LLM-Powered SOC Assistant

A Natural Language Interface for Open-Source Security Monitoring

Progress Report 1

Umesh Kalupathirannehelage – 300389749

Applied Research Project

CSIS4495-071

1. Work Log Table

Student Name: Umesh Kalupathirannehelage

Date	Number of Hours	Description of Work Done
Sept 15, 2025	2	Installed Ubuntu & Kali Virtual Machines on oracle virtual box.
Sept 16, 2025	4	Installed Wazuh (SIEM), Postgre SQL on Ubuntu and Wazuh agent on Kali and verified functionality.
Sept 17, 2025	1	Fixed missing auth.log by enabling rsyslog; tested SSH failed login detection.
Sept 17, 2025	2	Implemented ETL script load alerts from Wazuh into PostgreSQL. Encountered psycopg2 permission errors and resolved by adjusting DB roles.
Sept 18, 2025	2	Set up PostgreSQL schema (logs table with JSONB field); validated successful data ingestion from ETL script.
Sept 19, 2025	2	Installed and configured FastAPI app (main.py); added system prompt for schema-aware SQL generation.
Sept 20, 2025	3	Integrated Ollama (Windows) with FastAPI on Ubuntu VM using bridged networking; tested connectivity via REST API calls
Sept 22, 2025	2	Enhanced SQL normalization logic to fix errors (invalid GROUP BY, misplaced WHERE); successfully queried sshd events.
Sept 23, 2025	2	Researched about moving the VM to cloud as I faced lack of hardware resources.
Sept 25, 2025	2	Deployed an Ubuntu EC2 instance (m7i-flex.large) on AWS with 180 GB gp3 storage; installed Wazuh Manager and PostgreSQL; configured security groups for agent communication
Sept 26, 2025	2	Installed FastAPI and ETL components on the AWS instance; connected ETL to Wazuh alerts.json and validated ingestion into PostgreSQL; tested FastAPI queries against the cloud-hosted DB.

2. Description of Work Done

Sept 15–18, 2025

During this week, I laid the foundation of the SOC Assistant project by preparing the environments and becoming familiar with new tools. I installed Ubuntu and Kali VMs on Oracle VirtualBox and set up Wazuh (SIEM) and PostgreSQL on the Ubuntu machine, along with a Wazuh agent on Kali. Since this was my first experience with Wazuh, I encountered challenges understanding its components and how logs flow from agents to the manager. Another issue was that auth.log was missing on Kali, preventing SSH event collection. After researching, I discovered that rsyslog was disabled by default and re-enabled it, which resolved the problem and allowed SSH failed login attempts to appear. I also implemented an ETL script to parse alerts.json and load alerts into PostgreSQL, which initially failed due to psycopg2 permission errors. This was resolved by correcting database roles and privileges. Finally, I created the PostgreSQL schema with a JSONB field to store raw alerts and validated successful ingestion.

Sept 19–22, 2025

This week I advanced from data ingestion to building the application layer. I installed and configured a FastAPI app (main.py) to query the database using natural language prompts. One of the challenges was ensuring that the schema-aware SQL generation aligned with the Postgres schema, which required carefully defining the system prompt. I also worked on integrating Ollama, running on my Windows host, with FastAPI inside the Ubuntu VM. This required configuring bridged networking and testing REST connectivity across environments, which was successful after troubleshooting network settings. Once the integration worked, I began testing end-to-end queries but encountered errors in SQL generation (such as invalid GROUP BY usage and misplaced WHERE clauses). I improved the SQL normalization logic to automatically rewrite problematic queries, which finally enabled me to retrieve and analyze sshd failure events directly from the assistant.

Sept 23-26, 2025

After building the basic pipeline, I encountered resource constraints with the local VirtualBox environment. Running Wazuh, PostgreSQL, FastAPI, and ETL together caused performance issues, and Ollama could not be hosted reliably due to memory and disk limits. This led me to research cloud migration strategies. I investigated using AWS EC2 instances (e.g., m7i-flex.large with 8 GB RAM and 180 GB gp3 storage) to host Wazuh, PostgreSQL, and the FastAPI backend, while keeping Ollama on my Windows machine. The cloud approach addresses the lack of local hardware resources and provides a stable environment for the SOC Assistant.

During this week, I went beyond research and launched an Ubuntu EC2 instance on AWS. I installed the required software stack, Wazuh Manager, PostgreSQL, and the FastAPI backend with the ETL pipeline. I configured the database schema, set up the ETL script to read alerts.json from Wazuh, and confirmed that alerts were successfully ingested into PostgreSQL. I also tested the FastAPI service to ensure that natural language queries could generate SQL, query the database, and return results. The main challenges this week were evaluating AWS costs (e.g., storage billed 24/7 vs compute only when running), configuring security groups and firewall rules to allow agent communication, and ensuring that the system ran stably in the cloud. These steps validated that the AWS-hosted environment could serve as a reliable platform for continuing the SOC Assistant project.

The files/folders I have checked in the repo are as follows:

Under the implementation folder,

- .env – Environment configuration file containing database credentials and API variables.
- main.py – FastAPI backend application implementing the NL → SQL SOC Assistant API.
- etl_wazuh_to_pg.py – Python ETL script to extract alerts from Wazuh and load them into PostgreSQL.
- create_logs_table.sql – SQL script to create the logs table schema in PostgreSQL, including JSONB field for raw alerts.

Under the documents folder,

- Project_UKa749_Report1.pdf – This progress report document.

All of these files were pushed to the repository at the same time in a single commit after I completed the initial functional baseline of the project (database schema, ETL, and API setup). Future progress reports will reflect more granular and incremental commits as additional features are added and tested.