Fontys Hogescholen

Function Design

FDD

ZOLDER IO
19-9-2024

REVISION TABLE

Date	Remark	Reviewed By	Feedback
20 Sep 2024	*Init Document*		

Inhoud

1. Introduction

Cybersecurity tools can be difficult for small and medium-sized enterprises (SMEs) to fully utilize due to their complexity. Many SMEs struggle to interpret the data produced by these tools, especially when monitoring Office 365 environments. To address this challenge, Zolder B.V. is developing an AI agent. This agent will help users understand security incidents by providing clear explanations and actionable insights.

The AI agent will interact with AtticSecurity to process incidents and respond to user queries. It will also generate queries for Microsoft Sentinel to offer further context for alerts. The goal is to create a Proof of Concept (PoC) to demonstrate how such an agent can reduce the burden on support teams and improve the security response for SMEs.

1.1 Purpose

The AI agent is designed to reduce complexity for SMEs by helping users interpret security incidents from Office 365 environments. Leveraging the Llama 3.1 LLM, which utilizes distinct roles (System, User, Ipython, Assistant), it integrates with AtticSecurity and Microsoft Sentinel to provide explanations, actionable insights, and automated Kusto Query Language (KQL) queries.

2. Functional Requirements

2.1 AI Agent Capabilities

- **Incident Understanding:** The AI agent, utilizing the **Assistant** role of Llama 3.1, analyzes incidents from AtticSecurity, explains them in human-readable terms, and provides actionable insights to users.
- **Query Generation:** Leveraging the **Ipython** role, the agent generates KQL queries to retrieve additional context or information from Microsoft Sentinel based on user prompts.
- **User Interaction:** Through a chat interface, the AI agent interacts with users using the **User** and **Assistant** roles to interpret security incidents, answer queries, and offer recommendations, while the **System** role ensures context and guidelines are maintained.
- .

2.2 System Requirements

- **Input Data**:
  Office 365 security logs and incident data from AtticSecurity and Microsoft Sentinel.
- **Output**:
  Human-readable explanations, recommended actions for security incidents, and results from KQL queries.

3. Use Case Scenarios

3.1 User Queries Incident Data

- **Scenario**: A user asks the AI agent to explain a security alert from Office 365.
- **Process**:
  - The agent retrieves relevant data from AtticSecurity.
  - It processes the incident to offer a clear explanation of what happened.
  - It suggests next steps (e.g., block a suspicious IP or reset a user password).

3.2 Generating KQL Queries

- **Scenario**: The user requests additional information about a specific security event, such as identifying other users logging in from the same IP.
- **Process**:
  - The agent generates and executes a KQL query using Microsoft Sentinel.

- It formats and returns the query results in an easy-to-read format.

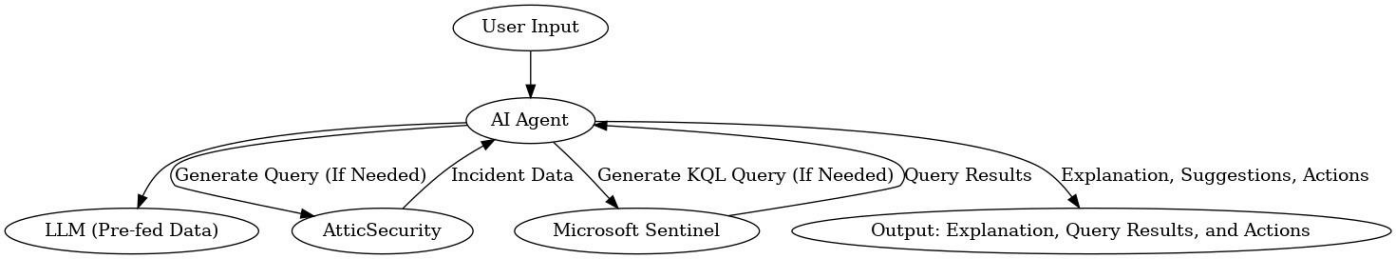
4. System Architecture

4.1 Overview

- **Components**:
  - **AI Agent:** Core component responsible for interaction, incident processing, and query generation, powered by Llama 3.1 hosted on Ollama.
  - **AtticSecurity:** Provides the initial security incident data.
  - **Microsoft Sentinel:** Executes KQL queries for additional data.
  - **User Interface (UI):** Chat interface for user interaction, built with Flask.

4.2 Flow of Data

1. **User Input**:
   The process begins when the user submits a query through the chat interface, initiating an interaction with the AI agent.
2. **AI Agent**:
   The AI agent, hosted on Ollama and utilizing Llama 3.1, receives the query. It processes the input using the **User** role and generates responses using the **Assistant** role. The **System** role sets the context and guidelines for the interaction.
3. **Query Generation (If Needed)**:
   - If the query cannot be fully answered using the pre-fed data, the AI agent employs the **Ipython** role to generate specific KQL queries for additional data retrieval from AtticSecurity or Microsoft Sentinel.
4. **Data Retrieval from AtticSecurity**:
   The AI agent retrieves incident data such as alerts, event logs, and security findings relevant to the query..
5. **KQL Query from Microsoft Sentinel**:
   The AI agent generates and executes Kusto Query Language (KQL) queries via Microsoft Sentinel SDKs to obtain deeper insights or detailed records of incidents.**AI Agent Processing**: Once data is retrieved, the AI agent analyzes and merges it with the original user query, utilizing the **Assistant** role to formulate an enriched response.
6. **Output**:
   The agent then generates a response that includes:
   - A clear explanation of the incident.
   - Query results (e.g., logs, suspicious activity patterns).
   - Recommended actions, such as blocking IP addresses or resetting passwords.


User Input
AI Agent
Generate Query (If Needed)
Incident Data
Generate KQL Query (If Needed)
Query Results
Explanation, Suggestions, Actions
LLM (Pre-fed Data)
AtticSecurity
Microsoft Sentinel
Output: Explanation, Query Results, and Actions

5. Technical Requirements

5.1 Technology Stack

- **Backend**: Python-based API using Flask for handling user interactions and integrating with the AI agent.
- **Machine Learning Models**Llama 3.1 LLM hosted on Ollama, utilizing its role-based messaging capabilities (System, User, Ipython, Assistant).
- **Cloud Integration**: Integration with Azure for Microsoft Sentinel via SDKs, managed through Flask and the AI agent.
- **Databases**: Logging and history database for tracking user queries and incidents, ensuring compliance and auditability.

5.2 APIs and SDKs

- Microsoft Sentinel REST API for querying security events.
- AtticSecurity API for incident retrieval.
- Azure Identity for secure access control.
- **Ollama API/SDK** (if applicable) for managing and interacting with the hosted Llama 3.1 model.

6. Security Considerations

6.1 Data Privacy and Compliance

- How sensitive information in security alerts and logs will be handled securely.
- Ensuring compliance with data protection regulations like GDPR.

6.2 Access Control

- Use of role-based access control (RBAC) to ensure that only authorized personnel can interact with the AI agent and access sensitive security data.

7. Performance Requirements

- **Response Time**: The AI agent must respond to user queries within a reasonable timeframe, ideally under 5 seconds, considering the latency introduced by the Ollama-hosted LLM.
- **Scalability**: The system should be scalable to handle multiple simultaneous queries from users. Utilizing Ollama's capabilities alongside cloud services like Azure ensures scalability and reliability.

8. Testing and Validation

- **Unit Testing**: Each function (incident interpretation, KQL query generation, etc.) will be tested individually.
- **Integration Testing**: Full system testing, ensuring smooth interaction between AtticSecurity, Microsoft Sentinel, and the AI agent.
- **User Acceptance Testing (UAT)**: Test the agent with SMEs to validate if the agent simplifies security incident interpretation as intended.

9. Deployment and Maintenance

- **Deployment Plan:** Initial PoC deployment on a cloud service (e.g., Azure) using Docker for containerization. The Llama 3.1 model is hosted and managed via Ollama, ensuring streamlined installation and updates.
- **Maintenance Schedule:** Regular updates to the AI model through Ollama, updates to the incident response knowledge base, and maintenance of API endpoints. Monitor and manage resource allocation and performance through Ollama's management tools.

10. Model and Hosting Details

10.1 Llama 3.1 LLM

We utilize the Llama 3.1 LLM, which supports four distinct roles to enhance interaction and functionality:

- **System:** Sets the context, including rules and guidelines for the AI's responses.
- **User:** Represents the human inputs, including commands and queries.
- **Ipython:** Marks messages with tool outputs, facilitating integration with external data sources.
- **Assistant:** Generates responses based on the context provided by the other roles.

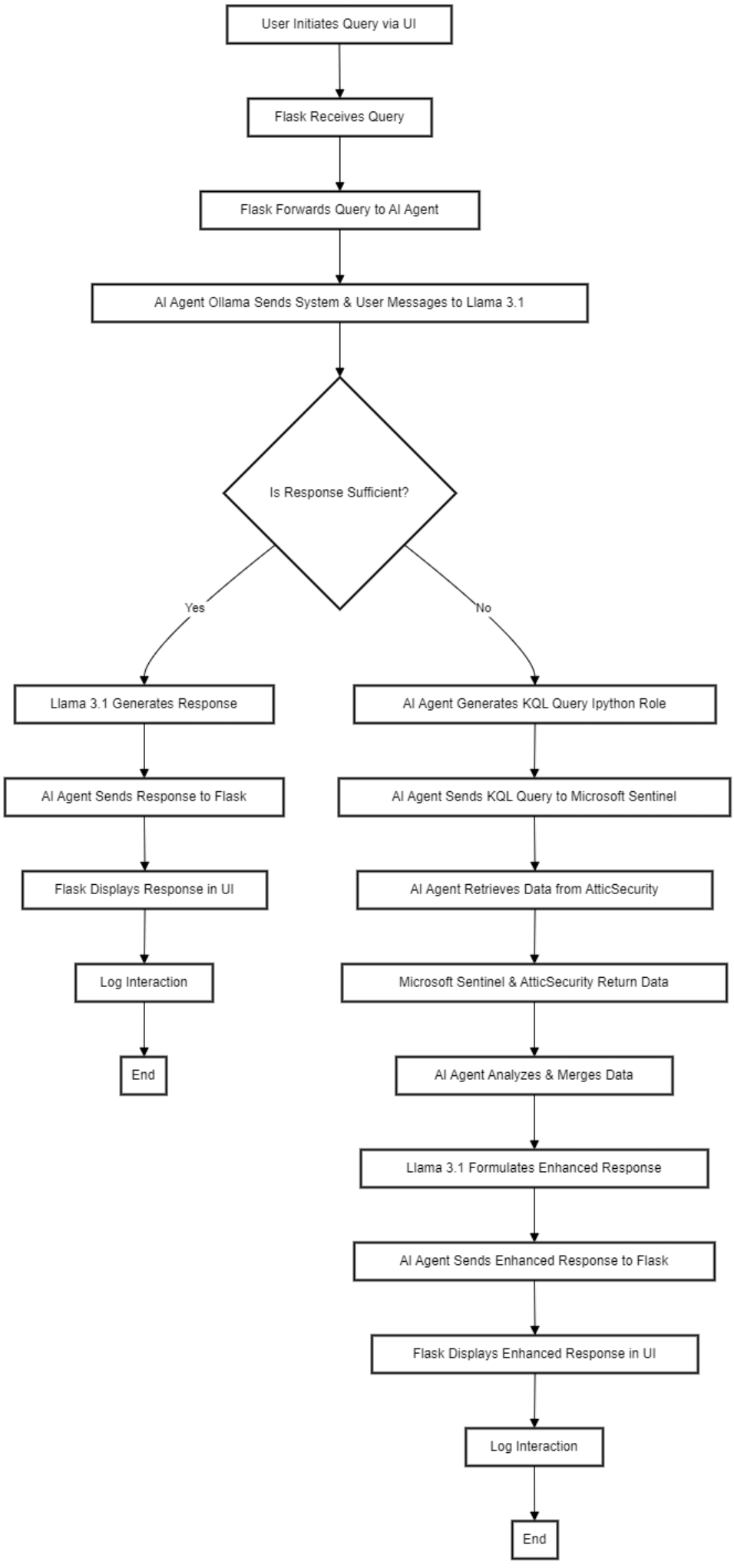
10.2 Hosting with Ollama

The Llama 3.1 model is hosted using Ollama, a tool that simplifies the installation and management of large language models on local or cloud-based systems. Ollama allows for:

- **Custom Model Management:** Easily update and customize the LLM as needed.
- **Scalability:** Efficiently handle multiple requests and scale resources based on demand.
- **Security:** Ensure secure deployment and access control for the AI agent.

Integration with Flask: The Flask application interfaces with Ollama to send and receive messages to the Llama 3.1 model, handling user interactions, incident processing, and query generation seamlessly.

Diagram & FlowChart


User Initiates Query via UI
Flask Receives Query
Flask Forwards Query to AI Agent
AI Agent Ollama Sends System & User Messages to Llama 3.1
Is Response Sufficient?
Yes
No
Llama 3.1 Generates Response
AI Agent Sends Response to Flask
Flask Displays Response in UI
Log Interaction
End
AI Agent Generates KQL Query Ipython Role
AI Agent Sends KQL Query to Microsoft Sentinel
AI Agent Retrieves Data from AtticSecurity
Microsoft Sentinel & AtticSecurity Return Data
AI Agent Analyzes & Merges Data
Llama 3.1 Formulates Enhanced Response
AI Agent Sends Enhanced Response to Flask
Flask Displays Enhanced Response in UI
Log Interaction
End