Threat Intelligence and Risk Management Framework

For ShopSmart Solutions

Joseph Kimbrough, Jaylen Sinclair, Seymon

COMPSCI 361

Professor Hasan

26 April 2025

Table of Contents

Abstract & Introduction

In today's rapidly evolving cybersecurity landscape, real-time visibility into potential threats is essential for timely response and mitigation. Our project presents a comprehensive Real-Time Threat Intelligence Dashboard that integrates Open Source Intelligence (OSINT) data with automation, machine learning, and actionable insights. Leveraging the Shodan API, we collect and store threat intelligence such as open ports and exposed services in a PostgreSQL database. A Python-based backend, coupled with an automated scheduler, ensures threat data is updated every six hours. The system integrates a risk-scoring module using a pre-trained HuggingFace language model to simulate large language model (LLM) reasoning, aiding in dynamic risk prioritization. The React-based front-end dashboard offers a live, interactive view of threat data, while alert systems and mitigation modules provide proactive defenses. Additional features include threat-vulnerability-asset (TVA) mapping, automated risk prioritization, and the generation of incident response plans—creating a complete threat management pipeline.

Cybersecurity threats are growing in complexity and frequency, requiring organizations to adopt proactive, intelligence-driven approaches for defense. Traditional static monitoring tools often fail to provide the timely and contextual awareness needed for effective response. To address this gap, our group project introduces a Real-Time Threat Intelligence Dashboard, a unified platform that collects, analyzes, and visualizes live OSINT threat data while offering actionable insights for security teams.

This system integrates the Shodan API to fetch external threat intelligence, particularly focusing on open ports and services that may pose vulnerabilities. The data is stored in a PostgreSQL database and refreshed automatically every six hours using a Python scheduler. On the backend, a Flask API manages data flow between components, while an alerting module sends real-time email notifications for threats with risk scores above a set threshold. To simulate intelligent risk analysis, we incorporate ChatGPT, enabling nuanced risk assessment and prioritization.

Key enhancements include a dynamic Threat-Vulnerability-Asset (TVA) mapping, an automated mitigation recommendation engine, and an incident response plan generator. The frontend, developed using React, visualizes live threat intelligence and provides an intuitive user experience for monitoring and management. Overall, the project serves as a practical implementation of a modern threat intelligence framework, emphasizing automation, machine learning, and actionable cybersecurity operations.

System Architecture

```
├── api
│   ├── __pycache__              # Python cache
│   ├── api_optimizer
│   ├── security_trails.py
│   ├── fetch_osint.py           # Shodan API Integration & OSINT data fetch
│   ├── osint.py                 # Additional OSINT API integrations
│   └── scheduler.py             # Automated script for real-time data fetching
├── db
│   └── (Database files)         # Database scripts & schema
├── docs
│   ├── nist_framework_summary.md    # NIST framework summary
│   ├── osint_research.md        # Research on OSINT sources
│   ├── team_structure.md        # Team roles and structure
│   └── tech_stack.md            # Tech stack used in the project
├── src
│   ├── threat-dashboard         #Frontend files
│   ├── threat-intelligence-platform
│   │   ├── app.py               # Flask app entry point
│   │   └── components           # React components folder
├── src
│   ├── alerts.py                # Email alert system
│   ├── incident_response.py     # Incident Response Plan generator
│   ├── mitigation_recommendations.py# Automated risk mitigation recommendations
│   ├── risk_analysis.py         # Risk scoring using ML (T5 model integration)
│   └── risk_prioritization.py   # Dynamic risk prioritization model
├── tests
│   └── api_tests.py             # Unit tests for API endpoints
├── .gitignore                   # Git ignored files
├── database.txt                 # Notes on database setup
├── README.md                    # Project documentation
└── requirements.txt             # Python dependencies
```

Technologies Used

1. Back-End : Python/Flask
2. Front-End : React
3. Database : Postgres
4. OSINT API : Shodan, SecurityTrails API

5. LLM Model for Risk Scoring : Chat GPT-4o

Implementation Details

As you can see above in the System Architecture, our code design and structure is illustrated. We have integrated the Shodan and SecurityTrails API for retrieving detailed information about them, and subscribing to real-time data feeds. It provides methods for both interactive search and data streaming, catering to various needs like security research, network monitoring, and vulnerability assessment. Our implementation of TVA Mapping, or Threat-Vulnerability-Asset Mapping, connects threats, the vulnerabilities they exploit, and the assets they affect. The information that we store in a Postgres database that we implemented is fully displayed in the frontend, allowing for quick access to data along with the function to search vulnerabilities that are stored.

Security Features & Blue Teaming Strategies

In our real-time threat intelligence model, we utilized ChatGPT to analyze the Threat and Vulnerability Assessment (TVA) data, enabling the generation of actionable security recommendations. By leveraging ChatGPT's natural language processing capabilities, we efficiently interpreted complex security logs and vulnerability data, streamlining the identification of critical threats. This process facilitated the generation of a comprehensive Risk Report, outlining potential vulnerabilities and offering mitigation strategies tailored to our system architecture. Key areas of focus included API key protection through secure handling practices and access restrictions, as well as the implementation of environmental variables to safeguard sensitive configuration data. These enhancements contribute to a more robust and adaptive security posture within our threat intelligence framework.

Testing & Performance Results

For the Testing & Performance Results section, we conducted comprehensive evaluations by running the full threat intelligence program in a simulated real-time environment. All core components—including data ingestion, threat analysis, risk report generation, and security recommendation modules—were executed successfully without errors. The system demonstrated reliable performance under typical load conditions, with timely processing of vulnerability data and consistent output of accurate security insights. Integration points, such as API communication and environmental variable handling, operated as intended, confirming the

stability and efficiency of our framework. These results validate the effectiveness and readiness of the system for deployment in dynamic cybersecurity environments.

Cost-Benefit Analysis & Business Justification

In the Cost-Benefit Analysis & Business Justification, our real-time threat intelligence model demonstrates a high return on investment by significantly enhancing cybersecurity posture with minimal resource expenditure. Utilizing tools like ChatGPT for automated analysis reduced the need for extensive manual review, lowering labor costs and accelerating threat response times. Open-source technologies and secure implementation practices, such as API key protection and the use of environmental variables, minimized infrastructure expenses while maintaining robust security standards. The system's scalability and integration capabilities further support long-term cost efficiency. From a business perspective, the proactive identification and mitigation of threats protect critical assets, ensure compliance, and reduce the potential costs associated with data breaches or downtime, making this solution both economically and strategically advantageous.

Challenges Faced & Lessons Learned

We encountered several obstacles during the development and implementation of our real-time threat intelligence model. One major challenge was ensuring the secure handling of sensitive data, particularly API keys and configuration variables, which required the adoption of best practices like environmental variable management and strict access controls. Additionally, integrating multiple data sources and ensuring compatibility with our analysis tools posed technical difficulties, particularly in maintaining real-time performance under varying loads. We also faced limitations in interpreting some unstructured or incomplete TVA data, which highlighted the importance of data normalization and quality assurance. Through these challenges, we learned the critical value of early-stage security planning, modular system design for easier debugging and updates, and the importance of continuous testing in dynamic cybersecurity environments. These lessons will inform future iterations and enhancements of our framework.

Future Enhancements & Recommendations

We aim to further strengthen the security and responsiveness of our real-time threat intelligence model. A key enhancement under consideration is the implementation of automated email alerts, which will notify security teams when a threat is detected with a risk score of 20 or higher. This proactive communication feature will ensure timely awareness and faster incident response. Additionally, we recommend expanding the system's security measures by

incorporating advanced encryption protocols for data in transit and at rest, along with multi-factor authentication for administrative access. Future updates will also explore integrating machine learning to improve threat prediction accuracy over time. These enhancements will not only bolster the system's resilience but also ensure it remains agile and effective against evolving cybersecurity threats.