SOC AI AGENT Report

BY: Mohamad Hassan Ibrahim 6321

Reem Kanaan 6315

ULFG III, Year 4, Computer and Telecommunication Engineering

2024/2025 – Mini Project Course

Presented for: Dr. Mohammad Aoude

1. Project Overview

SOC_AI_Agent is a fully open-source, modular, and Dockerized Security Operations Center (SOC) AI analyst prototype. The platform demonstrates how a modern AI agent, empowered by the latest LLM technology and automated threat intelligence enrichment, can radically accelerate alert triage, reduce analyst workload, and provide step-by-step explainability.

- Domain: Cybersecurity, Management Automation
- Main Goal: Automate triage and analysis of security alerts, deliver actionable, explainable results, and make it easy for anyone to add new playbooks or enrichment tools for their own use case.

---

2. Problem Statement & System Functionality

Problem Statement

Modern SOCs face overwhelming numbers of alerts, manual enrichment tasks, and the constant risk of analyst fatigue and human error. Most current SIEMs lack built-in explainable automation or easy extensibility for new detection logic and threat intelligence integrations.

System Functionality

- Ingests security alerts from simulated SIEM or log sources.
- Stores all raw and analyzed alerts in PostgreSQL (accessible via pgAdmin).
- Let users trigger one-click or bulk automated analysis from a web dashboard.
- Uses an LLM-driven agent (OpenAI or local Deepseek) to:
  - Choose the correct playbook for each alert.
  - Dynamically select which threat enrichment tools to use.
  - Integrate external threat intelligence in real time.
  - Write a structured, fully explainable report for every alert.
- All steps, from alert ingestion to analysis and display, are fully containerized for easy deployment anywhere.

---

3. Architecture & Technology Stack

Layer	Technologies
Backend	Python, FastAPI, SQLAlchemy
Database	PostgreSQL (pgAdmin for DB browsing)
Frontend	React, Vite, Tailwind CSS, Lucide-React
Orchestration	Docker, Docker Compose
AI Agent	LangChain, OpenAI GPT-4o, Ollama + Deepseek
Enrichment	AbuseIPDB, VirusTotal, IPInfo.io
Explainability	Step-by-step JSON, rationale, and artifacts for review

All services run together with a single docker compose up command.

---

4. System Flow & Features

A. Alert Workflow

1. Alert Reception: Ingests alerts (JSON, simulating SIEM input).
2. Database Storage: Alerts are stored in the raw_alerts table.
3. Dashboard: “Alerts” tab lists all open (unanalyzed) alerts.
4. User Triggered Analysis: User can analyze a single alert or all at once via the frontend.
5. AI Agent (LLM) Process:
   - Selects the correct playbook from the playbook index (e.g., Brute Force, SQL Injection, Suspicious URL).
   - Recommends which enrichment tools to use based on playbook steps and alert fields, referencing the tool index.
   - Performs real-time threat intelligence lookups, or reuses previous enrichment if already available in the DB.
   - Collects related logs (by source or destination IP).

    - Runs step-by-step investigation (as per playbook), reasoning, and generates a final structured report (including: isolation advice, TP/FP, MITRE mapping, severity, recommendations, IOCs, and a human-readable summary for review).

6. Persistence: Final results stored in analyzed_alerts and raw alert is flagged as "analyzed".

7. Closed Alerts: All analyzed alerts (with full audit trail) are displayed in the "Closed Alerts" tab.

B. Enrichment & Playbook Engine

- Playbooks: All workflows are defined as editable JSON. Anyone can add new workflows, change steps, or introduce new logic just by updating a file.
- Threat Intel APIs: Each enrichment is a modular Python function (AbuseIPDB, VirusTotal, IPInfo.io, etc.). Adding new APIs is as simple as writing one new method and updating the tool index and .env file.
- AI Agent: Supports both OpenAI (like GPT-4o, GPT-o3, GPT-4.1, etc.… (cloud)) and Deepseek (local via Ollama). Easily switch LLMs in code or config.
- Log Ingestion: The logs table allows enrichment with real network or authentication logs, aiding deeper investigation.

C. Frontend Dashboard

- Responsive, modern UI built in React and Tailwind CSS.
- Tabs for "Logs", "Alerts", "Closed Alerts" with intuitive navigation.
- Expandable cards show all relevant fields and steps.
- Clear badges, colored indicators for severity/type, and live action feedback.
- All frontend API calls are made via a single REST interface (FastAPI backend).

---

5. How to Extend the System

A. To Add a New Playbook:

1. Add to playbook.json:
   - Define the workflow, step-by-step, with action types, instructions, and relevant fields.
2. Update the Playbook Index (playbook_index.txt):
   - Add a short summary so the AI agent can choose this playbook when appropriate.
3. (Optional) Add supporting logic or fields as needed in the frontend/backend.

B. To Add a New Enrichment Tool/API:

1. Add the tool name to the tool_index.txt file with a brief description of what it does.
2. Add the API key to the .env file (for local and Docker deployment).
3. Implement a Python function in enrichment.py to call the external API, process the response, and save results in the enrichment table.
4. Add a new conditional branch in the enrichment logic (usually in workflow.py) to call this function when the tool is requested by the AI agent.

The system is designed to be modular and "plug-and-play" for both playbooks and enrichment APIs.

---

6. Ethics, Explainability & Openness

- Every LLM-generated report includes explainability: the agent must reason step by step and produce auditable, human-readable output.
- No user or real production data is processed (prototype only).
- The codebase is 100% open source; anyone can clone, extend, and use it freely for education or real-world SOCs.

---

7. Scalability & Extensibility

- Playbooks: Just update a JSON file; instantly available to the AI agent.
- Enrichment: Add one Python method and the .env key; new API is in use.
- Frontend: Easily adapt to new alert types, fields, or user needs.
- Deployment: Simple, reproducible, and platform-agnostic: all you need is Docker.

---

9. Future Work / Improvements

- Live SIEM/log streaming for real-time alert ingestion.
- Add search, advanced filtering, and alert correlation in the frontend.
- Add more enrichment APIs (Shodan, URLScan, etc.).
- Automated remediation (network isolation, blocking, etc.) via SOAR.
- User authentication and role-based permissions for multi-analyst environments.

---

10. Conclusion

This project is a robust prototype for a next-generation SOC automation platform. It offers a transparent, explainable, and highly extensible AI-driven analyst. Anyone can clone and adapt it, add playbooks, add APIs, and customize the UI to fit their own SOC, educational, or research needs. The code is a foundation for both practical SOC automation and AI security research.

For any questions, contributions, or extensions, see the project README or contact the authors.

CONTACTS:

mohamadhassan.ibrahim711@gmail.com

reemkanaann20@gmail.com