# A black and red text Description automatically generated

**PROJECT AND TEAM INFORMATION**

## **Project Title**

|  |
| --- |
| Intrusion Detection System using Packet Tracing |

## **Student/Team Information**

|  |  |
| --- | --- |
| *Team Name:*  *Team #* | *Parashuram*  *CYBER-IV-T087* |
| **Team member 1 (Team Lead)** | *Gupta, Krish – 230211625*  [*krishgupta.udh@gmail.com*](mailto:krishgupta.udh@gmail.com) |
| **Team member 2** | *Nauriyal, Shivam – 23021725*  [*shivamnauriyal1224@gmail.com*](mailto:shivamnauriyal1224@gmail.com) |
| **Team member 3** | *Srivastava, Bhaavya – 230221050*  [*srivastava.bhaavya02@gmail.com*](mailto:srivastava.bhaavya02@gmail.com) |
| **Team member 4** | *Bisht, Ashish – 23021782*  [*ashishbisht242005@gmail.com*](mailto:ashishbisht242005@gmail.com) |

**PROJECT PROGRESS DESCRIPTION (35 pts)**

## **Project Abstract**

|  |
| --- |
| This project focuses on developing a lightweight and efficient Intrusion Detection System (IDS) based on real-time packet tracing techniques. In the current landscape of cybersecurity threats, networks are increasingly vulnerable to sophisticated attacks such as port scans, SYN floods, and unauthorized access attempts. Traditional IDS tools like Snort or Suricata, while powerful, often demand significant system resources and present steep learning curves, making them less accessible for smaller institutions or resource-constrained environments.  Our system aims to bridge this gap by offering a streamlined, user-friendly IDS that can operate effectively on low-power or personal devices. It captures live network traffic using tools such as **Wireshark** and **tcpdump**, and analyzes packets at the header level to detect anomalies and malicious patterns. The system integrates a rule-based detection engine for known threats and explores the use of lightweight machine learning models to identify unknown or evolving attack patterns.  By focusing on simplicity, portability, and modular design, the proposed IDS is particularly suitable for educational use, small-scale enterprises, and personal cybersecurity monitoring. The final delivery will be a deployable software package featuring real-time alerts, logging, and an intuitive user interface, along with clear documentation to facilitate easy setup and customization. |

## **Updated Project Approach**

|  |
| --- |
| The system is built using a modular architecture that separates packet capturing, backend processing, and frontend visualization. This design ensures maintainability, scalability, and clarity in functionality.  1. **Packet Capturing (Python - Scapy)**: The backend uses Python with the Scapy library to sniff live network packets. It monitors the network interface in real-time and parses packet headers to extract key features like source/destination IPs, ports, and protocol flags. Detection logic is embedded to identify anomalies such as SYN floods or port scans based on defined patterns and packet frequency.  2. **Backend API (Node.js with Express)**: The backend component serves as middleware, developed using Node.js and Express. It acts as an API layer that receives threat data from the Python engine via HTTP requests or sockets and provides structured JSON responses to the frontend. This decouples the detection logic from the user interface and ensures platform independence.  3. **Frontend (React.js)**: The frontend interface is developed using React.js. It provides a dashboard for real-time monitoring, allowing users to view alerts, packet logs, and threat summaries. React's component-based structure ensures a responsive and interactive user experience.  4. **Communication and Protocols**: Communication between the packet engine and Node.js backend occurs over HTTP or WebSocket. The backend exposes RESTful APIs that the frontend consumes to fetch logs and status updates. Cross-Origin Resource Sharing (CORS) is configured to enable secure interaction between domains.  5. **Libraries & Tools**:  - Python: Scapy (packet sniffing), JSON (data formatting)  - Node.js: Express, body-parser  - React: Axios (API calls), Bootstrap (UI styling)  This architecture ensures a balance between functionality, responsiveness, and modularity. It supports extensibility for future integration of machine learning models and additional network protocols. |

## 

**System Architecture (High Level Diagram)**

|  |
| --- |
|  |

## **Tasks Completed**

|  |  |
| --- | --- |
| **Task Completed** | **Team Member** |
| Developed a packet sniffing engine using Python and Scapy to capture and analyze real-time network traffic. | Shivam Nauriyal |
| Implemented a middleware API using Node.js and Express to handle communication between the backend and frontend. | Krish Gupta |
| Designed and built a responsive dashboard using React.js to display live alerts and network activity logs. | Ashish Bisht |
| Integrated the packet engine, API middleware, and frontend dashboard into a single functional system. | Krish Gupta |
| Conducted functional testing of detection logic and ensured end-to-end data flow between modules. | Shivam Nauriyal |
| Created initial documentation for installation, usage, and configuration of the IDS system. | Bhaavya Srivastava |

## **Challenges/Roadblocks**

|  |
| --- |
| Throughout the development of the Intrusion Detection System, our team encountered several technical and integration challenges. One major issue was **real-time packet processing**, which required optimizing the performance of the packet sniffing engine to handle continuous network traffic without causing system lags or delays. We resolved this by implementing efficient buffer management and reducing unnecessary processing overhead.  Another significant challenge was ensuring **cross-platform compatibility**. Packet capturing libraries behave differently on various operating systems, especially between Linux and Windows. We addressed this by testing the system across platforms and modifying configurations to maintain consistent behavior.  Integrating **multiple technologies**, namely Python for backend processing, Node.js for the API, and React for the frontend, introduced synchronization and data formatting challenges. Careful planning of communication protocols and consistent data structuring helped us achieve smooth integration between components.  Lastly, designing a **user-friendly and responsive interface** was initially challenging due to the complexity of displaying real-time data dynamically. Through iterative testing and UI refinement, we developed a dashboard that is both functional and accessible to users with limited technical backgrounds.  These challenges helped us strengthen the system architecture and improve its usability and reliability. |

## **Future Scope**

## The Intrusion Detection System (IDS) developed in this project has strong foundational features, but there are several areas where it can be further improved and expanded to enhance performance, usability, and real-world applicability.

1. **Machine Learning Integration**  
   Incorporating lightweight machine learning models can help detect unknown or evolving threats that may not match existing rule sets. This would significantly enhance the system's ability to adapt to new attack patterns.
2. **Enhanced Visualization**  
   The dashboard can be upgraded with interactive visual tools such as traffic graphs, heatmaps, and timeline-based threat tracking. These additions will offer better insights into network behaviour and threat trends over time.
3. **Performance and Scalability**  
   Optimizing packet processing and memory usage will allow the system to operate smoothly under high network loads and during extended monitoring sessions.
4. **Improved Detection Accuracy**  
   Enhancing rule sets and refining feature selection for analysis can help reduce false positives and ensure more accurate detection of actual threats.
5. **Cross-Platform Support**  
   Packaging the system using cross-platform tools like Docker can simplify deployment across Windows, macOS, and Linux systems.
6. **Security Features**  
   Implementing user authentication and access control will make the system more secure and suitable for organizational environments.
7. **Protocol and Rule Expansion**  
   Adding support for more network protocols and dynamic rule updates will increase the system’s flexibility and detection coverage.

These enhancements will make the IDS more powerful, adaptable, and ready for real-world deployment.

## **Project Outcome/Deliverables**

|  |
| --- |
| The outcome of this project is a fully functional and modular Intrusion Detection System (IDS) that utilizes real-time packet tracing techniques to identify and report suspicious network activity. The system captures live traffic using Python and Scapy, analyzes packet-level features such as IP addresses, ports, and protocols, and detects anomalies using rule-based logic, with future integration of machine learning capabilities for improved threat identification.  The IDS follows a modular architecture, comprising three main components: a packet sniffing engine (Python), a middleware API (Node.js), and a front-end dashboard (React.js). Together, these components provide seamless end-to-end functionality and efficient communication.  Key deliverables include a lightweight detection engine suitable for low-resource systems, a real-time alerting and logging interface, and a user-friendly dashboard for monitoring threats. The system is easy to deploy, with minimal dependencies, and comes with comprehensive documentation for installation, configuration, and usage.  The project is tested for accuracy, responsiveness, and usability, making it ideal for academic use, small enterprises, and personal cybersecurity applications. |

# Progress Overview

|  |
| --- |
| The project is progressing well, with key components fully implemented and integrated. The packet sniffing engine, developed using Python and Scapy, captures and analyzes live network traffic effectively. A rule-based detection system has been implemented to identify known threats, including SYN floods and port scans. The middleware, built using Node.js and Express, serves as a communication layer between the detection engine and the front end. The user interface, designed with React.js, offers a responsive dashboard for real-time monitoring of alerts, packet logs, and network activity.  System integration across all modules—backend, middleware, and frontend—has been completed and tested for real-time performance. The application runs smoothly under standard network loads and provides accurate threat detection with minimal false positives.  Pending tasks include integrating lightweight machine learning models to improve detection of unknown or evolving threats, enhancing the dashboard with advanced visualizations, optimizing performance for high-traffic environments, and finalizing detailed documentation for setup and use. Despite these pending elements, the project is on schedule and aligned with the initial goals of building a lightweight, modular, and user-friendly intrusion detection system. |

# Codebase Information

|  |
| --- |
| Repository Link: <https://github.com/GitShivamNauriyal/Real-Time-Intrusion-Detection-System>Branch: master  Key Directories: - engine/: Python-based packet engine - middleware/: Node.js API - client/: React frontend |

## **Testing and Validation Status**

|  |  |  |
| --- | --- | --- |
| Test Type | Status (Pass/Fail) | Notes |
| Packet capturing and analysis | Pass | Successfully captured live traffic using Scapy. Verified against Wireshark logs. |
| Threat detection accuracy | Pass | Accurately detected SYN floods and port scans during simulated attacks. |
| Frontend usability | Pass | Dashboard is responsive and intuitive; tested across Chrome, Firefox. |
| System under load | Ongoing | Currently optimizing for performance under high traffic. Minor lag observed. |

# Deliverables Progress

|  |
| --- |
| * The **packet sniffing engine** has been fully implemented using Python and Scapy. It effectively captures live network traffic and extracts relevant features for analysis. * The **middleware API**, developed in Node.js using Express, is complete and functions as the communication layer between the packet engine and the front-end interface. * The **front-end dashboard**, built with React.js, has been successfully developed. It provides real-time visual alerts, logs, and status updates in a clean and interactive format. * **System integration** across all three layers—packet capture, API middleware, and frontend—has been completed. Components communicate seamlessly, ensuring smooth data flow and real-time threat reporting. * The **documentation**, including installation steps, system architecture, and usage instructions, is currently in progress and will be finalized before deployment. * Development of **advanced detection features**, such as machine learning-based anomaly detection and additional threat signatures, is ongoing. Initial prototypes are being tested and integrated gradually into the system. |