**DATABASE AUDIT TOOL**

**Report**

**21K-4947 Samia Ilyas**

**21K-3199 Khadeejah Mansoor**

**21K-3342 Aarib Azfar**

**Introduction:**

This codebase represents a **Laravel**-based security audit tool with authentication, role-based access control (RBAC), and comprehensive security monitoring capabilities. The application is designed to track and monitor database activities, suspicious queries, and authentication attempts.

**Role Based Access Control (RBAC):**

The application implements a comprehensive RBAC system that controls user access based on assigned roles:

Role Definitions:

* **Admin**: Full system access with all privileges and protected API
* **Auditor**: Access to audit logs and certain monitoring features
* **Viewer**: Limited read-only access to basic system information

Access Control Hierarchy:

1. Unauthenticated users: Access only to public endpoints
2. Authenticated viewers: Basic system access
3. Authenticated auditors: Access to logs and monitoring data
4. Authenticated admins: Complete system control

**Authentication System:**

The application uses a multi-layered authentication system combining Laravel's built-in auth with Sanctum token-based API authentication:

Components:

* Login Endpoint: /api/login route handled by AuthController::login()
* Token Management: The personal\_access\_tokens table stores API authentication tokens
* User Authentication: The User model extends Laravel's Authenticatable class
* Failed Login Recording: Unsuccessful attempts are logged to the failed\_logins table

Authentication Flow:

* User submits credentials to the login endpoint
* If valid, a Sanctum token is generated and returned
* The token is used in subsequent API requests (Bearer token)
* Protected routes verify the token and role before granting access

**Failed Login Monitoring:**

The system implements comprehensive tracking of authentication failures:

Components:

* Failed Login Table: The failed\_logins table records unsuccessful attempts
* Automatic Logging: The /api/failed-login endpoint captures failed login data
* Data Captured: Username (when provided), source IP address, and timestamp
* Visualization: The /failed-logins-page web route displays a formatted view of failed login attempts
* API Access: The /api/failed-logins endpoint provides JSON access to the data

Security Benefits:

* Detecting brute force attacks
* Identifying compromised accounts
* Monitoring suspicious activity patterns
* Supporting security audits and investigations

**Suspicious Query Detection:**

The application includes a database query monitoring system that identifies and logs potentially malicious or problematic SQL queries:

Components:

* Suspicious Query Table: The suspicious\_queries table stores flagged database operations
* Query Metrics: Records query text, execution time, and detection timestamp
* Detection Focus: Particularly monitors for:
* SQL injection patterns (e.g., SLEEP() calls)
* Performance issues (slow-running queries)
* Access pattern anomalies

**System Integration and Architecture:**

The Laravel Audit Tool integrates multiple security components that work together:

1. Authentication → RBAC:

* Authentication verifies user identity
* Successful authentication establishes session/token
* RBAC then determines permitted actions based on role

1. Authentication → Failed Login Monitoring:

* Failed authentication attempts trigger logging
* Pattern analysis can detect attack attempts
* IP addresses from repeated failures can be monitored

1. Database Activity → Suspicious Query Detection:

* Database queries are monitored for patterns
* Execution times are recorded for performance issues
* Suspicious patterns are flagged and logged

1. All Components → Audit Logging:

* Security events from all subsystems feed into audit logs
* Provides a centralized timeline of system activity
* Supports investigation and compliance requirements

**API and Web Interface Integration:**

The system provides both API endpoints for programmatic access and web interfaces for human interaction:

API Features:

* RESTful endpoints for all major functions
* Token-based authentication using Sanctum
* JSON response format for machine consumption
* Role-based endpoint restrictions

Web Interface:

* Failed login visualization page
* User Activity visualization page
* Schema Changes tab
* Query performance tab

Integration: Failed logins accessible via both /api/failed-logins (API) and /failed-logins-page (web)

**Conclusion:**

The application implements three types of database security monitoring:

1. Audit Logs: Tracks system events in the audit\_logs table, including:
   * Login attempts from various hosts
   * Privilege verification checks
   * Database access verification
   * Schema version checks
2. Failed Login Tracking: Records unsuccessful login attempts in the failed\_logins table, capturing:
   * Username (when provided)
   * Source IP address
   * Timestamp
3. Query Monitoring: Captures suspicious database queries in the suspicious\_queries table, focusing on:
   * Potentially harmful queries (e.g., SLEEP() function calls that could indicate SQL injection)
   * Performance problems (queries with long execution times)
   * Query text and execution metrics

All these logs can be retrieved through API endpoints, with audit logs having both public and authenticated access methods.