Implement Threat Intelligence Principles

Document understanding of threat intelligence through practical implementation. The project must include analysis of 2 Indicators of Compromise (IoCs) with detection methods used and how they indicate threats. The project must also demonstrate implementation of OpenCTI Threat Intelligence Platform using either Docker or system installation, including configuration of at least 2 connectors, documentation of platform setup and connector integration, and basic usage demonstration. All implementations must include proper documentation and evidence of functionality.

Analysis
Analysis

- Query successful

Okay, let's break down how to approach this Threat Intelligence Principles implementation project based on the provided information.

I. Analysis of 2 Indicators of Compromise (IoCs)

To fulfill this part, we need to define what IoCs are, select two, explain detection methods, and describe how they indicate threats. Here's a structured approach:

- **What are IoCs?** Indicators of Compromise are forensic data, such as data found in system log files or files, that identify potentially malicious activity on a system or network. IoCs can include file hashes, IP addresses, URLs, or registry keys.
- **IoC Selection and Analysis:**
    - **IoC 1: IP Address - 192.168.1.100**
        - **Detection Method:** Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS) can be used to monitor network traffic and flag connections to or from this IP address. Firewall logs can also show connections. SIEM (Security Information and Event Management) systems can aggregate and correlate logs from various sources to detect patterns.
        - **Indication of Threat:** If this IP address is associated with a known command-and-control server for malware, or if there are unusual patterns

of communication (e.g., excessive data transfer, connections to unusual ports), it could indicate a compromised system communicating with an attacker.

- **IoC 2: File Hash - "a1b2c3d4e5f6..." (SHA256)**
  - **Detection Method:** Endpoint Detection and Response (EDR) tools can monitor file activity and calculate hashes of files. Antivirus software also uses hash databases to identify known malware. File integrity monitoring tools can detect unauthorized changes to files.
  - **Indication of Threat:** If this hash matches a known malware file, it's a strong indicator that the file on the system is malicious. Even if it's an unknown hash, unusual file activity (e.g., execution from a temporary directory, attempts to modify system files) in conjunction with the hash can be suspicious.

- **Documentation:** This section requires detailed documentation of the analysis:
  - Clear definitions of IoCs.
  - Justification for the chosen IoCs.
  - Step-by-step explanation of the detection methods.
  - A thorough explanation of how each IoC indicates a potential threat, including possible attack scenarios.

II. OpenCTI Threat Intelligence Platform Implementation

The project requires setting up OpenCTI and integrating connectors. Based on the provided `docker-compose.yml`, Docker is the chosen implementation method.

- **OpenCTI Implementation using Docker:**
  - **Setup:**
    1. **Install Docker and Docker Compose:** (This is a prerequisite, and you'll need to document the installation process based on your OS).
    2. **`docker-compose.yml`:** The provided `docker-compose.yml` file is the core of the setup. It defines the OpenCTI instance and the connectors.
    3. **Configuration:** The `docker-compose.yml` file shows environment variables used to configure OpenCTI and the connectors. Important variables include:
       - `OPENCTI_URL`: The URL of the OpenCTI instance (e.g., `http://localhost`).

        - `OPENCTI_TOKEN`: The administrator's token for authentication.
        - Connector-specific variables (e.g., `CISA_CATALOG_URL` for the CISA connector).
    4.
    5. **Running OpenCTI:** Use `docker-compose up -d` to start the OpenCTI platform and the connectors. The `-d` flag runs it in detached mode (in the background).
  - **Documentation:**
    1. Document the Docker and Docker Compose installation.
    2. Include the `docker-compose.yml` file in the documentation.
    3. Explain each configuration variable in the `docker-compose.yml` and its purpose.
    4. Document the commands used to start OpenCTI.
    5. Provide screenshots of the running containers.
- **Connector Integration:**
  - The `docker-compose.yml` file already defines two connectors:
    6. **CISA Known Exploited Vulnerabilities:** This connector imports data on known exploited vulnerabilities from the CISA catalog.
    7. **OpenCTI Datasets:** This connector imports various datasets, including information on sectors, geography, and companies.

- **Configuration:** The `docker-compose.yml` file shows the configuration of these connectors through environment variables. For example, `CISA_CATALOG_URL` specifies the source of the CISA data. `CONNECTOR_INTERVAL` determines how often the OpenCTI Datasets connector updates.

- **Documentation:**
  1. Explain the purpose of each connector.
  2. Document any specific configuration done for the connectors.
  3. Provide evidence (e.g., screenshots from the OpenCTI interface) that the connectors are running and importing data.

- **Basic Usage Demonstration:**

  - **Accessing OpenCTI:** Access the OpenCTI web interface (e.g., `http://localhost`, as defined in the `docker-compose.yml`). The screenshot from the browser shows the OpenCTI interface.

  - **Basic Operations:**
    1. **Searching:** Demonstrate searching for imported data (e.g., search for a specific vulnerability or a company).
    2. **Viewing Data:** Show how to view details of imported objects (e.g., vulnerability information, indicator details). The dashboard in the screenshot shows some initial data views.
    3. **Creating Objects (Optional):** If feasible, demonstrate manually creating a simple object (e.g., an indicator).
  - **Documentation:**
    1. Include screenshots of the OpenCTI interface.
    2. Provide step-by-step instructions for the basic operations performed.
    3. Explain how the connectors' data is reflected in the OpenCTI platform.

III. Key Documentation Points

- **Clear Structure:** Organize the documentation with clear headings and subheadings.

- **Step-by-Step Instructions:** Provide detailed, step-by-step instructions for all implementation steps.
- **Screenshots:** Include screenshots to provide visual evidence of the setup, configuration, and usage. The provided screenshots are examples of what to include.

- **Configuration Details:** Thoroughly document all configuration settings, including environment variables and any changes made to configuration files.
- **Explanation of Concepts:** Explain the underlying concepts of threat intelligence, IoCs, and the OpenCTI platform.
- **Troubleshooting (Optional):** If you encounter any issues during implementation, document the problems and their solutions.

This detailed breakdown should give you a solid framework for completing your Threat Intelligence Principles implementation project. Remember to focus on clear documentation and providing evidence for each step.