Project Quorum – AI-Powered Log Analysis for Secure Offline Environments

Project Synopsis

By

Ankit Vishwakarma

Bachelor of Science (Computer Science)

PIONEERING EDUCATION
SINCE 1992

DEPARTMENT OF COMPUTER SCIENCE

SHREE L. R. TIWARI DEGREE COLLEGE OF ARTS, COMMERCE & SCIENCE

(Affiliated with University of Mumbai)

THANE, 401107

MAHARASHTRA

2025-26

1. Introduction

Project Quorum is a novel, self-contained forensic analysis platform specifically designed to solve the challenge of security analysis in air-gapped or offline environments. It functions as a portable, AI-powered "offline security operations center" that runs entirely on an analyst's workstation from a single, easy-to-deploy executable. By seamlessly integrating an embedded, high-performance analytical database and robust on-device machine learning models, Quorum provides advanced, AI-driven threat detection, deep forensic capabilities, and log analysis to environments that have zero external network connectivity or cloud access.

2. Project Category

Offline AI-Powered Cybersecurity / Digital Forensics Tool

3. Objective

The primary objectives of Project Quorum are:

- To design and develop a fully self-contained, portable log analysis system that operates in completely offline (air-gapped) environments.
- To integrate a high-performance, in-process analytical (OLAP) database (DuckDB) to enable complex, high-speed SQL querying on large, locally-stored log files.
- To implement a cross-platform log parsing engine for structuring raw log formats, specifically Windows EVTX and Linux Syslog, for analysis.
- To build an embedded AI engine using TinyML (TensorFlow Lite) and the PyOD library to perform on-device, log-based anomaly detection without any cloud dependency.
- To develop a "Secure Offline Update Protocol" (SOUP) to provide a verifiable and cryptographically-signed mechanism for updating AI models and detection rules in a disconnected setting.
- To create a modern, React-based forensic dashboard that visualizes detected anomalies and intelligently maps them to the MITRE ATT&CK framework for rapid investigation.

4. System Requirements

4.1. Hardware Requirements:

- **CPU:** 64-bit multi-core (recommended for parallel queries).
- **RAM:** 8 GB min, 16 GB+ recommended for large-scale analysis.
- **Storage:** 1 GB for app; sufficient space for log databases.
- **Peripherals:** USB port (for portable deployment).

4.2. Software Requirements:

- **Operating System:** Cross-platform (Windows 10/11, macOS, Linux).
- **Application Stack (Bundled):**
    - **Frontend:** Modern web browser (runs in OS-native WebView).

    - **Backend:** Python 3.9+.
- **Key Libraries:** FastAPI, DuckDB, PyOD, TensorFlow Lite, React, Tauri.

5. Scope of the Project

- The project will focus on the offline, static analysis of historical log files.
- The system will parse, ingest, and index provided Windows EVTX and Linux Syslog files into an embedded DuckDB database.
- The AI engine will be capable of running multiple anomaly detection models from the PyOD library (e.g., KNN, Isolation Forest) on the ingested data.
- The React-based UI will provide a dashboard for data visualization, SQL querying, and viewing AI-detected anomalies.
- The final deliverable will be a single, portable executable.
- A proof-of-concept of the SOUP mechanism will be developed to demonstrate a secure update.

6. Expected Results

- A functional, cross-platform, portable executable that launches a self-contained forensic analysis dashboard on Windows, macOS, and Linux.
- A robust system capable of successfully parsing and ingesting multi-gigabyte structured log files (EVTX, Syslog) into a local DuckDB database for high-speed SQL analysis.
- A successful demonstration of the AI engine identifying anomalies in benchmark log datasets (e.g., LogPai, HDFS, BGL).
- A user-facing dashboard that visually represents detected threats by mapping them to their corresponding TTPs (Tactics, Techniques, and Procedures) on a MITRE ATT&CK matrix.
- A successful test of the Secure Offline Update Protocol (SOUP), where the application correctly accepts a valid, cryptographically-signed model update and rejects an invalid or tampered update file.