Intrusion Detection and Prevention System (IDPS) with Blockchain and ML Enhancements

Submitted by

Aashish Singh
Roll No: 229301585

Under the Guidance of

Dr. Prashant Vats

Department of Computer Science and Engineering
Manipal University Jaipur

February 2025

CERTIFICATE

This is to certify that the project entitled **Intrusion Detection and Prevention System (IDPS) with Blockchain and ML Enhancements** is an authentic work carried out as a Minor Project for the course CS3270 in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering by Aashish Singh (Roll No: 229301585) during the academic semester VI of the year 2024–2025.

Place: _____ Date: _____

Guide Name: _____
Signature (with date): _____

Acknowledgement

I express my sincere gratitude to my minor project guide, Dr. Prashant Vats, for his invaluable guidance, unwavering support, and insightful suggestions throughout this project. I also extend my heartfelt thanks to all the faculty members and staff of Manipal University Jaipur, whose constructive feedback and encouragement have been instrumental in the successful completion of this project. Special thanks to my classmates for their collaboration and support.

Abstract

This report presents the design and implementation of an Intrusion Detection and Prevention System (IDPS) enhanced with blockchain and machine learning capabilities. The system is developed as a comprehensive web-based application that integrates secure JWT-based authentication, real-time intrusion monitoring, and detailed event logging (including geolocation data). The IDPS detects and prevents unauthorized access by analyzing network traffic and recording security events such as failed login attempts and suspicious activities. Future enhancements include blockchain integration for tamper-proof logging and machine learning for predictive threat analysis, which will further strengthen the system's security posture.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective of the Project

The primary objective of this project is to develop a robust Intrusion Detection and Prevention System (IDPS) that improves network security by detecting unauthorized access and malicious activities. The system integrates secure user authentication, real-time monitoring, and detailed logging of security events with geolocation information. Future work will incorporate blockchain technology for immutable logging and machine learning for enhanced threat prediction and anomaly detection.

1.2 Brief Description of the Project

The IDPS is implemented as a web-based application that continuously monitors network traffic and logs intrusion events such as failed login attempts and suspicious behavior. It leverages JWT-based authentication, MongoDB for data storage, and geolocation tracking to pinpoint the origin of threats. Planned enhancements include blockchain integration to provide an immutable audit trail and machine learning algorithms to refine threat detection.

1.3 Technology Used

1.3.1 Hardware Requirements

- Processor: Intel Core i5 or higher
- RAM: Minimum 8 GB
- Storage: Minimum 256 GB SSD or 500 GB HDD
- Network: Reliable Internet connectivity

1.3.2 Software Requirements

- Operating System: Windows 10/11, Linux (Ubuntu), or macOS
- Front-End: React with Vite

- Back-End: Node.js with Express
- Database: MongoDB
- Future Integrations: Blockchain (e.g., Ethereum) and Machine Learning libraries (e.g., TensorFlow, scikit-learn)

Chapter 2

Design Description

2.1 Flow Chart

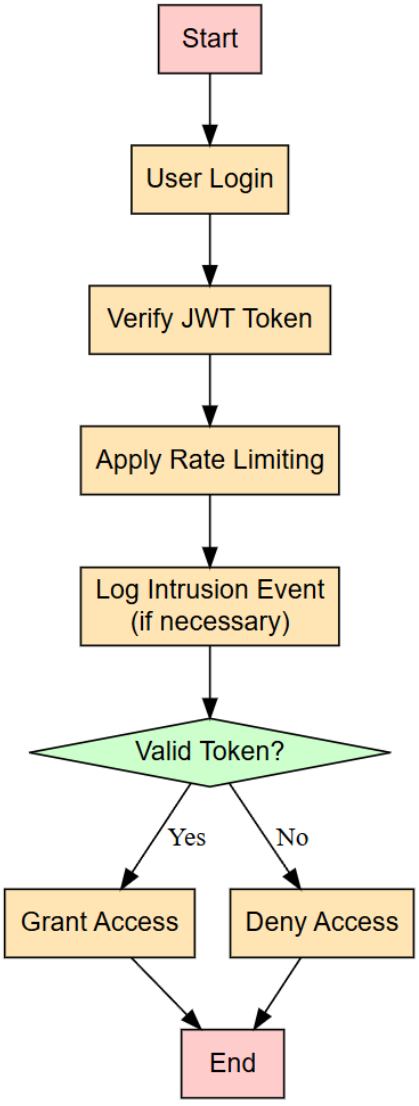

Figure 2.1: System Flow Chart

2.2 Data Flow Diagrams (DFDs)

2.2.1 Level 0 DFD

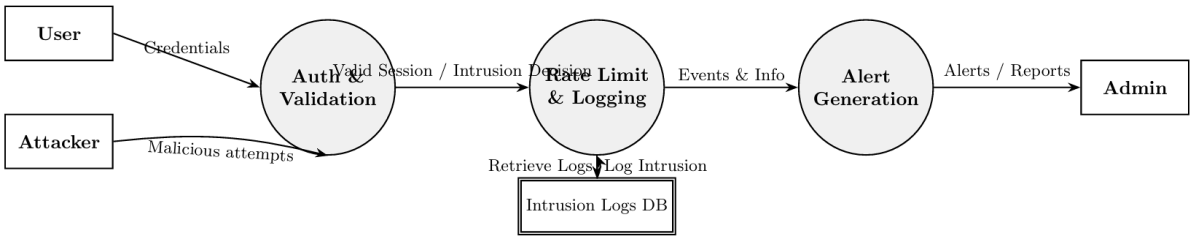

Figure 2.2: Level 0 Data Flow Diagram

2.2.2 Level 1 DFD

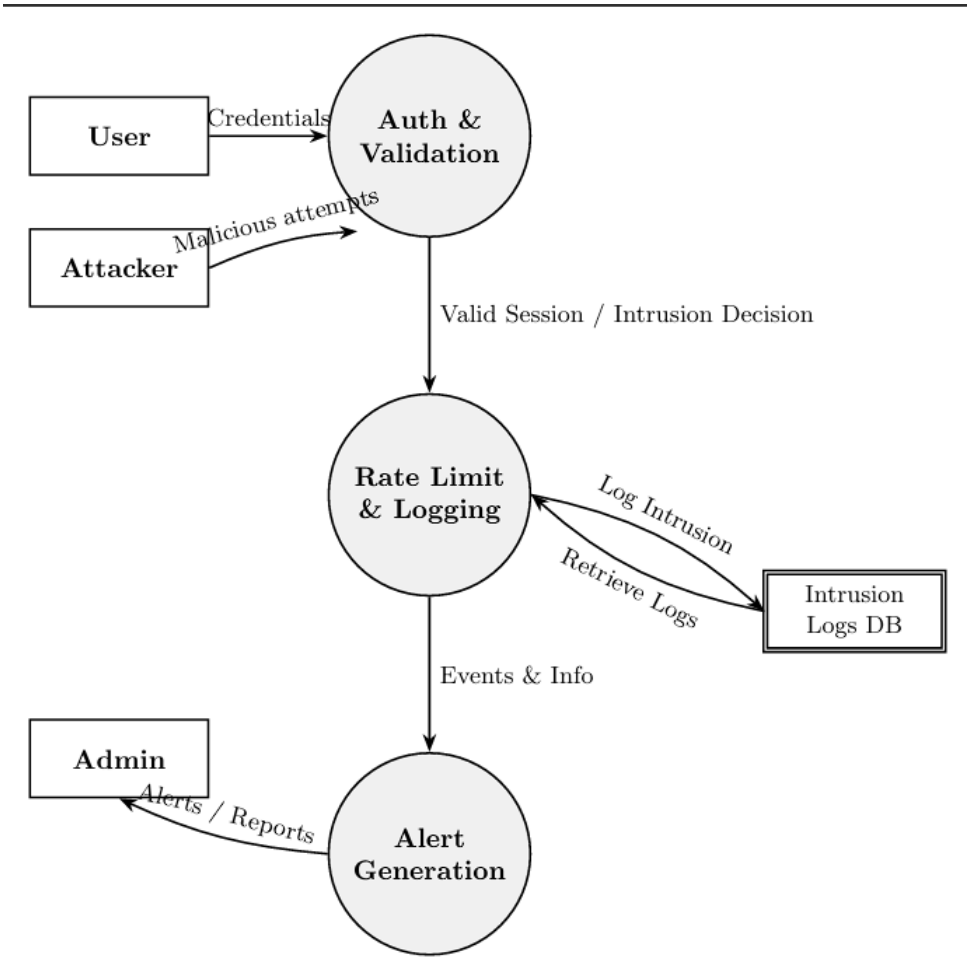

Figure 2.3: Level 1 Data Flow Diagram

2.3 Entity Relationship Diagram (ER Diagram)

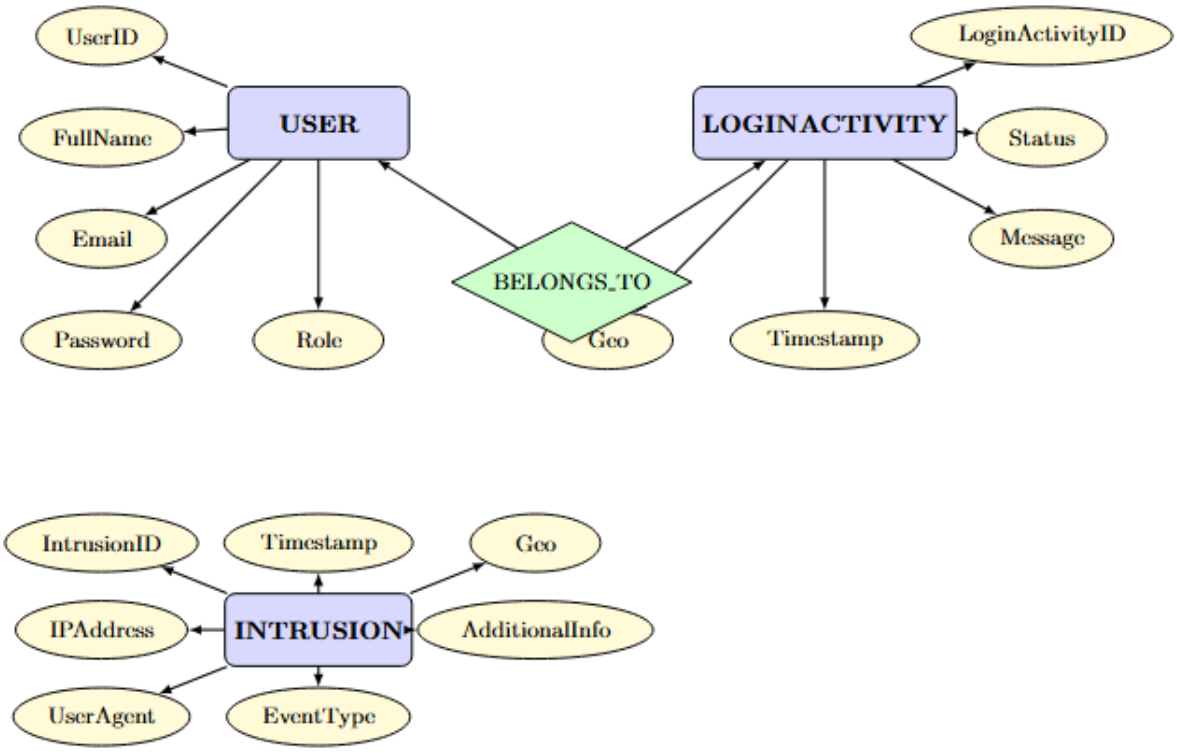

Figure 2.4: Entity Relationship Diagram

2.4 Class Diagram

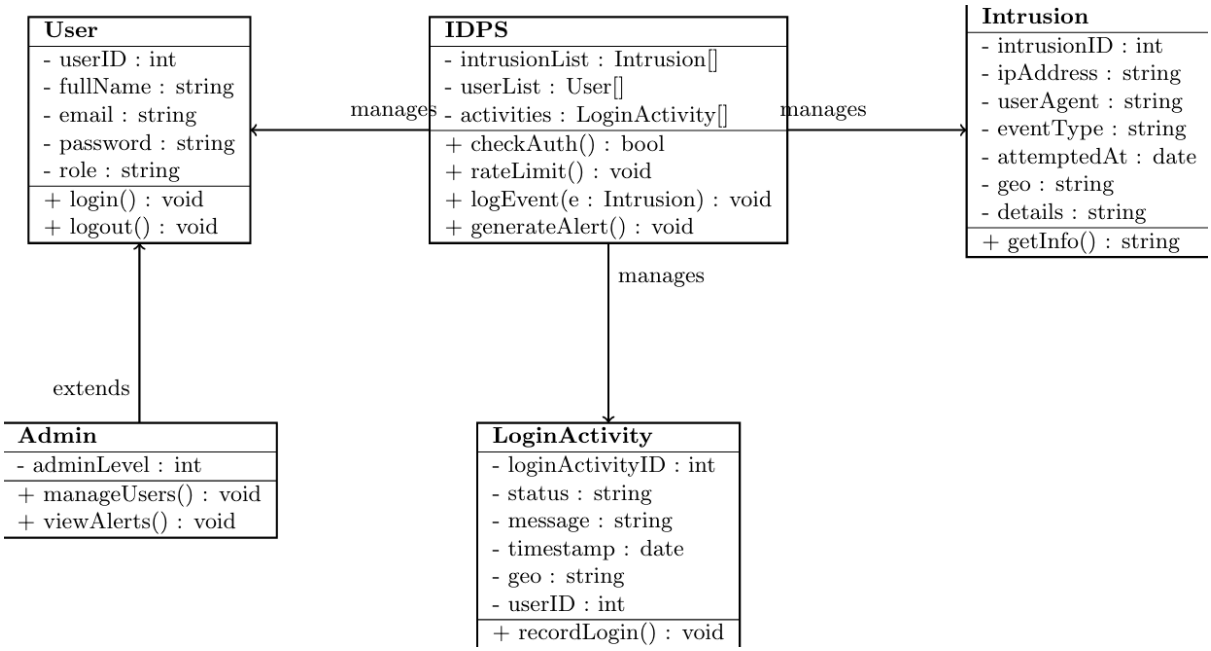

Figure 2.5: Class Diagram of the System

Chapter 3

Project Description

3.1 Database Design and Table Descriptions

The system uses MongoDB as a NoSQL database to store critical data. Key collections include:

- **User Collection:** Stores user information such as full name, email, hashed password, and role (admin or user).
- **Intrusion Collection:** Records security events (e.g., failed login attempts, brute-force attacks) with details including IP address, user agent, event type, additional information, timestamp, and geolocation data.
- **LoginActivity Collection:** Logs each login attempt with details such as the user reference (if applicable), IP address, user agent, status (success/failure), descriptive message, timestamp, and geolocation data.

3.1.1 Table Descriptions

1. **User Collection:**
   - **Fields:** `_id, fullName, email, password, role, ...`
2. **Intrusion Collection:**
   - **Fields:** `_id, ipAddress, userAgent, eventType, additionalInfo, attemptedAt, geo (city, region, country, lat, lon)`
3. **LoginActivity Collection:**
   - **Fields:** `_id, user (reference), ipAddress, userAgent, status, message, timestamp, geo`

3.1.2 File/Database Design

The system follows the Model-View-Controller (MVC) architecture:

- **Models:** Mongoose schemas define the data structures for Users, Intrusions, and LoginActivities.

- **Controllers:** Manage business logic such as authentication, logging intrusion events, and recording login attempts.
- **Routes:** Provide API endpoints for authentication (`/api/auth`), intrusion logs (`/api/intrusions`), user activity (`/api/user/activity`), and dashboard metrics (`/api/metrics`).
- **Front-End:** Built using React with Vite; Axios (configured with HTTP-only cookies) is used to securely communicate with the back-end.

Chapter 4

Input/Output Form Design

4.1 User Interface Design

The system offers user-friendly forms and interfaces that ensure seamless interactions:

- **Authentication Forms:** Login and Registration forms with validation and instant feedback.
- **Admin Dashboard:** Interactive interfaces to view intrusion logs, security metrics, and manage users.
- **User Dashboard:** Simplified interfaces for regular users to view their account status and recent activity.

4.2 Form Design Details

4.2.1 Login and Registration Forms

- **Inputs:** Email, password (and full name for registration).
- **Outputs:** Authentication confirmation, error messages, and success notifications.

Table 4.1: Login and Registration Form Fields

Field	Description
Email	User's email address
Password	User's password (secured)
Full Name	User's full name (for registration)

4.2.2 Admin Dashboard Interface

- **Inputs:** Search fields, filters, and control buttons.
- **Outputs:** Tables and charts displaying intrusion alerts and security metrics.

Table 4.2: Admin Dashboard Data Fields

Field	**Description**
IP Address	Source IP of the intrusion
User Agent	Browser/OS details
Event Type	Type of security event
Additional Info	Further event details
Date	Timestamp of the event
Geolocation	City, region, country (and coordinates)

Chapter 5

Testing & Tools Used

5.1 Testing Methodology

The system was rigorously tested using multiple approaches:

- **Unit Testing:** Each module (authentication, intrusion logging, rate limiting) was tested in isolation using frameworks such as Mocha and Chai.
- **Integration Testing:** API endpoints were verified using Postman and Insomnia, ensuring smooth interactions between the front-end and back-end.
- **Functional Testing:** End-to-end testing simulated real user scenarios (both successful and failed logins).
- **Security Testing:** Stress tests simulated rapid login attempts to trigger rate limiting, verifying that the system returns appropriate 429 responses with wait time messages.

5.2 Tools Used

- Postman/Insomnia for API testing.
- Chrome Developer Tools for front-end debugging.
- Mocha & Chai for back-end unit testing.
- Nodemailer for verifying email notifications.
- MongoDB Compass for database visualization.
- Visual Studio Code for development and debugging.

5.3 Test Cases

1. **Authentication Tests:** Valid and invalid credentials, token verification, and error handling.
2. **Rate Limiting Tests:** Simulate multiple rapid login attempts to ensure a 429 response is returned with a wait time message.

3. **Intrusion Logging Tests:** Verify that failed login attempts are accurately logged along with geolocation data.
4. **Dashboard Data Retrieval:** Ensure that admin endpoints return correct intrusion alerts and security metrics, while non-admin access is appropriately forbidden.

Chapter 6

Implementation & Maintenance

6.1 Implementation

The system is developed using a modern web application stack:

- **Back-End:**
  - **Node.js and Express:** The RESTful API is built on Node.js with Express, handling user authentication, intrusion logging, and rate limiting.
  - **MongoDB:** Data is stored in MongoDB using Mongoose schemas for Users, Intrusions, and LoginActivities.
  - **JWT Authentication:** Secure authentication is implemented via JSON Web Tokens (JWT) stored in HTTP-only cookies.
  - **Rate Limiting:** Custom middleware prevents brute-force attacks by enforcing request limits, with dynamic feedback on wait times.
- **Front-End:**
  - **React with Vite:** The front-end is built using React, with Vite for fast build times.
  - **State Management:** Global state is managed using Zustand.
  - **Axios:** Configured to send HTTP-only cookies, Axios ensures secure communication with the back-end.
- **Future Enhancements:**
  - **Blockchain Integration:** Plans include integrating blockchain (e.g., Ethereum) to record immutable hashes of intrusion logs.
  - **Machine Learning (ML):** Future work aims to incorporate ML algorithms to analyze historical data, predict threats, and minimize false positives.

6.2 Maintenance

To ensure the system remains robust and scalable, the following maintenance strategies are implemented:

- A modular MVC architecture simplifies future updates and debugging.

- Comprehensive documentation and inline code comments guide future development.
- Automated unit and integration tests are regularly run to ensure reliability.
- Continuous monitoring and logging of critical security events facilitate proactive maintenance.
- Scalability measures are in place to easily integrate future enhancements, including blockchain and ML modules.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The Intrusion Detection and Prevention System (IDPS) project effectively demonstrates a comprehensive, web-based application for enhancing network security. By integrating JWT-based authentication, real-time intrusion monitoring, detailed event logging with geolocation data, and robust rate limiting, the system lays a strong foundation for managing and mitigating security threats. Its modular design ensures maintainability and scalability, providing a clear pathway for future enhancements.

7.2 Future Work

Potential future enhancements include:

- **Blockchain Integration:** Implement smart contracts (e.g., on Ethereum) to record cryptographic hashes of intrusion logs, ensuring a tamper-proof, immutable audit trail.
- **Machine Learning Enhancements:** Integrate ML algorithms to analyze historical intrusion data, predict potential threats, and reduce false positives, thereby improving overall threat detection.
- **Mobile Application Development:** Develop a mobile app to deliver real-time security alerts and facilitate remote system management.
- **Advanced Analytics and Reporting:** Enhance the dashboard with sophisticated data visualization and real-time reporting features to support data-driven decision-making.
- **Cloud Deployment:** Deploy the system on a cloud platform to improve scalability, availability, and performance across multiple locations.

Chapter 8

Outcome

The project has resulted in a robust Intrusion Detection and Prevention System that:

- Implements secure JWT-based authentication and role-based access control.
- Logs intrusion events and login activities with detailed geolocation data.
- Utilizes rate limiting to protect against brute-force attacks while providing user feedback on remaining login attempts and wait times.
- Offers distinct dashboards for admin and regular users, ensuring that sensitive security metrics are accessible only to authorized personnel.
- Establishes a scalable and modular foundation for future enhancements, including blockchain integration and machine learning for predictive analytics.

Chapter 9

Bibliography

1. M. Cantelon, M. Harter, T.J. Holowaychuk, and N. Rajlich, *Node JS in Action*, Manning, August 2020.
2. Thomas Mark, *React in Action*, Simon and Schuster, September 2018.
3. Duckett Jon, *HTML & CSS: Design and Build Websites*, Wiley, November 2020.
4. Joren Marynissen and E. Demeulemeester, *Literature Review on Multi-Appointment Scheduling Problems in Hospitals*, August 2020.
5. Torsten O. Paulussen, A. Heinzl, and Christian Becker, *Multi-Agent-Based Information Systems for Patient Coordination in Hospitals*.
6. Jyoti R Munavalli, *Real-time Scheduling in Outpatient Clinics*, September 2020.
7. James F. Cox and Lynn Boyd, *How to Manage Red Alert in Emergency and Disaster Unit in the Hospital?*, September 2020.
8. Hend Mohamed, Hani Kandiel, and Sanaa Abd Elmonem Gharib, *Economic Analysis of the Latent Factors Related to the Nursing Shortage*, September 2020.
9. Millar Ross, Iain Snelling, and Hilary Brown, *Lyberate the NHS: Orders of Change?*, 2011.
10. John Thomas Graham and M. D. Glas, *HEALTHPLIX Spot Appointments*, Development, vol. 5:10.
11. Fatma Poni Mardiah and Mursyid Hasan Basri, *The Analysis of Appointment System to Reduce Outpatient Waiting Time at Indonesia's Public Hospital*, Human Resource Management Research, vol. 3, no. 1, 2013.
12. Chauhan Roma and Amit Kumar, *Practo Technologies: The Online Way of Life!*, Emerald Emerging Markets Case Studies, 2013.
13. S.V. Patil and D.B. Kulkarni, *Graph Partitioning Using Heuristic Kernighan-Lin Algorithm for Parallel Computing*, Advances in Intelligent Systems and Computing, Springer, 2021.

Appendix A

Blockchain Integration Details

This appendix outlines the proposed blockchain integration for the IDPS. The goal is to provide an immutable, tamper-proof log of intrusion events by storing cryptographic hashes on a blockchain network.

A.1 Overview

Blockchain integration will involve:

- Generating a SHA-256 hash for each intrusion log.
- Recording the hash, along with the timestamp and IP address, on a deployed smart contract (e.g., on Ethereum).
- Utilizing the blockchain as an audit trail that can be independently verified.

A.2 Smart Contract Design

The smart contract, written in Solidity, will include:

- A function to log new intrusion event hashes.
- An event to emit details such as the reporter, event hash, and timestamp.

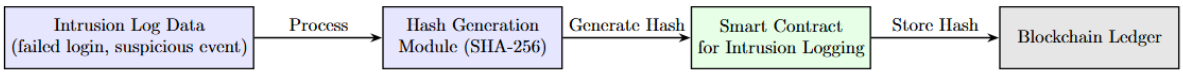

Figure A.1: Smart Contract Architecture for Intrusion Logging

Appendix B

Machine Learning Integration (Future Work)

This appendix describes the potential integration of machine learning techniques to enhance threat detection and predictive analytics in the IDPS.

B.1 Overview

The ML component aims to:

- Analyze historical intrusion data to identify patterns and anomalies.
- Predict potential security breaches in real-time.
- Reduce false positives by continuously learning from new data.

B.2 Proposed Workflow

1. **Data Collection:** Aggregate historical intrusion and login activity logs from the database.
2. **Data Preprocessing:** Clean and normalize the data, including converting geolocation information to numerical features.
3. **Model Training:** Utilize supervised learning algorithms (e.g., Random Forest, SVM) to classify and predict intrusion events.
4. **Deployment:** Integrate the trained model into the back-end to provide real-time threat predictions and alerts.

B.3 Tools and Libraries

Potential libraries for ML integration include:

- **Python:** scikit-learn, TensorFlow, or PyTorch.
- **Node.js:** TensorFlow.js for integrating ML capabilities directly in the back-end.