SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

CYBER THREAT INTELLIGENCE DASHBOARD: A REAL-TIME VISUALIZATION PLATFORM

G Madhan, B Manoj, M Nithish Kannan, Mr. KS Arun
UG Student, Department of Cyber Security, Mahendra Engineering College, Mallasamudram, Tamil Nadu, India.
Under the guidance of: Mr. KS Arun, M.Tech., Department of Cyber Security, Mahendra Engineering College, Mallasamudram, Tamil Nadu, India.

Article DOI: https://doi.org/10.36713/epra22041
DOI No: 10.36713/epra22041

ABSTRACT
The "Cyber Threat Intelligence Dashboard" is a web application developed to deliver real-time visualization of cyber threat data through an intuitive, user-friendly interface. Threat intelligence, encompassing details on ransomware, phishing, malware, and other attack vectors, is primarily sourced from the AlienVault Open Threat Exchange (OTX) API. The system dynamically fetches, classifies, and presents this data, offering features like live search functionality and a dark mode toggle to enhance user experience. Built using Python with the Flask framework for backend operations and HTML, CSS, and Chart.js for frontend rendering and data visualization, the dashboard aims to provide a clear and responsive overview of the current cyber threat landscape. Error handling is incorporated to manage potential API request issues, ensuring a stable platform for threat awareness.
KEYWORDS: *Cyber Threat Intelligence, Dashboard, AlienVault OTX, Data Visualization, Flask, Real-time Threats, Web Application, API Integration.*

INTRODUCTION

In the modern digital environment, the capacity to monitor and comprehend the evolving landscape of cyber threats is crucial for security professionals, organizations, and system administrators. Various forms of cyberattacks, including malware, phishing, and ransomware, represent significant risks with unique impact potentials. The "Cyber Threat Intelligence Dashboard," a web application meticulously developed using the Python Flask framework, directly addresses the critical need for real-time, consolidated threat data, providing users with an accessible and comprehensive overview of ongoing cyber threats.

The motivation for this dashboard originates from the escalating volume and sophistication of cyber threats and the consequent necessity for timely, actionable intelligence. The dashboard seeks to consolidate and visualize data from a recognized threat intelligence source, enabling security teams to more rapidly identify, understand, and potentially preempt emerging threats. Instead of navigating multiple complex data feeds, users can access current threat information within a single, cohesive application interface. This consolidation is intended to save time and effort, fostering a more holistic understanding of the threat environment.

The core functionality revolves around fetching and displaying real-time threat data from the AlienVault OTX API. AlienVault OTX is a community-driven platform that provides access to a vast repository of threat "pulses," which are collections of indicators of compromise (IOCs) and related threat information. The application focuses on processing these pulses to extract meaningful information, such as threat names, descriptions, and creation dates, and then classifying them into understandable categories.

The "Cyber Threat Intelligence Dashboard" application incorporates a range of features designed to enhance user experience and provide valuable threat insights:

- Real-Time Data Updates: The application dynamically fetches the latest threat pulses from AlienVault OTX.
- Automated Threat Classification: Threats are categorized (e.g., Ransomware, Phishing, Malware) based on keywords found in their descriptions and names.
- Visual Threat Representation: Threat information is displayed in individual "cards," each featuring a color-coded badge indicating the threat type for quick identification.
- Graphical Data Overview: A bar chart, implemented using Chart.js, provides a visual summary of the distribution of different threat types.
- Keyword-Based Search: A search functionality allows users to filter the displayed threats based on specific keywords.
- User-Friendly Interface: Built with HTML, CSS, and JavaScript for the frontend, the application emphasizes a clean, responsive, and intuitive user interface.

SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

- Dark Mode Option: A toggle switch allows users to switch between light and dark themes, enhancing visual comfort in different lighting conditions.
- Error Handling: The application includes mechanisms to gracefully manage potential API request failures, maintaining a stable user experience.

Technical Architecture

The "Cyber Threat Intelligence Dashboard" is built using Python and the Flask web framework for the backend, with HTML, CSS, and JavaScript for the frontend. The application follows a client-server architectural pattern. The user interacts with the frontend via a web browser (client), which sends requests to the Flask backend (server). The backend processes these requests, interacts with the external AlienVault OTX API to retrieve threat data, performs data processing and classification, and then serves the formatted data along with the web pages to the client. The requests library is utilized for making HTTP requests to the OTX API.

WORKING PROCESS

The "Cyber Threat Intelligence Dashboard," leveraging the Flask framework, is a dynamic tool designed to present users with real-time cyber threat intelligence. The application's architecture is structured for modularity and a clear user experience.

1. Application Initialization and UI Construction

The application's operation begins with the Flask app instance. The build process is responsible for constructing the user interface when a user accesses the application.

- Setting the Stage: The application title "Cyber Threat Intelligence Dashboard" is set in the HTML template.
- Root Page Structure: The main HTML page (dashboard.html) serves as the primary container for all UI elements. It includes a header, a dark mode toggle, a canvas for the threat chart, a search form, and a container for threat cards.
- Styling: CSS (style.css) is used to define the visual appearance, including background colors, font styles, layout of elements, and specific styles for light and dark modes.

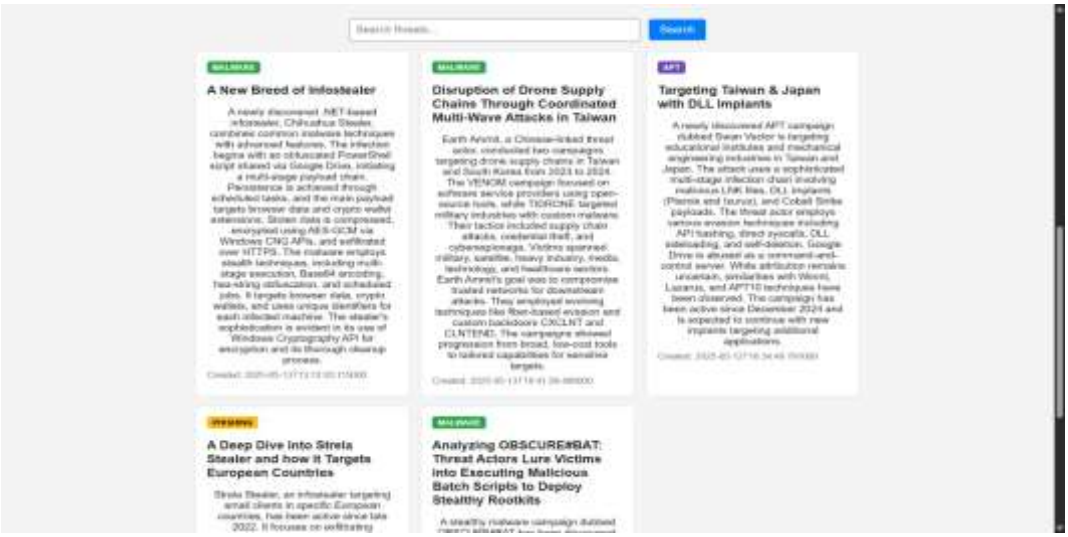
2. Threat Data Display Workflow (Main Page)

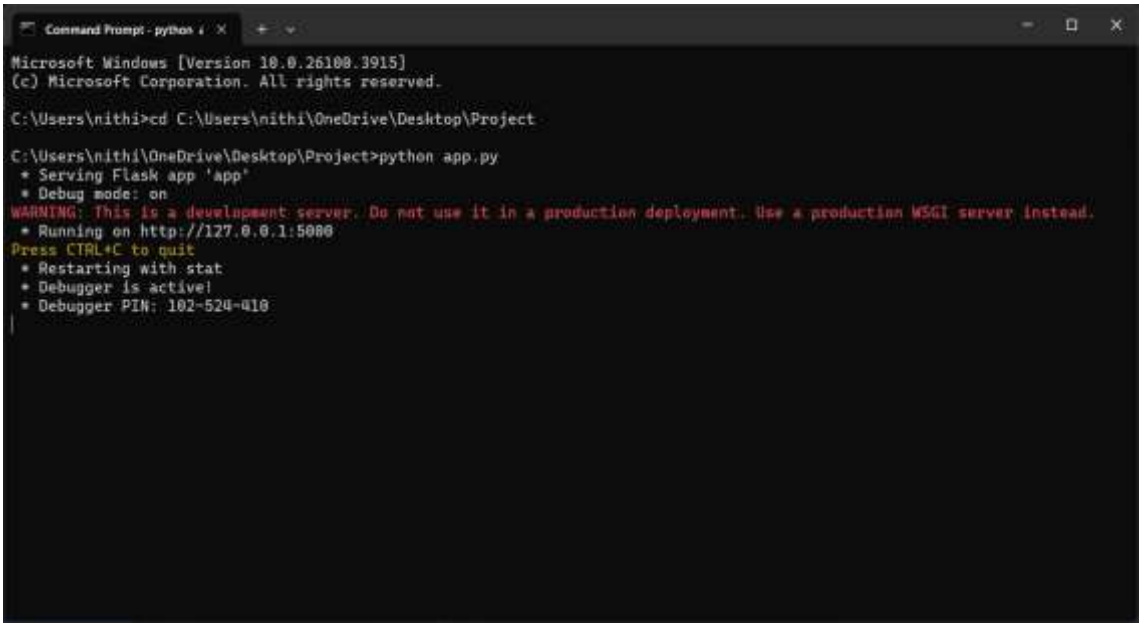
The main page is designed to display real-time threat information.

- Layout Management: The main content area uses a CSS grid layout to arrange threat cards in a responsive manner. A search form is placed prominently.
- Label Initialization (Threat Cards): When the page loads, threat pulses fetched by the backend are iterated through. For each pulse, a "card" is generated. These cards display information like threat name, description, creation date, and a type badge. The badge's text and color are determined by the backend classification.
- Button/Form Configuration: A search button triggers a GET request to the same page with the search query, which the backend then uses to filter results.
- Widget Assembly: The individual threat cards are dynamically added to the main container by the Flask templating engine based on the data passed from the Python backend.
- Data Fetching and Updating
- Threat Data Fetching:
    - API Request: The application makes a GET request to the OTX API endpoint. The request includes the API key in the headers.
    - Response Handling: The JSON response is parsed. The classification function determines the threat category based on keywords.
    - Label Updates: The fetched and classified pulses are passed to the HTML template and rendered as threat cards.
    - Error Handling: A try-except block manages errors during API requests, returning an empty list if an error occurs.
- Chart Data
    - An API endpoint provides aggregated threat counts by type in JSON format.
    - JavaScript initializes a bar chart using Chart.js with predefined labels and data.

SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

OUTPUT





SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

RESULTS AND DISCUSSION

1. Threat Data Display and Classification
- Observed Results: The dashboard fetches threat data from the AlienVault OTX API, displaying threat pulses with names, descriptions, and creation dates on cards. The classification categorizes threats like "Ransomware," "Phishing," etc.
- Discussion: AlienVault OTX is effective for a broad overview. Keyword-based classification is simple and useful but may miss complex threats. Data updates currently rely on page refresh.
- Potential Improvements:
    - Use machine learning for better classification.
    - Add JavaScript-based automatic data refresh.
    - Improve error messaging for missing data.

2. Search Functionality
- Observed Results: The search bar filters threat pulses based on keyword matches in names or descriptions.
- Discussion: Effective for narrowing results, though limited to basic matching.
- Potential Improvements:
    - Enable advanced queries by field or pattern.
    - Highlight matched terms in the UI.

3. Data Visualization (Chart)
- Observed Results: A static bar chart shows threat types. An API endpoint exists for dynamic data.
- Discussion: Visualizes threat distribution but not updated in real-time.
- Potential Improvements:
    - Connect the frontend chart to dynamic data.
    - Add interactive features like tooltips.
    - Use diverse chart types for better analysis.

4. User Interface and User Experience
- Observed Results: Clean interface with cards and dark mode. Search and badges improve usability.
- Discussion: Responsive design and theme toggle enhance comfort.
- Potential Improvements:
    - Refine CSS styling and add icons.
    - Improve layout across devices.
    - Add accessibility features.

5. Error Handling
- Observed Results: Backend uses try-except blocks for API errors, returning empty lists if failed.
- Discussion: Ensures backend stability but doesn't notify users.
- Potential Improvements:
    - Show frontend error notifications.
    - Implement retry logic.
    - Add logging for debugging.

6. Performance
- Observed Results: Adequate performance with API limits. Acceptable load and search times.
- Discussion: Performance depends on API response and connection speed.
- Potential Improvements:
    - Add caching to reduce API load.
    - Use asynchronous requests for real-time updates.
    - Optimize backend data handling.

7. Security
- Discussion: API communication uses HTTPS. API key is hardcoded.
- Potential Improvements:
    - Store API key securely using environment variables.
    - Sanitize user input to prevent XSS or injection.
    - Regularly update dependencies.

CONCLUSIONS

The "Cyber Threat Intelligence Dashboard" is a centralized platform for real-time monitoring and visualization of cyber threats. Using Python Flask for the backend and standard web technologies for the frontend, it provides a responsive and accessible interface. Integration with the AlienVault OTX API ensures access to current threat data.

SJIF Impact Factor (2025): 8.688| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 10 | Issue: 5 | May 2025 - Peer Reviewed Journal

Key features like threat cards, color-coded badges, search, and visual charts enhance usability. Dark mode improves user comfort. Future improvements could include advanced classification logic, dynamic chart updates, and asynchronous threat feed refresh. These enhancements would increase the application's robustness, interactivity, and effectiveness as a cybersecurity awareness tool.