Report – Cyber Threat Intelligence Dashboard

1. Introduction

Cyber threats are growing in scale and complexity, targeting organizations' digital assets and critical infrastructure. To address these challenges, actionable, real-time cyber threat intelligence (CTI) is vital for effective defence and rapid incident response. The Cyber Threat Intelligence Dashboard project was developed to provide security analysts and IT teams with an integrated, user-friendly platform for threat lookup, trend visualization, and reporting.

2. Abstract

This project is a web-based dashboard built using Flask that aggregates and visualizes threat intelligence from open-source APIs. Users can input IP addresses or domain names to verify them against multiple threat intelligence databases, visualize trends over time, and export lookup results. The dashboard is designed for modularity, extensibility, and ease of use, making it suitable for both academic and professional environments. With interactive graphs, tagging, and robust error handling, the platform reduces manual effort and improves situational awareness for cybersecurity operations.

3. Tools Used

- Flask: A lightweight Python web framework for building the server-side application logic and handling HTTP requests.
- Plotly: Used for creating interactive and dynamic data visualizations, such as trend graphs and activity charts.
- PyMongo: Enables integration with MongoDB for storing and retrieving lookup history, tags, and user activity logs.
- Requests: Facilitates communication with external CTI APIs for real-time threat data retrieval.
- dnspython: Provides advanced parsing and validation for domain names and IP addresses.
- HTML/CSS: Used for frontend development to ensure a responsive, visually appealing, and intuitive user interface.
- MongoDB: Serves as the primary database for storing recent lookups and tagged intelligence data.

4. Features Overview

- Real-Time Threat Lookup: Users can submit IP addresses or domain names to check against multiple threat intelligence feeds.
- Historical Metrics Visualization: The dashboard displays time-series charts of threat activity, lookup counts, and threat levels using Plotly.
- Tagging and Export: Lookups can be tagged for future reference, and results can be exported in common formats (CSV, PDF, Excel).
- Recent Lookups: Maintains a searchable, filterable table of recent user queries and their threat statuses.
- Robust Error Handling: Comprehensive input validation and contextual error messages guide users through the lookup process.
- User Experience: Clean, modern interface with clear navigation and interactive elements for efficient analysis.

5. Steps Involved in Building the Project

5.1. Requirement Analysis: Defined key functionalities including IOC (Indicator of Compromise) lookup, visualization, tagging, export capabilities, and error management. Mapped out user stories and system architecture.

5.2. Environment Setup: Configured a Python virtual environment and installed all dependencies listed in `requirements.txt` (Flask, PyMongo, Requests, Plotly, dnspython).

5.3. Backend Development: Developed the main application logic in `app.py`, implementing routes for dashboard views, API integrations, and database operations. Ensured modular code structure for scalability.

5.4. Frontend Development: Designed HTML templates (`dashboard.html`, `index.html`, etc.) in the `templates` folder and styled them using `style.css` in the `static` folder. Integrated Plotly for dynamic chart rendering.

5.5. API Integration: Established secure connections to open-source threat intelligence APIs, parsed responses, and normalized data for display and storage.

5.6. Feature Implementation: Implemented tagging, exporting, recent lookups, and error handling. Developed the `errors-check.txt` table to systematically test all input validation and error scenarios.

5.7. Testing and Quality Assurance: Conducted extensive manual and automated testing of user flows, error messages, data visualizations, and export features to ensure reliability.

5.8. Documentation and Reporting: Prepared a detailed `README.md` file with installation, usage, and troubleshooting instructions. Created comprehensive project documentation and this report for evaluation.

6. Use Case Scenario

A security analyst suspects a phishing attempt from a suspicious domain. Using the dashboard, they quickly input the domain name and receive instant feedback on its threat status, historical activity trends, and related IOCs. The analyst tags the lookup for further investigation, exports the result for reporting to management, and reviews recent lookups to identify patterns of repeated threat activity. This workflow illustrates how the dashboard accelerates threat identification and streamlines reporting.

7. Challenges & Solutions

- Challenge: Integrating multiple, often inconsistent, CTI APIs.
- Solution: Developed a unified data model and robust error-handling routines to gracefully manage API errors or downtime.
- Challenge: Ensuring real-time responsiveness with growing data volumes.
- Solution: Leveraged efficient MongoDB indexing and optimized Flask route handling for rapid query processing.
- Challenge: Presenting complex threat data in an intuitive way.
- Solution: Used Plotly for interactive visualizations and designed clear, user-friendly templates for all dashboard components.

8. Conclusion

The Cyber Threat Intelligence Dashboard demonstrates how open-source tools can be combined to deliver a comprehensive, real-time CTI platform. By automating threat lookups, providing intuitive visualizations, and supporting tagging and export, the dashboard streamlines the cyber threat analysis workflow. The modular design supports easy integration of new threat feeds and advanced analytics in future iterations. This project showcases the effective application of Python, Flask, and modern web technologies in addressing real-world cybersecurity challenges.

9. References

- Flask Documentation – https://flask.palletsprojects.com
- Plotly Python Open-Source Graphing Library – https://plotly.com/python
- PyMongo Documentation – https://pymongo.readthedocs.io
- MongoDB Manual – https://docs.mongodb.com
- dnspython Documentation – https://dnspython.readthedocs.io
