Enhancing Threat Intelligence for Internet of Things (IoT) Systems

University Polytechnique Montréal

Student: Joseph Mansour

Supervisor: Dr. Omar Abdul Wahab

Date: August 28, 2025

Table of Contents

Abstract

This report summarises a project that aimed to build a full-stack platform for enhancing threat intelligence for Internet of Things (IoT) systems. The system inventories IoT devices, associates them with critical infrastructure sectors, enriches them with vulnerability intelligence from the National Vulnerability Database (NVD) common vulnerability and exposures (CVE) feed, and provides AI-assisted summaries of relevant threats. The front-end interface, built with SvelteKit, allows users to browse and manage devices and sectors. A Django back-end exposes a REST API and triggers automatic enrichment when devices are created or updated. A local large language model, served by Ollama, generates concise risk summaries without sending sensitive data to external services.

The platform contributes to the cybersecurity community by centralizing IoT asset inventories and providing contextualized vulnerability insights for critical infrastructure sectors. Automated CVE enrichment reduces manual effort and accelerates analyst workflows. Limitations include the reliance on device identification heuristics and restricted scalability. Future work will focus on network discovery using tools like Nmap and improving the scalability and automation of threat intelligence operations.

Introduction

The rapid proliferation of IoT devices across industry, healthcare, transportation and home environments has created new attack surfaces for cyber adversaries. In critical infrastructure sectors, sensors and controllers connect operational technologies to the internet, yet they often lack robust security controls and receive infrequent updates. Known vulnerabilities may remain unpatched because operators lack visibility into the devices they own or the relevant threats. Traditional vulnerability management tools target servers and desktop systems, leaving a gap in situational awareness for IoT environments.

Threat intelligence aims to bridge this gap by collecting and analyzing vulnerability data, then contextualizing it for specific assets and sectors. However, manual enrichment is time-consuming and prone to omissions. This project responds to the need for automated, sector-aware threat intelligence for IoT devices. Over the summer, Joseph Mansour developed a prototype platform under the supervision of Dr. Omar Abdul Wahab that inventories IoT devices, maps them to critical infrastructure categories, enriches them with NVD CVE data, and generates AI-assisted summaries to support security analysts.

System Overview

The system is composed of three main layers: a front-end user interface, a back-end API with enrichment logic, and an intelligence layer for vulnerability data and AI summarisation. Key objectives guiding the design include:

• Providing a clean user interface to browse, search and manage IoT devices and their associated sectors.
• Assigning each device to one or more critical infrastructure sectors to enable contextual risk assessments.
• Automating enrichment with CVE intelligence drawn from the NVD feed and generating concise natural‑language summaries.
• Presenting colour‑coded risk indicators and links to authoritative sources to facilitate analyst decision‑making.

Key features of the platform include a RESTful API for creating, reading, updating and deleting (CRUD) operations on devices and sectors. When a device is added, an enrichment routine queries the NVD feed for CVEs matching device identifiers, calculates severity scores and stores a list of threats. A local large language model is then prompted to produce a concise summary of likely attack vectors, impacts and mitigations. The front‑end displays devices and their associated threats with risk colour cues and supports filtering by sector.

Implementation Details

Data model

The back‑end uses Django and Django REST Framework with a SQLite database for development. The `IoT Device` model stores device identity and description fields and maintains a many‑to‑many relationship with the `Sector` model. Each sector represents a critical infrastructure category. The `Threat` model stores CVE identifiers, severity scores and risk ratings, while the `Threat_Detail` model records human‑readable descriptions and the AI‑generated summary of each CVE. A `Threat_Info_Category` model groups threat details into categories such as description and mitigation. These classes and relationships are defined in the backend code.

Automatic enrichment

A Django signal triggers the vulnerability enrichment pipeline when a device is created. The enrichment function queries the NVD API with device identifiers, extracts CVSS metrics and determines a risk level (Low, Medium, High or Critical) based on CVSS base scores. It then generates a friendly attack name and instructs the local LLM via Ollama to summarise the impact, attack vectors and mitigations in plain language. The routine creates `Threat` and `Threat_Detail` records linking the CVEs to the device. Because Ollama runs locally, no sensitive device information leaves the system.

Front‑end and user interface

The front‑end is implemented with SvelteKit, using Tailwind CSS and DaisyUI for styling. Upon loading, it fetches the list of sectors and devices from the back‑end. Devices are sorted

by risk severity and displayed in a scrollable list. Selecting a sector filters the device list to those belonging to the chosen sector. Each device card shows a coloured badge representing the highest risk among its associated threats (e.g., red for Critical). Clicking a device reveals a detailed view of its associated CVEs and the AI-generated summary. Users can add new devices through a dialogue that collects the device name, description and sector; upon submission, the enrichment pipeline runs asynchronously.

Deployment

For ease of use, the project includes Docker Compose files that orchestrate the deployment of the front-end, back-end, Ollama, and an optional Nginx reverse proxy. Separate stacks are provided for CPU, NVIDIA GPU and AMD GPU environments. In the default configuration, the front-end listens on port 4173, the back-end on 8080 and Ollama on 11434. Developers can spin up the entire stack with a single command, enabling rapid local testing without external dependencies.

Contribution to Cybersecurity and Threat Intelligence

This platform contributes to the cybersecurity community in several ways:

Centralized IoT asset and sector classification

By cataloguing devices and mapping them to critical infrastructure sectors, the tool provides a consistent foundation for risk assessment across domains.

Automated CVE aggregation

Integration with the NVD feed retrieves up-to-date vulnerability information for each device, eliminating the need for manual lookups and reducing analyst workload.

Contextualization of risk

Sector assignments inform the impact and prioritization of vulnerabilities. For example, a high-severity CVE on a medical sensor may warrant more urgent action than the same CVE on a consumer gadget.

AI-assisted summaries

The use of a local large language model provides concise, digestible descriptions of threats, attack vectors and recommended mitigations, helping analysts quickly understand complex vulnerabilities.

Privacy and control

Because the AI model runs locally via Ollama, the system avoids sending sensitive device metadata to external services and supports air-gapped deployments.

Extensibility

Clear separation between the UI, API and enrichment logic facilitates integration with additional data sources such as intrusion detection systems, threat intelligence platforms (TIPs) or security information and event management (SIEM) tools.

Developer convenience

The Docker-based deployment and REST API accelerate experimentation and adoption by researchers and practitioners.

Limitations

Despite its contributions, the prototype has some limitations that should be acknowledged:

Dependence on accurate device identification

The enrichment routine relies on matching device names or identifiers to relevant CVEs. Incomplete or ambiguous device descriptions can lead to missed vulnerabilities or false positives.

Heuristic mapping

Mapping CVEs to devices uses heuristic keyword matching. More sophisticated techniques, such as firmware fingerprinting or machine-learning-based classification, could improve accuracy.

Scalability

The current implementation uses SQLite and synchronous tasks suitable for development. Scaling to large fleets or real-time operations would require migrating to a production-grade database (e.g., PostgreSQL) and adopting asynchronous processing and message queues.

Limited analytics

Risk scores are derived directly from CVSS metrics, which are not intended for patch management prioritization. Integrating additional threat feeds or behavioural analytics like EPSS (‘Exploit Prediction Scoring System’) could provide richer risk insights.

Manual review required

AI-generated summaries should be reviewed by analysts before action. Large language models can hallucinate or omit important details.

User roles and security

The current prototype assumes a trusted environment. A production deployment would need authentication, authorization and audit logging to prevent misuse.

Future Enhancements

Several avenues for future work were identified during development:

Network discovery with Nmap

Integrating a network scanner such as Nmap would allow automatic discovery of IoT devices on a subnet, seeding the inventory without manual entry. Service fingerprinting could provide richer metadata for CVE mapping and threat modelling.

Scalability improvements

Migrating the database to PostgreSQL and adopting asynchronous task queues (e.g., Celery) would enable processing of large numbers of devices and CVEs. Container orchestration could support horizontal scaling.

GPU‑accelerated AI inference

Hosting the large language model on a server with a GPU would reduce inference latency and allow for more complex summarisation models. Users could still run the model locally if desired.

Enhanced heuristics and machine learning

Training models on firmware signatures, service banners or vendor metadata could improve the accuracy of CVE mapping. Natural language processing could suggest mitigation strategies tailored to specific sectors.

Integrations with security tools

Exposing enrichment results through APIs or webhooks would allow ingestion by SIEMs or TIPs, enabling automated prioritization and response. Automated alerting when new CVEs appear for existing devices would further enhance situational awareness.

User experience improvements

Adding dashboards, search capabilities, and graphical representations of risk posture would make the tool more intuitive for non‑technical stakeholders.

Conclusion

The Enhancing Threat Intelligence for Internet of Things (IoT) Systems project demonstrates a practical approach to automating vulnerability enrichment for IoT devices and presenting the results in a user‑friendly interface. By combining a SvelteKit front‑end, a Django REST API and a local large language model, the system centralizes asset inventory, enriches devices with CVE intelligence and delivers concise threat summaries. The tool contributes to the cybersecurity community by reducing manual workload, contextualizing risk through sector assignments and enabling offline deployment. While the prototype has limitations related to

identification accuracy, scalability and the need for manual review, the roadmap outlines clear directions for improvement, including automated network discovery, enhanced heuristics and integration with broader security ecosystems. This report provides a foundation for future research and development to further advance IoT threat intelligence and protect critical infrastructure.

Acknowledgements

The author thanks Dr. Omar Abdul Wahab for his guidance and supervision during the project. His expertise in cybersecurity and IoT systems was invaluable in shaping the design and evaluation of this prototype. The author also acknowledges the open-source communities behind Django, SvelteKit, Tailwind, DaisyUI, Ollama and the National Vulnerability Database for providing the tools and data that made this project possible.