Zero-Day Cyber Sentinel

A Real-Time Threat Intelligence System

Team CICtards

January 2026

Abstract

This report presents the design and implementation of **Zero-Day Cyber Sentinel**, a real-time threat intelligence dashboard that fuses live CVE data from the NIST National Vulnerability Database (NVD) with an organization's asset inventory. The system leverages Pathway's streaming engine for incremental data processing and Google Gemini for AI-powered risk analysis. The result is a sub-5-second pipeline from threat publication to actionable alert, significantly outperforming traditional batch-based vulnerability management systems.

Contents

1 Problem Statement

In modern cybersecurity, **time is the most critical resource**. The attack surface of organizations is constantly expanding, and adversaries are increasingly leveraging zero-day vulnerabilities—exploits that are published and weaponized before defenders can react.

Traditional vulnerability management systems suffer from three fundamental limitations:

1. **Polling/Batching Delays:** Most commercial tools check for new CVEs on a scheduled basis (e.g., nightly or weekly scans). This creates a dangerous window where a published vulnerability is known to attackers but not yet flagged by defenders.
2. **Manual Asset Matching:** Security teams often maintain spreadsheets or static databases to cross-reference CVEs with their internal asset inventory. This process is slow, error-prone, and fails to scale as the number of CVEs and assets grows.
3. **Lack of Contextual Analysis:** A CVE with a CVSS score of 9.8 is not equally critical for every organization. Without context about *which specific assets* are affected and *why* the vulnerability matters, security teams are left to manually triage an overwhelming volume of alerts.

Zero-Day Cyber Sentinel addresses these problems by implementing a **real-time streaming architecture** that:

- Ingests CVE data the moment it is published.
- Automatically filters threats against a live asset inventory.
- Uses Generative AI to explain the risk in plain English.

2 System Workflow

The system is designed as a **three-stage pipeline**, with each stage decoupled for resilience and debuggability.

2.1 Architecture Diagram

$$\begin{tikzcd}
\texttt{NIST API} \arrow[r] & \texttt{stream\_generator.py} \arrow[r] & \texttt{stream.jsonl} \arrow[d] \\
& \texttt{inventory.csv} \arrow[r] & \texttt{Pathway Engine (logic.py)} \arrow[d] \\
& & \texttt{Gemini API} \arrow[d] \\
& & \texttt{alerts.jsonl} \arrow[d] \\
& & \texttt{Streamlit (app.py)}
\end{tikzcd}$$

Figure 1: High-Level Data Flow

2.2 Stage 1: Ingestion (`stream_generator.py`)

The ingestion layer is responsible for polling the NIST NVD API and writing raw threat data to a local JSONL file.

- **Polling Frequency:** Every 5 seconds.
- **Data Source:** NIST NVD REST API v2.0.
- **Simulated Data:** For demonstration purposes, the generator can also inject simulated threats or accept manual input via a text file (`manual_input.txt`).

2.3 Stage 2: Processing (`logic.py` - Pathway)

This is the core of the system. Pathway reads from `stream.jsonl` and `inventory.csv`, performs a streaming join, and enriches matched threats with AI analysis.

```
matches = threats.join(
    inventory,
    pw.left.join_key == pw.right.join_key
).filter(
    is_relevant(pw.this.description, pw.this.product)
)
```

Listing 1: Pathway Streaming Join (Simplified)

2.4 Stage 3: Visualization (`app.py` - Streamlit)

The Streamlit dashboard polls `alerts.jsonl` and displays matched, analyzed alerts in real-time. It also provides:

- Toast notifications for new critical alerts.
- A historical trend chart of threat volume.
- An AI chatbot ("SENTINEL") for security Q&A.

3 Pathway Usage

Pathway (`https://pathway.com`) is a Python framework for building real-time data pipelines. Unlike traditional batch-processing tools (e.g., Pandas, Spark), Pathway processes data **incrementally** as it arrives.

3.1 Why Pathway?

Feature	**Traditional (Pandas)**	**Streaming (Pathway)**
Data Model	Static DataFrames	Live, updating Tables
Join Semantics	One-time, in-memory	Continuous, stateful
Latency	Minutes to hours	Milliseconds to seconds
Parallelism	Manual (multiprocessing)	Automatic (Rust engine)

Table 1: Comparison of Batch vs. Streaming Paradigms

3.2 Key Pathway Concepts Used

1. **Connectors (pw.io):** The system uses `pw.io.jsonlines` to read from and write to JSONL files. Pathway automatically watches these files for new lines.
2. **Streaming Joins:** The core logic performs a `join` between the live threat stream and the static inventory. Pathway maintains the inventory in memory and applies the join to each new threat as it arrives.
3. **User-Defined Functions (UDFs):** The `@pw.udf` decorator allows embedding custom Python logic (e.g., the `is_relevant` filter) directly into the pipeline.
4. `pw.apply`**:** For more complex operations like calling an external API (Gemini), `pw.apply` allows invoking arbitrary Python functions on table columns.

```python
@pw.udf
def is_relevant(desc: str, prod: str) -> bool:
    if not desc or not prod:
        return False
    return prod.casefold() in desc.casefold()
```

Listing 2: UDF for Relevance Filtering

4 Key Design Decisions

4.1 JSONL as Inter-Process Communication (IPC)

Instead of using a message queue (e.g., Kafka, RabbitMQ), the system uses simple `.jsonl` files for communication between stages.

Rationale:

- **Simplicity:** No external infrastructure required. The system can run on a single machine with no dependencies.
- **Debuggability:** Files can be inspected with `cat` or `tail -f` for real-time debugging.
- **Pathway Compatibility:** Pathway's `jsonlines` connector natively supports file watching.

Trade-off: This approach is not suitable for high-throughput, distributed systems. For production, a Kafka connector would be recommended.

4.2 Hybrid Data Stream (Real + Simulated)

The `stream_generator.py` mixes real NIST data with occasional simulated threats.

Rationale:

- **Demo Reliability:** NIST may not publish new CVEs during a live demo. Simulated data ensures the dashboard is always "alive."
- **Testing:** Custom threats (e.g., "fake_virus_of_CICs") can be injected to test specific matching logic.

4.3 AI-First Analysis

Every matched threat is sent to Google Gemini for a contextual, plain-English explanation.

Rationale:

- **Actionability:** A CVSS score of 9.8 is not actionable. "Update your Nginx server immediately due to a remote code execution flaw" is.
- **Knowledge Base Augmentation:** The AI can be "pre-taught" about organization-specific threats by injecting learnings into `knowledge_base.jsonl`.

4.4 Decoupled Microservice Architecture

The three Python scripts (`stream_generator.py`, `logic.py`, `app.py`) run as independent processes.

Rationale:

- **Fault Isolation:** If the Streamlit dashboard crashes, the data pipeline continues to run.
- **Independent Scaling:** In a production environment, each component could be scaled independently (e.g., multiple Pathway workers).
- **Development Velocity:** Developers can modify one component without restarting the others.

5 Technology Stack

Component	**Technology**
Streaming Engine	Pathway (Python + Rust)
AI Analysis	Google Gemini 1.5 Flash
Dashboard	Streamlit
Data Source	NIST NVD API v2.0
Containerization	Docker

Table 2: Technology Stack Summary

6 Future Work

- **Slack/Discord Alerts:** Push critical alerts to messaging platforms for immediate operator notification.
- **Kafka Connector:** Replace JSONL files with a Kafka topic for production-grade message durability and scalability.
- **Historical Trend Analysis:** Store historical alerts in a time-series database (e.g., InfluxDB) for long-term trend analysis.
- **Multi-Tenant Inventory:** Support multiple organizations with separate inventories and access controls.

7 Conclusion

Zero-Day Cyber Sentinel demonstrates that real-time threat intelligence is achievable with modern streaming frameworks and Generative AI. By combining Pathway's sub-second processing latency with Gemini's contextual analysis, the system reduces the mean-time-to-detect (MTTD) from hours to seconds, providing a significant defensive advantage against emerging threats.