Incident Response Automation Platform

A System for End-to-End SOC Automation

Document Version: 1.0

Author: Ritvik Indupuri

Date: October 19, 2025

1.0 Executive Summary

This document provides a detailed technical overview of the AI-Powered Incident Response (IR) Platform, a proof-of-concept system designed to automate the complete workflow of a Security Operations Center (SOC) analyst. Architected entirely within **Google's Opal** framework, the platform leverages a multi-agent system to manage the incident lifecycle, from initial data ingestion and analysis to the generation of a comprehensive, multi-section security report.

The system's core innovation is an autonomous agent with **Code Execution** capabilities, which dynamically writes and runs **Python (Pandas)** code to perform log analysis—automating the most resource-intensive task in a typical investigation. By orchestrating this with agents for threat intelligence, root cause analysis, business impact assessment, and remediation planning, the platform serves as a powerful "force multiplier." It validates a new paradigm for security operations, dramatically reducing Mean Time to Respond (MTTR) and enabling human analysts to focus on high-value strategic initiatives.

2.0 System Architecture & Data Flow

The system is architected as a Directed Acyclic Graph (DAG) where each node represents a specialized AI agent orchestrated by Google Opal. Data flows from user inputs through a series of parallel and sequential processing steps, with outputs from one agent serving as inputs for subsequent agents.

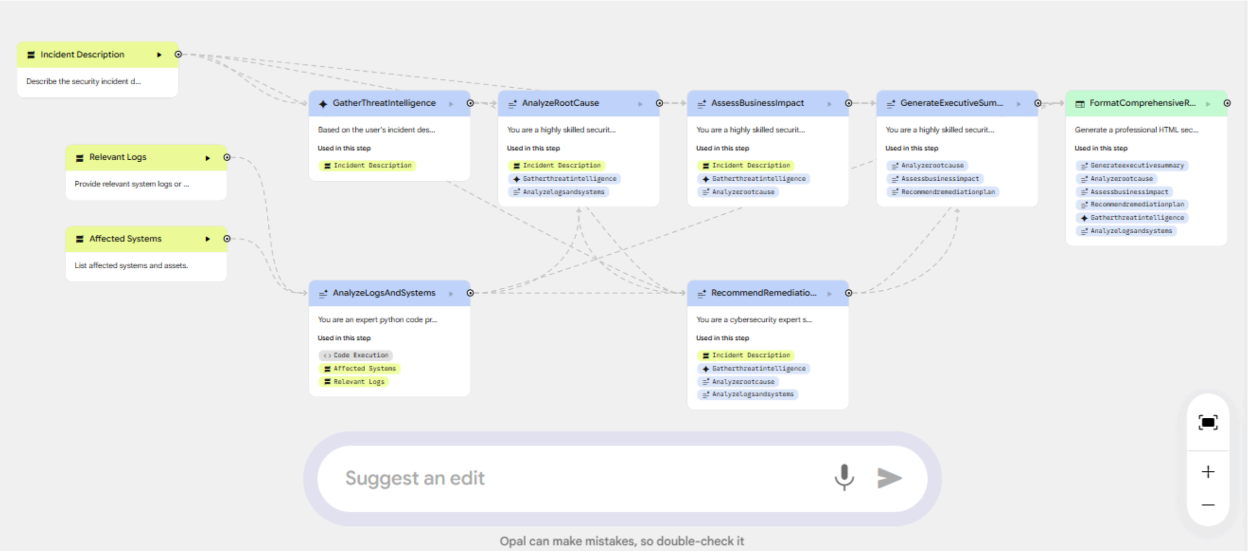


Figure 1: System Workflow Architecture

This diagram shows the complete data flow, from the three initial inputs through the parallel and sequential processing of the seven specialized AI agents, culminating in the final node.

Data Flow Logic:

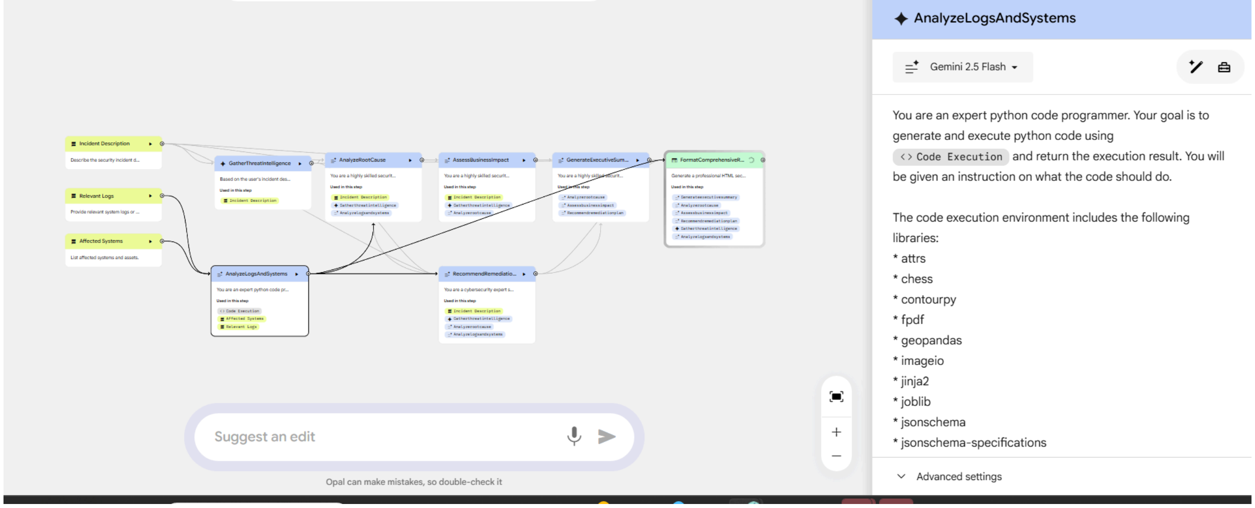
1. **Initiation (Input):** The workflow is triggered by three user inputs: Incident Description, Relevant Logs, and Affected Systems.
2. **Parallel Processing (Initial Analysis):**
   - The Incident Description is passed to the **GatherThreatIntelligence** agent.
   - The Relevant Logs and Affected Systems are passed to the **AnalyzeLogsAndSystems** agent.
3. **Sequential Analysis (Synthesis):**
   - Outputs from the initial analysis agents are fed into the **AnalyzeRootCause** and **RecommendRemediationPlan** agents.
   - The output from AnalyzeRootCause is then used by the **AssessBusinessImpact** agent.
4. **Executive Layer Synthesis:** Outputs from the synthesis agents are passed to the **GenerateExecutiveSummary** agent.
5. **Final Assembly (Reporting):** The **FormatComprehensiveReport** node collects the outputs from all preceding agents and formats them into the final HTML report.

3.0 Core Innovation: The Autonomous Code-Generating Agent

The platform's most significant technical capability is the **AnalyzeLogsAndSystems** agent. This component moves beyond standard natural language processing tasks into the realm of autonomous code generation and execution.

- **Function:** To act as an "expert Python programmer" with the objective of analyzing raw, unstructured, or semi-structured log files.
- **Capability:** The agent is granted **Code Execution** permissions within a sandboxed environment.
- **Process:** Upon receiving the log files, the agent autonomously determines the appropriate method for parsing and analysis, writes the necessary **Python (Pandas)** script, executes it, and returns the structured results (e.g., identified IoCs, attack timeline, user actions).

This is a critical differentiator, as it allows the system to remain adaptive and scalable. It is not constrained by pre-written parsers and can dynamically create the logic required to analyze new or unfamiliar log formats on the fly.



You are an expert python code programmer. Your goal is to generate and execute python code using `<> Code Execution` and return the execution result. You will be given an instruction on what the code should do.

The code execution environment includes the following libraries:
* attrs
* chess
* contourpy
* fpdf
* geopandas
* imageio
* jinja2
* joblib
* jsonschema
* jsonschema-specifications

Figure 2: The Core Engine - An Autonomous Python Agent

This agent is explicitly granted Code Execution capabilities and access to libraries like pandas to autonomously write and run Python code for log analysis, replacing a traditionally manual and time-consuming process.

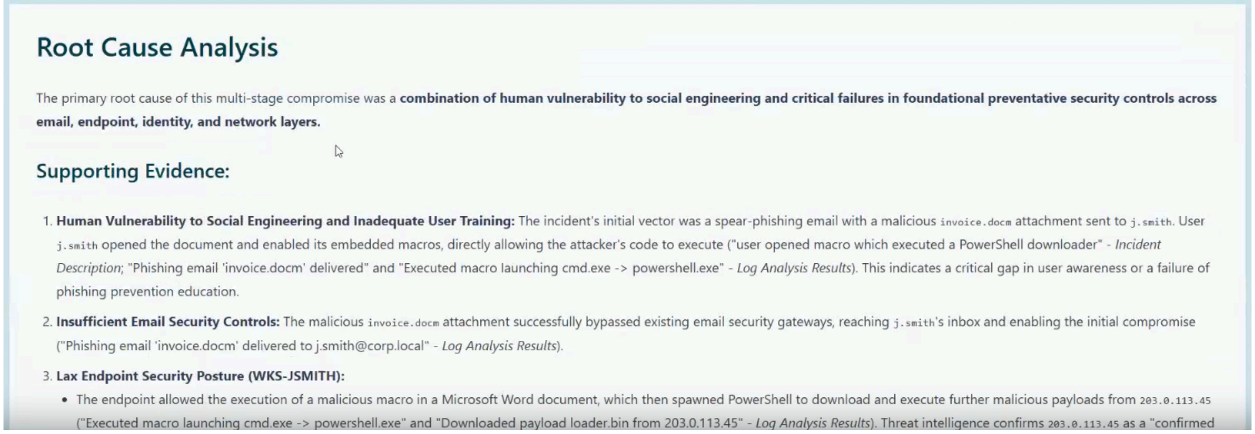
4.0 Output Specification & Verification

The final deliverable is a comprehensive, multi-section HTML security report designed for both technical analysts and executive stakeholders. The system's output quality has been verified to meet a professional standard.

The report includes the following sections, synthesized from the outputs of the various agents:

- Executive Summary
- Root Cause Analysis
- Business Impact Assessment
- Threat Intelligence Summary
- Detailed Remediation Plan
- Raw Log Analysis Results

The sample below demonstrates the **Root Cause Analysis** section, which proves the system's ability to synthesize evidence from both the user's initial description and the structured data returned by the autonomous Python agent.



Root Cause Analysis

The primary root cause of this multi-stage compromise was a **combination of human vulnerability to social engineering and critical failures in foundational preventative security controls across email, endpoint, identity, and network layers.**

Supporting Evidence:

1. **Human Vulnerability to Social Engineering and Inadequate User Training:** The incident's initial vector was a spear-phishing email with a malicious `invoice.docm` attachment sent to `j.smith`. User `j.smith` opened the document and enabled its embedded macros, directly allowing the attacker's code to execute ("user opened macro which executed a PowerShell downloader" - *Incident Description*; "Phishing email 'invoice.docm' delivered" and "Executed macro launching cmd.exe -> powershell.exe" - *Log Analysis Results*). This indicates a critical gap in user awareness or a failure of phishing prevention education.
2. **Insufficient Email Security Controls:** The malicious `invoice.docm` attachment successfully bypassed existing email security gateways, reaching `j.smith`'s inbox and enabling the initial compromise ("Phishing email 'invoice.docm' delivered to j.smith@corp.local" - *Log Analysis Results*).
3. **Lax Endpoint Security Posture (WKS-JSMITH):**
   - The endpoint allowed the execution of a malicious macro in a Microsoft Word document, which then spawned PowerShell to download and execute further malicious payloads from `203.0.113.45` ("Executed macro launching cmd.exe -> powershell.exe" and "Downloaded payload loader.bin from 203.0.113.45" - *Log Analysis Results*). Threat intelligence confirms `203.0.113.45` as a "confirmed

Figure 3: Sample Report Output - Professional-Grade Analysis

This tangible output shows the high-quality report generated in minutes. It clearly links the "Supporting Evidence" back to both the Incident Description and the Log Analysis Results.

5.0 Conclusion & Roadmap for Production

This proof-of-concept successfully validates that a multi-agent AI system, orchestrated via a platform like Google Opal, can automate the core functions of a SOC. The system's end-to-end nature—and particularly its ability to perform autonomous code generation for data analysis—represents a significant step toward a next-generation, autonomous SOC.

The following roadmap outlines the steps required to move this platform from a proof-of-concept to a production-ready system:

1. **Live API Integration:** Connect the platform directly to our SIEM and EDR (e.g., Splunk, CrowdStrike) to enable fully automated, real-time alert triage and response.
2. **Stateful Memory:** Implement a vector database to give the system memory of past incidents, allowing it to identify recurring attacker TTPs and improve its analysis over time.

3. **SOAR Capabilities:** Empower the platform to not just *recommend* remediation actions but, with approval, to *execute* them via API calls (e.g., disabling a user account, blocking an IP on a firewall, isolating an endpoint).
