SAPI-G

Secure API Gateway with AI Threat Detection

Phase 3: System Architecture & Secure Design

Secure Software Development

CY-321

Project Team:

Umar Tariq (2022604)
M Zeeshan (2022644)
Ayela Israr (2022130)
Ahmad Amjad (2022063)

System Architecture & Secure Design

1. System Architecture

The Secure API Gateway with AI Threat Detection consists of multiple interconnected components, each serving a critical role in securing API interactions and detecting threats dynamically.

1.1 High-Level Architecture

The architecture is divided into the following layers:

1. **Client Layer**:
    - Users, applications, or external services interacting with the API gateway.
    - Frontend application (React.js) consuming API services.

2. **API Gateway (Node.js)**:
    - Handles authentication, authorization, rate limiting, and request validation.
    - Logs and forwards API requests to the respective microservices.

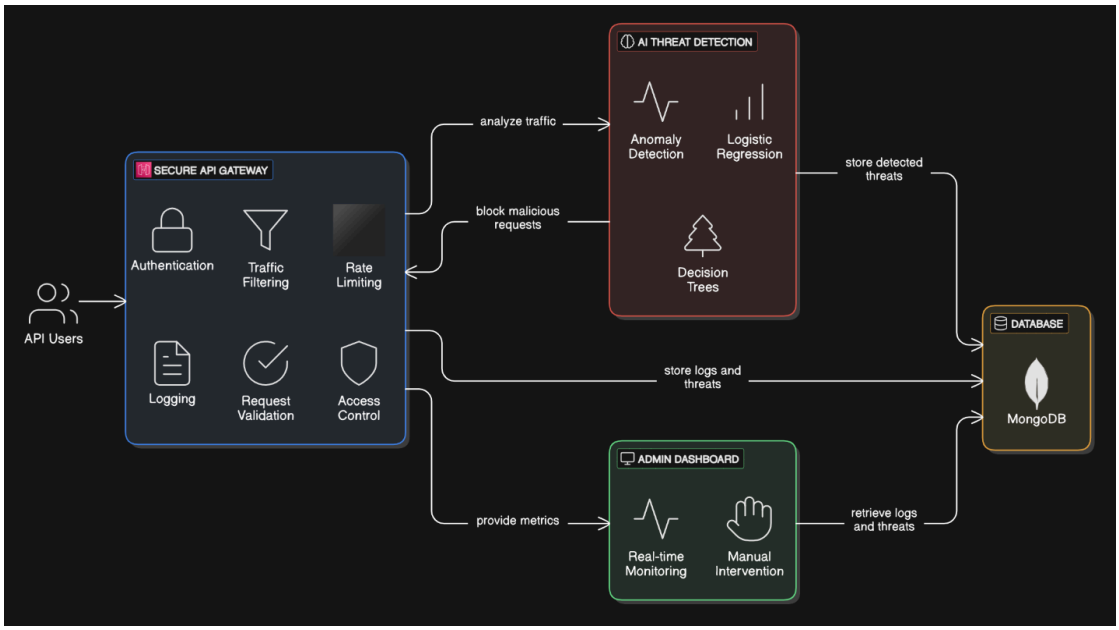
3. **Threat Detection Engine (AI/ML-based - Python)**:
    - Monitors API traffic for anomaly detection.
    - Uses Isolation Forest and Logistic Regression models to detect threats.
    - Blocks malicious requests dynamically.

4. **Backend Services**:
    - Microservices handling business logic and database operations.
    - Database layer (MongoDB/PostgreSQL) storing API logs, detected threats, and blacklisted IPs.

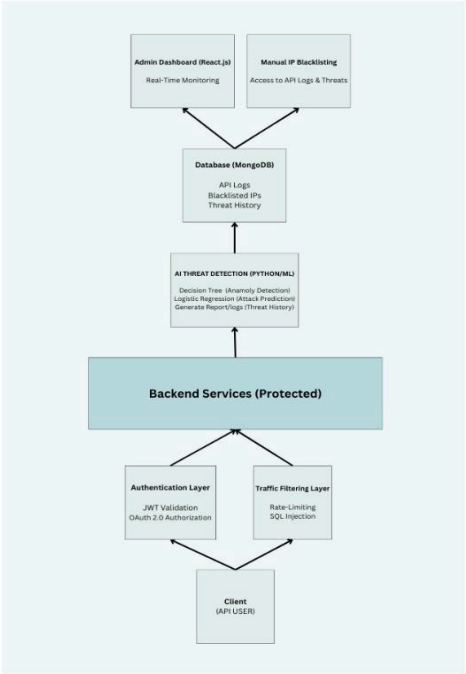
5. **Security Dashboard (React.js)**:
    - Provides real-time monitoring for API traffic and security incidents.
    - Enables administrators to manually blacklist/unblacklist IPs.

6. **Storage & Logging**:
    - Cloud-based storage and logging mechanism (AWS/GCP).
    - Logs API traffic, security events, and ML-detected threats.

1.2 System Architecture Diagram



1.3 Flowchart Diagram



2. Security Controls

To ensure the robustness of the API Gateway against cyber threats, the following security controls are implemented:

2.1 Authentication & Authorization

- **JWT (JSON Web Token) & OAuth 2.0**:
  - JWT ensures secure, stateless authentication.
  - OAuth 2.0 enables secure token-based authorization.
- **API Key Management**:
  - Only registered applications can access APIs using assigned keys.

2.2 Encryption & Secure Communication

- **HTTPS with TLS 1.2+**: All API requests are encrypted using TLS to prevent MITM attacks.
- **Data Encryption**:
  - AES-256 encryption for sensitive data at rest.
  - End-to-end encryption (E2EE) for communication between services.

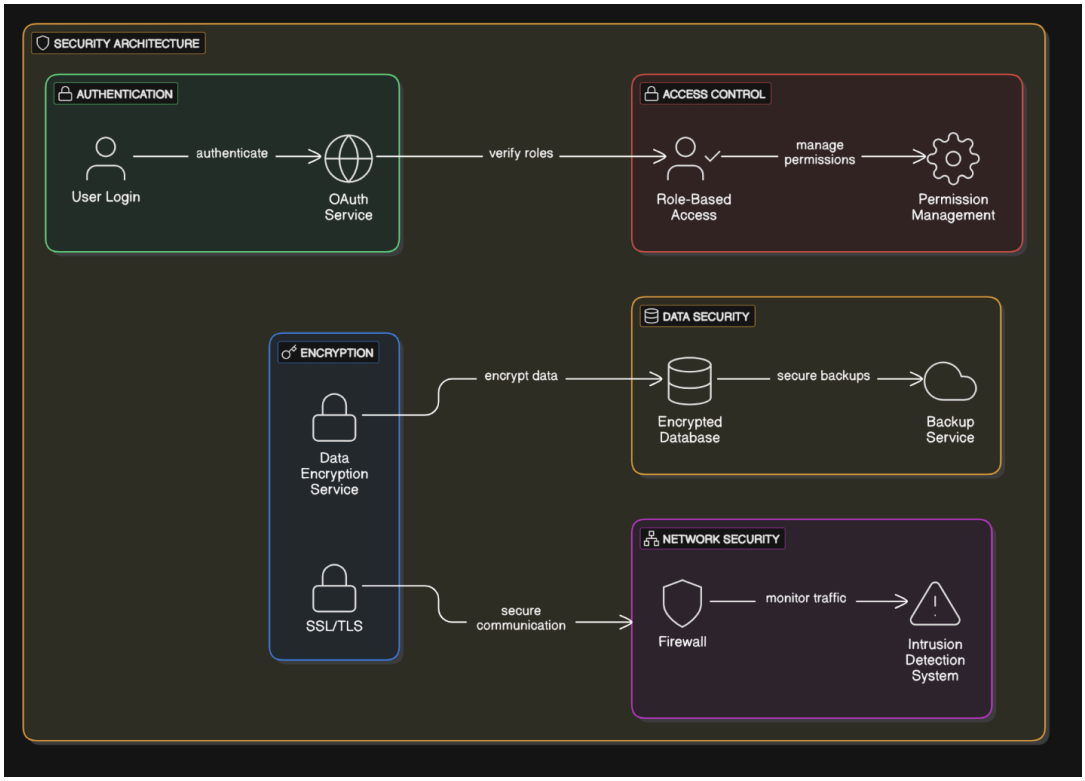
2.3 Access Control & Rate Limiting

- **Role-Based Access Control (RBAC)**:
  - Ensures users and services have the minimum required privileges.
- **Rate Limiting & IP Blacklisting**:
  - Limits API request frequency to prevent DDoS attacks.
  - Blacklists IPs with malicious activity detected by AI models.

2.4 Threat Detection & Logging

- **AI-driven Anomaly Detection**:
  - Identifies and mitigates potential attacks (e.g., SQL injection, XSS, brute force attacks).
- **Real-time API Traffic Monitoring**:
  - Logs every API request for forensic analysis.
  - Alerts security admins on suspicious activity.

2.5 Security Design



3. Security Design Measures

3.1 Secure API Development Practices

- **Input Validation & Sanitization**:
  - Prevents injection attacks by validating and sanitizing user inputs.
- **CORS Policy Enforcement**:
  - Restricts access to APIs from trusted domains only.

3.2 Secure Deployment & Monitoring

- **Containerization with Docker**:
  - Isolates API gateway and backend services for better security.
- **Cloud-Based Security (AWS/GCP)**:
  - Uses cloud-based firewalls and WAF (Web Application Firewall) for additional security layers.
- **Continuous Security Audits & Penetration Testing**:
  - Regularly tests the API Gateway for vulnerabilities and security gaps.

Conclusion

This document outlines the secure design and architecture for the Secure API Gateway with AI Threat Detection. By incorporating AI-driven threat detection, robust authentication, encryption, and access control, this solution ensures proactive protection against evolving cyber threats while maintaining API performance and reliability.