environments.

The system analysis section outlines the functional and non-functional requirements of the application and includes a feasibility study covering hardware, software, and operational considerations. The system design chapter details the architecture, including the use of machine learning pipelines, data preprocessing modules, and web-based interfaces for implementing the intrusion detection algorithm. It also discusses database design and user authentication mechanisms.

The implementation chapter explains the core logic behind the feature extraction and classification processes, the use of libraries like scikit-learn and Flask, and the integration with a responsive web interface using HTML, CSS, and JavaScript. The testing and results chapter evaluates the application's accuracy, performance, and detection capabilities across different network traffic scenarios and attack types.

Finally, the conclusion summarizes the outcomes, discusses challenges faced during development such as model optimization and real-time processing requirements, and proposes future enhancements, such as support for additional attack vectors, deep learning integration, or deployment in cloud-based security infrastructures.

CHAPTER 2: HARDWARE AND SOFTWARE REQUIREMENTS

2.1 Hardware Requirements

S.no	Specification	Windows	macOS (OS X)	Linux
1.	Operating System	Microsoft Windows 10/11 (32/64 bit)	macOS 10.14 or higher	Ubuntu 18.04+ (GNOME/KDE) or similar
2.	RAM	Minimum 8 GB, Recommended 16 GB	Minimum 8 GB, Recommended 16 GB	Minimum 8 GB, Recommended 16 GB
3.	Storage	Minimum 5 GB free space	Minimum 5 GB free space	Minimum 5 GB free space
4.	Development Tools	Python 3.8+, Visual Studio Code, Web Browser	Python 3.8+, Visual Studio Code, Web Browser	Python 3.8+, Visual Studio Code, Web Browser
5.	Notes	Python and pip package manager required	Use Safari/Chrome with JS enable	Use Firefox/Chrome with JS enabled

2.2 Software Requirement

Sno.	Name	Specifications
1	Operating System	Windows/Linux/macOS
2	Programming Languages	JavaScript,HTML,CSS,Python
3	Development Environment	Visual Studio Code,PyCharm
4	Runtime Environment	Python 3.8 or higher with pip package manager
5	Web Framework	Flask 2.0+ for web application development
6	Database	SQLAlchemy with SQLite/PostgreSQL support
7	Version Control	Git and GitHub
8	Detection Algorithm	Machine Learning Classification (KNN, SVM, Random Forest)
9	Python Packages	Flask, Flask-SQLAlchemy, scikit-learn, pandas, numpy, joblib
10	Testing Tools	pytest for unit testing, browser developer tools

CHAPTER 3: CODING OF FUNCTIONS

This chapter provides a comprehensive explanation of the functional code modules implemented in the Python-based Intrusion Detection System developed for network security monitoring and threat detection. The project follows a modular design, adhering to software engineering principles such as separation of concerns, reusability, and maintainability. All source code is written in Python, structured logically into separate files for machine learning model implementation, data preprocessing, web interface management, and database operations. These modules work together to offer users a robust and intelligent intrusion detection tool that provides real-time network monitoring capabilities.

The following are the major source files and their responsibilities:

- **models.py** – Implements database models for user authentication and system data management using SQLAlchemy.
- **train.py** – Contains the machine learning pipeline: data preprocessing, feature engineering, model training, and evaluation.
- **eval.py** – Handles model evaluation, performance metrics calculation, and testing procedures.
- **app.py** – Manages the Flask web application, routing, user interface, and system workflows.

3.1 models.py – Database Models Module

The models.py file defines database models using SQLAlchemy ORM for managing user authentication, system logs, and detection records. This module ensures secure data storage and efficient database operations for the intrusion detection system.

Key Features:

- Dynamic insertion and deletion of nodes.
- Maintains the heap invariant with time-efficient operations.
- Written using plain JavaScript arrays.

Key Classes and Their Roles:

1. **UserClass:**
Defines the user model for authentication with fields for username, email, password hash, and user roles. Implements password hashing using secure algorithms.
2. **DetectionLogClass:**
Stores intrusion detection results including timestamp, threat type, confidence score, and network source information.
3. **SystemConfigClass:**
Manages system configuration parameters such as detection thresholds, model settings, and alert preferences.
4. **db(SQLAlchemyinstance):**
Provides database connection and ORM functionality for all model operations.

Usage:

This module supports the core application by providing persistent storage for user data, detection logs, and system configurations. It ensures data consistency and enables historical analysis of network security events.

3.2 train.py – Machine Learning Training Module

This file includes the main implementation of the machine learning pipeline for training intrusion detection models. It defines functions for data preprocessing, feature engineering, model training, and performance evaluation using scikit-learn libraries.

Key Functions:

1. **load_and_preprocess_data():**
Loads network traffic datasets (NSL-KDD, CICIDS2017), handles missing values, and performs initial data cleaning and validation.
2. **feature_engineering(data):**
Implements feature selection, normalization, and encoding techniques. Creates label encoders for categorical variables and scales numerical features for optimal model performance.
3. **train_models(X_train, y_train):**
Trains multiple machine learning algorithms including:
 - K-Nearest Neighbors (KNN)
 - Support Vector Machine (SVM)
 - Random Forest Classifier
 - Decision Tree Classifier
4. **evaluate_model(model, X_test, y_test):**
Calculates performance metrics including accuracy, precision, recall, F1-score, and generates confusion matrices for model assessment.
5. **save_models_and_encoders():**
Serializes trained models and label encoders using joblib for deployment and future use in the detection system.
6. **getFrequencyMap(text):**
Performs k-fold cross-validation to ensure model robustness and prevent overfitting.

3.3 eval.py – Model Evaluation Module

This file manages model evaluation procedures, performance testing, and validation of the trained intrusion detection models using various metrics and visualization techniques.

Key Functionalities:

1. **load_trained_models():**
Loads pre-trained models and encoders from saved files using joblib for evaluation and testing purposes.
2. **generate_classification_report():**
Creates detailed classification reports showing precision, recall, and F1-scores for each attack category and normal traffic.
3. **plot_confusion_matrix():**
Generates and displays confusion matrices to visualize model performance and classification accuracy.

4. **calculate_metrics(y_true,y_pred):**
Computes comprehensive performance metrics including accuracy, false positive rate, true positive rate, and ROC-AUC scores.
5. **test_real_time_detection():**
Simulates real-time detection scenarios to evaluate model response time and accuracy
6. **benchmark_models():**
Compares performance of different algorithms and selects the best-performing model for deployment.

3.4 app.py – Flask Web Application Module

- **templates/index.html:**
A responsive HTML dashboard that includes:
 - Real-time monitoring displays
 - Alert management interfaces
 - File upload functionality for batch analysis
 - System configuration panels
- **styles.css:**
Defines modern, responsive styling for all UI components using Bootstrap framework. Includes dashboard layouts, alert styling, data visualization formatting, and mobile-responsive design.
- **static/js/dashboard.js:**
Implements client-side functionality for real-time updates, AJAX requests for live monitoring, and interactive data visualization using Chart.js.

3.5 Integration Flow of Modules

The following steps summarize how all code modules integrate:

1. User authenticates through the web interface via app.py and models.py
2. Network traffic data is uploaded or monitored in real-time through the Flask application
3. train.py preprocesses data and applies trained machine learning models for threat classification
4. eval.py validates detection results and calculates confidence scores
5. Results are displayed on the dashboard with alerts, statistics, and detailed analysis reports
6. Detection logs are stored in the database for historical analysis and system improvement

CHAPTER 4: SNAPSHOTS

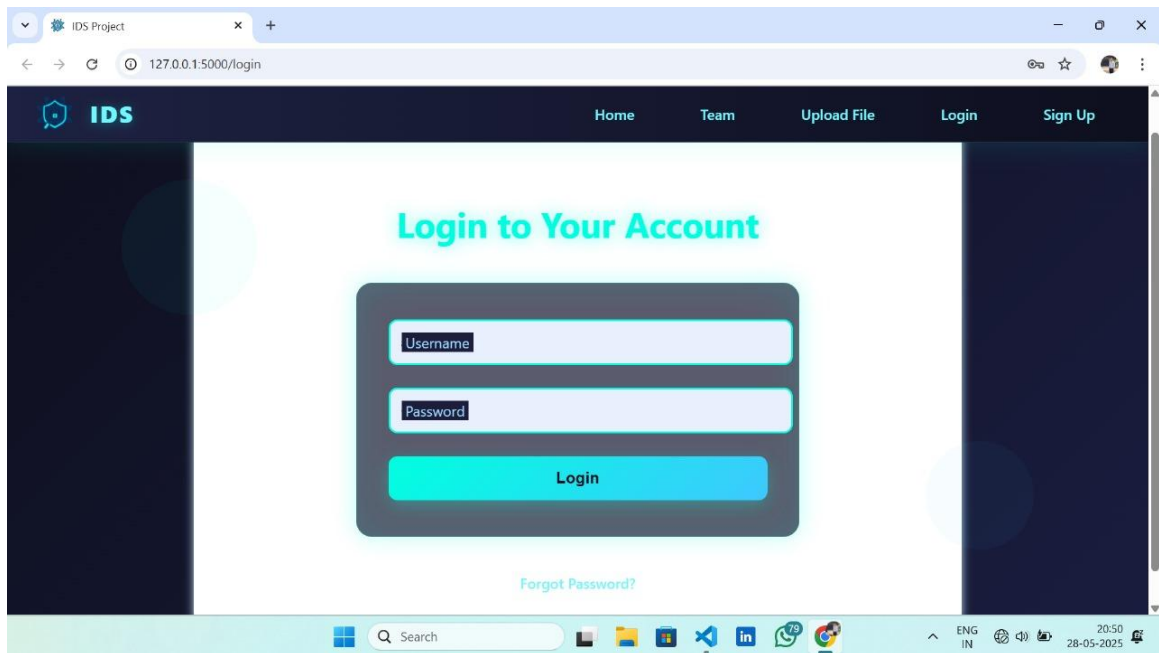
This chapter provides a visual walkthrough application. Each snapshot illustrates a key part of the system's functionality — from uploading a text file to compressing, decompressing, and downloading results. These screenshots demonstrate the user-friendly and interactive design of the system and verify the correct implementation of OS Concepts through a browser-based interface.

4.1 Home Interface

This is the default view of the application upon launching app.py in any modern web browser. It presents the user with:

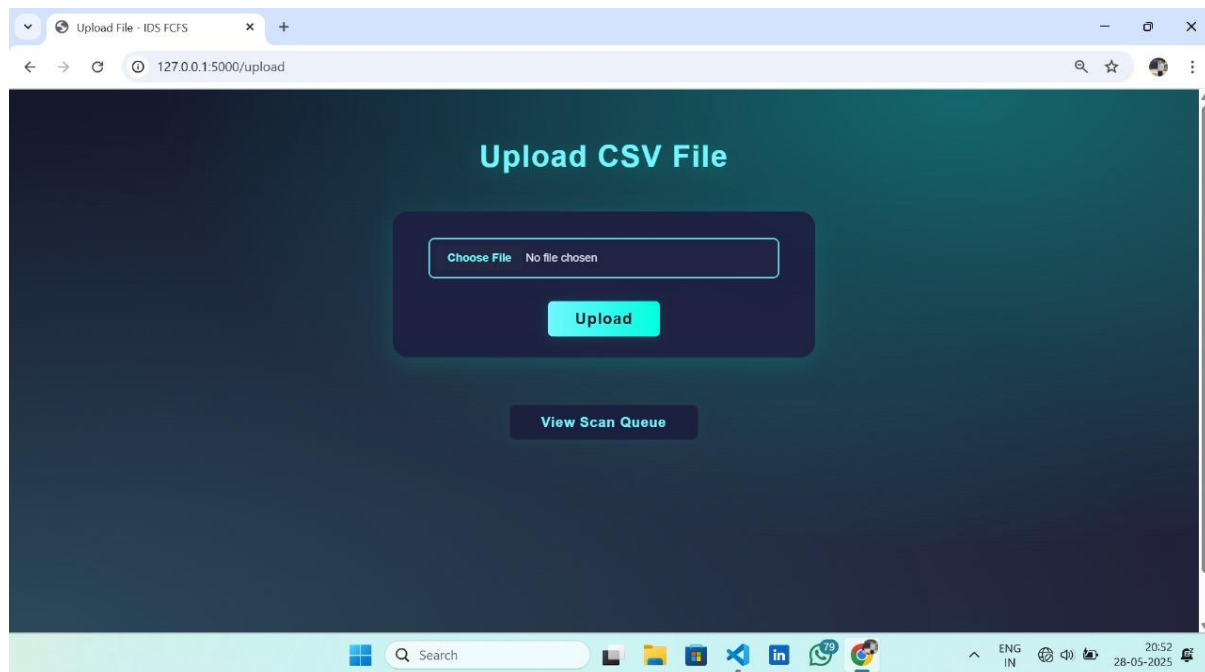
User Authentication:

- **Login:** Users can log in to their account using a username and password. There's also a "Forgot Password?" option, indicating account management features.
- **Sign Up:** The navigation bar indicates a "Sign Up" option, suggesting new users can register for an account.



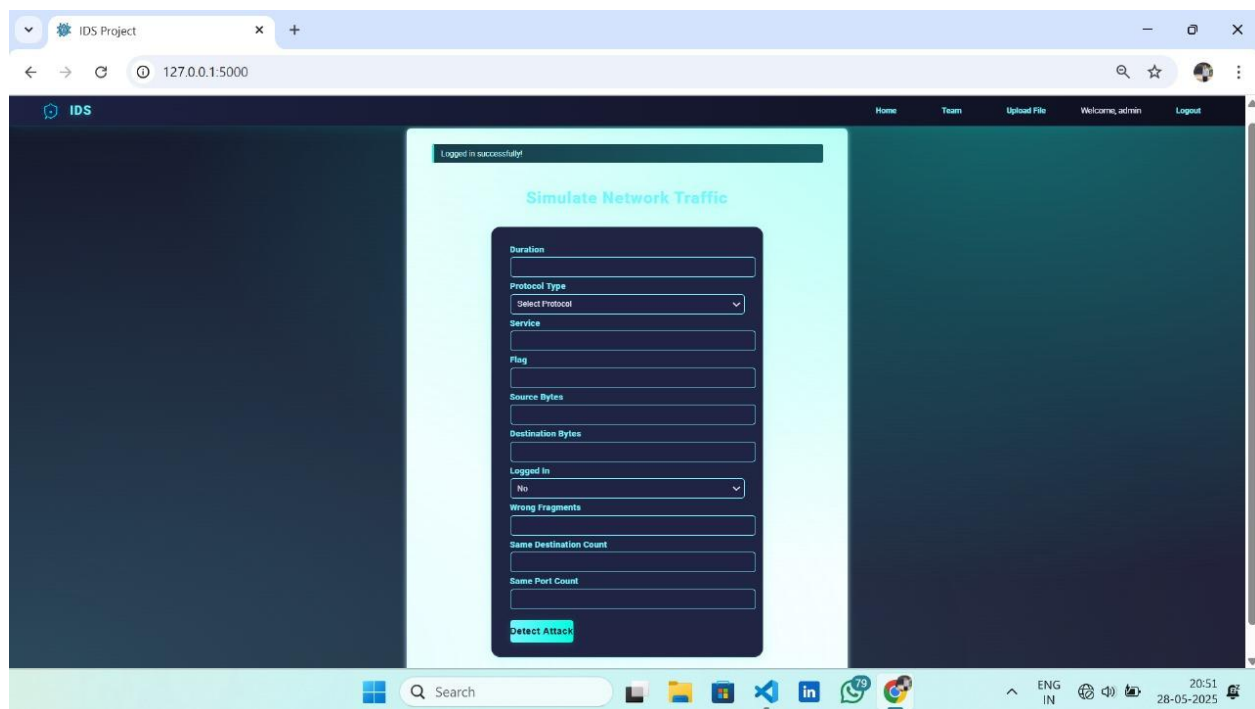
4.2 Uploading a File

The user selects a .txt file for compression. Once the file is uploaded, it is read and processed by



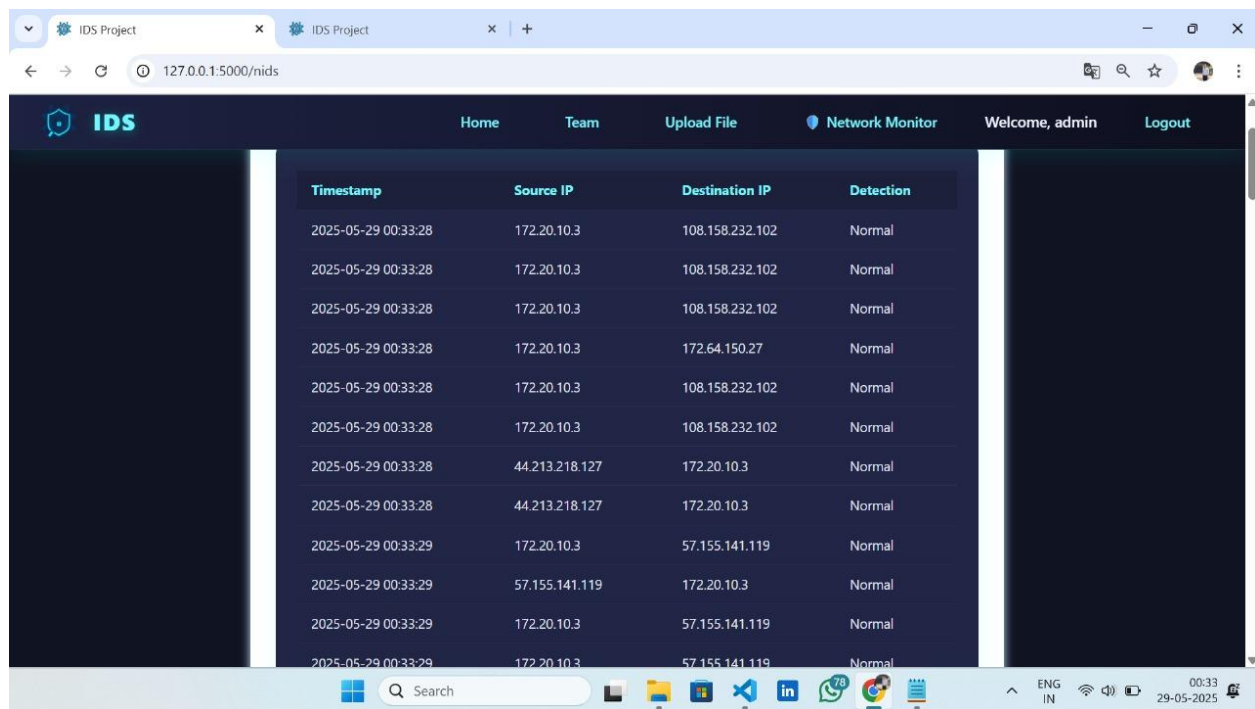
4.3 Network Traffic Simulation

Simulate Network Traffic: After logging in, the user is presented with a form to "Simulate Network Traffic." This form allows the user to input various network parameters such as:



File Scan Queue: After uploading a file, it enters a "File Scan Queue (FCFS Order)" which implies a First-Come, First-Served processing mechanism. The table shows:

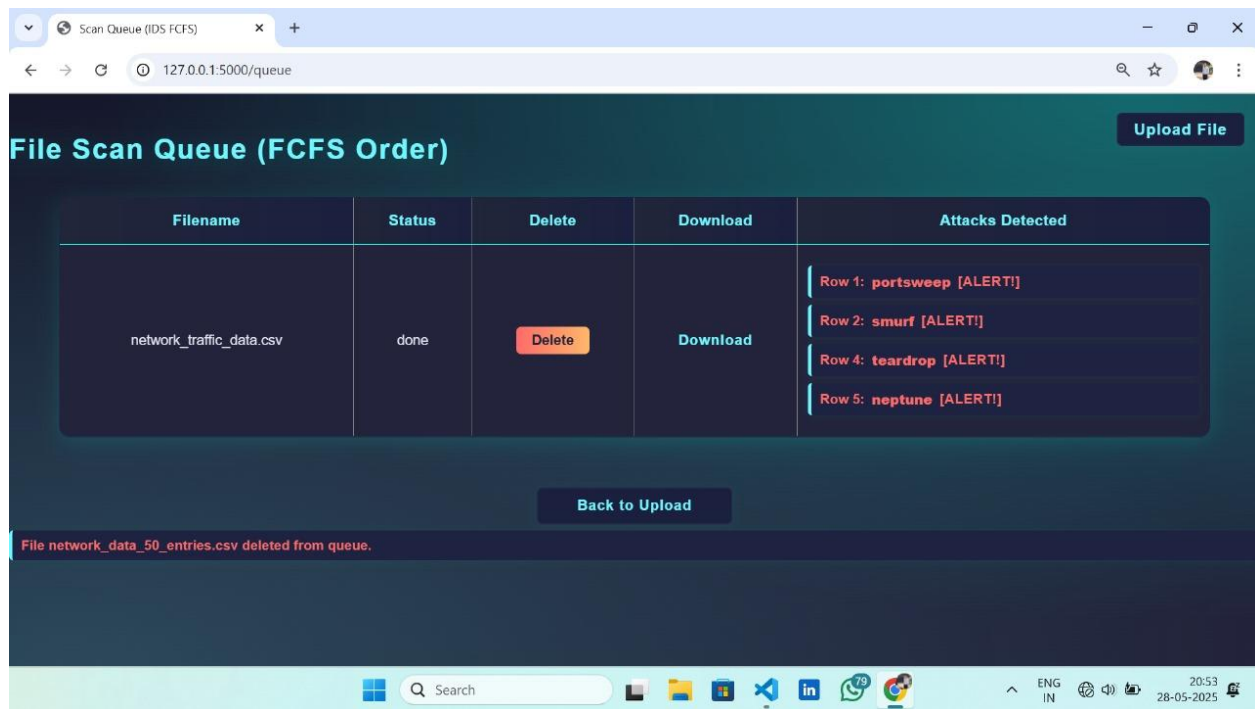
- Filename
- Status (e.g., "done")
- Delete option
- Download option
- **Attacks Detected:** This crucial column lists the types of attacks detected within the uploaded file, along with an "[ALERT!]" tag (e.g., "portsweep," "smurf," "teardrop," "neptune").



The screenshot shows a web browser window with two tabs labeled "IDS Project". The address bar displays "127.0.0.1:5000/nids". The web application has a dark blue header with the "IDS" logo and navigation links: "Home", "Team", "Upload File", "Network Monitor", "Welcome, admin", and "Logout". The main content area features a table with the following data:

Timestamp	Source IP	Destination IP	Detection
2025-05-29 00:33:28	172.20.10.3	108.158.232.102	Normal
2025-05-29 00:33:28	172.20.10.3	108.158.232.102	Normal
2025-05-29 00:33:28	172.20.10.3	108.158.232.102	Normal
2025-05-29 00:33:28	172.20.10.3	172.64.150.27	Normal
2025-05-29 00:33:28	172.20.10.3	108.158.232.102	Normal
2025-05-29 00:33:28	172.20.10.3	108.158.232.102	Normal
2025-05-29 00:33:28	44.213.218.127	172.20.10.3	Normal
2025-05-29 00:33:28	44.213.218.127	172.20.10.3	Normal
2025-05-29 00:33:29	172.20.10.3	57.155.141.119	Normal
2025-05-29 00:33:29	57.155.141.119	172.20.10.3	Normal
2025-05-29 00:33:29	172.20.10.3	57.155.141.119	Normal
2025-05-29 00:33:29	172.20.10.3	57.155.141.119	Normal

The Windows taskbar at the bottom shows the search bar, several application icons, and system status information including "ENG IN", signal strength, and the date/time "00:33 29-05-2025".



Summary

Each code module in this project is built with a clear security-focused objective. The database models ensure secure data management and user authentication. The machine learning pipeline provides accurate and efficient threat detection capabilities. The evaluation module ensures model reliability and performance monitoring. The Flask application integrates all components into a cohesive, user-friendly intrusion detection system that enhances network security through intelligent automation and real-time monitoring capabilities.

CHAPTER 5: LIMITATIONS

Dataset Dependency:

- The system relies on pre-existing datasets (NSL-KDD, CICIDS2017) which may not reflect the latest attack patterns and emerging threats
- Limited to the attack types present in training data, potentially missing zero-day exploits or novel attack vectors.

Real-time Processing Constraints:

- Current implementation may experience latency issues when processing large volumes of network traffic in real-time
- Limited scalability for enterprise-level network monitoring with high-speed data streams.

Model Performance:

- Machine learning models may produce false positives, leading to alert fatigue for security administrators
- Detection accuracy depends heavily on the quality and representativeness of training data
- Models require periodic retraining to maintain effectiveness against evolving threats

Storage Constraints:

- SQLite database may become a bottleneck for large-scale deployments.
- Limited historical data retention capabilities due to storage constraints.

Network Coverage:

- System primarily focuses on network-level intrusions and may miss host-based attacks
- Limited support for encrypted traffic analysis

Basic Visualization:

- Limited advanced data visualization and analytics dashboards.
- Minimal customization options for different user roles and preferences.

Alert Management:

- Basic alert system without advanced filtering, prioritization, or correlation capabilities
- No integration with external security information and event management (SIEM) systems

CHAPTER 6: FUTURE ENHANCEMENTS

6.1 Advanced Machine Learning Integration

Deep Learning Implementation:

- Integration of neural networks and deep learning models for improved pattern recognition
- Implementation of recurrent neural networks (RNNs) for sequential attack pattern detection
- Exploration of ensemble methods combining multiple algorithms for enhanced accuracy

Adaptive Learning:

- Development of online learning capabilities to adapt to new attack patterns automatically
- Implementation of reinforcement learning for dynamic threshold adjustment
- Integration of federated learning for collaborative threat intelligence

6.2 Real-time Processing Enhancements

Stream Processing:

- Integration with Apache Kafka or similar streaming platforms for real-time data processing
- Implementation of sliding window algorithms for continuous monitoring
- Development of edge computing capabilities for distributed deployment

Performance Optimization:

- Multi-threading and parallel processing implementation for improved throughput
- GPU acceleration for machine learning computations
- Caching mechanisms for frequently accessed data and models

Threat Intelligence Integration:

- Integration with external threat intelligence feeds and databases
- Implementation of indicator of compromise (IoC) matching
- Development of threat hunting capabilities with advanced search and correlation

Advanced Dashboard:

- Implementation of interactive dashboards with real-time visualizations
- Development of customizable widgets and reporting capabilities
- Integration of geographic mapping for attack source visualization

Mobile Application:

- Development of mobile applications for remote monitoring and alert management
- Implementation of push notifications for critical security events
- Mobile-responsive design optimization

CHAPTER 7: CONCLUSIONS

The Python-based Intrusion Detection System represents a significant contribution to the field of cybersecurity by demonstrating the practical application of machine learning techniques in network security monitoring. The project successfully addresses the critical need for intelligent, adaptive security solutions capable of detecting evolving cyber threats in real-time.

Project Impact:

Educational-Value:

The system serves as an excellent educational tool for understanding the intersection of machine learning and cybersecurity, providing **hands-on experience with industry-standard libraries and frameworks.**

Practical-Applications:

The IDS offers immediate deployment potential for small to medium-scale network environments, providing cost-effective security monitoring without requiring extensive infrastructure investments.

Research-Foundation:

The modular architecture and comprehensive implementation provide a solid foundation for future research in areas such as deep learning integration, behavioral analysis, and advanced threat intelligence.

Technical Excellence:

The project demonstrates proficiency in multiple technical domains including machine learning algorithm implementation, web application development, database design, and user interface creation. The integration of these components into a cohesive, functional system showcases strong software engineering principles and cybersecurity awareness.

Future Potential:

The system's modular design and extensible architecture position it well for future enhancements including cloud deployment, enterprise-scale integration, and advanced analytics capabilities. The foundation established in this project supports continued development toward more sophisticated threat detection and response mechanisms.

REFERENCES

1. Denning, D. E. (1987). "An Intrusion-Detection Model." *IEEE Transactions on Software Engineering*, 13(2), 222-232.
2. Anderson, J. P. (1980). "Computer Security Threat Monitoring and Surveillance." Technical Report, James P. Anderson Company, Fort Washington, Pennsylvania.
3. Tavallae, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). "A detailed analysis of the KDD CUP 99 data set." *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6.
4. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *4th International Conference on Information Systems Security and Privacy (ICISSP)*, 108-116.
5. Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, 12, 2825-2830.
6. Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, 2nd Edition.
7. Buczak, A. L., & Guven, E. (2016). "A survey of data mining and machine learning methods for cyber security intrusion detection." *IEEE Communications Surveys & Tutorials*, 18(2), 1153-1176.
8. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). "Survey of intrusion detection systems: techniques, datasets and challenges." *Cybersecurity*, 2(1), 1-22.
9. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). "Network intrusion detection system: A systematic study of machine learning and deep learning approaches." *Transactions on Emerging Telecommunications Technologies*, 32(1), e4150.