Cyber Threat Intelligence Data Pipeline

Cloud Data Processing Project

Real-Time Threat Intelligence Ingestion and Analysis using Azure & Databricks

1. Student Details

Student ID: Manleen Kaur (49231) | Arun Kumar Johnson (53034)

Email ID: manleen.kaur2@mail.dcu.ie | arunkumar.johnson2@mail.dcu.ie

2. Link for Dataset and Source Code Repository

GitLab Repository: Cloud Project GitLab

Dataset Original Link**:** https://catalog.data.gov/dataset/electric-vehicle-population-data

3. Link to Video Demonstration

Video Demonstration: https://drive.google.com/file/d/1n-kHIa5qORGE7mZBtmzU0T264qOlE4kY/view?usp=sharing

4. Introduction and Motivation

4.1 Project Overview

This project deploys a scale-to-production Cyber Threat Intelligence (CTI) data pipeline that automatically pushes threat intelligence feeds of the AlienVault Open Threat Exchange (OTX) into production. The solution will be based on Azure cloud and Databricks to produce a scalable medallion architecture (Bronze, Silver, Gold layers) with Change Data Capture (CDC) to manage real-time threat intelligence.

4.2 Business Problem

The volume of cyber threats that organizations have to deal with on a daily basis is overwhelming, and new Indicators of Compromise (IoCs) are found in the flow. The security teams require automated mechanisms that receive threat intelligence feeds, monitor evolving threat landscapes, deduplicate IoCs and detect critical threats that should be addressed urgently. The manual processing is time-consuming, prone to errors and it is incapable of scale to meet the contemporary cybersecurity requirements.

4.3 Solution Approach

This pipeline is meeting these challenges by:

1. The threat intelligence sources will ingest their API real-time every 15 minutes.
2. Medallion architecture of data quality and governance.
3. Delta Lake with CDC to monitor changes and modifications of IoC.
4. Threat categorization, normalization and automated data cleansing.
5. Critical threat alerting through Email using Azure Logic Apps.

4.4 Technology Selection

Technology	Justification
Azure Data Factory	Serverless orchestration with REST API connectors and 15-minute scheduling
Databricks	Optimized Spark engine for processing threat feeds at scale with Delta Lake support
Delta Lake	ACID transactions and CDC for tracking IoC changes over time
ADLS Gen2	Scalable storage for raw threat feeds and processed IoCs
Azure Logic Apps	Low-code automation for email alerts via Azure Monitor on critical threats

5. Related Work

Related Work This part will be discussing three available threat intelligence platforms and comparing them with the solution implemented.

5.1 Related Systems

1. **MISP (Malware Information Sharing Platform)** - Open-source threat intelligence platform aimed at sharing of IoCs between organisations. Is manual and automated ingestion, which is highly infrastructure-intensive and does not have inherent cloud scalability.

2. **OpenCTI** - Knowledge graph based threat intelligence platform, capable of visualization. Threat models everything, but is resource-consuming and requires a steep learning curve to implement and maintain.

3. **ThreatConnect** - API-based threat intelligence platform that includes orchestration capabilities. Strong and costly to small-medium organizations and mainly, threat sharing and not data lake architecture.

5.2 How This Solution Differs

Feature	MISP	OpenCTI	This Solution
Cloud-Native	No	Partial	Yes (Azure)
CDC Support	Basic	Yes	Yes (Delta)
Medallion	No	No	Yes
Auto-Scale	No	Limited	Yes

Notable Differentiator: This solution uses cloud-native Azure services to achieve automatic scaling, employs medallion architecture to achieve data quality, uses Delta Lake CDC to provide a sophisticated threat tracking, and the solution needs less infrastructure management than self-hosted solutions.

5.3 References

- MISP Project: https://www.misp-project.org/
- OpenCTI Platform: https://www.opencti.io/
- ThreatConnect: https://threatconnect.com/

6. Description of the Dataset

6.1 Source of the Data

Primary Source: AlienVault Open Threat Exchange (OTX)

API Endpoint: https://otx.alienvault.com/api/v1/pulses/subscribed

Authentication: API Key (X-OTX-API-KEY header)

Data Format: JSON

Storage Location: Databricks Delta Lake tables (dbw_cti_processing database)

6.2 Process of Extraction/Collection

Automated Ingestion Process

1. **Scheduled Execution:** Azure Data Factory pipeline runs every 15 minutes
2. **API Request:** HTTP GET to OTX API with authentication header
3. **Raw Storage:** JSON responses stored in ADLS Gen2 raw layer
4. **Processing:** Databricks notebooks process through Bronze → Silver → Gold layers
5. **Final Storage:** Delta Lake tables in Databricks for querying and analysis

6.3 Dataset in Databricks

The processed data is stored in Databricks Delta Lake tables. The screenshot below shows a sample table from the Gold layer demonstrating the Delta Lake storage structure:

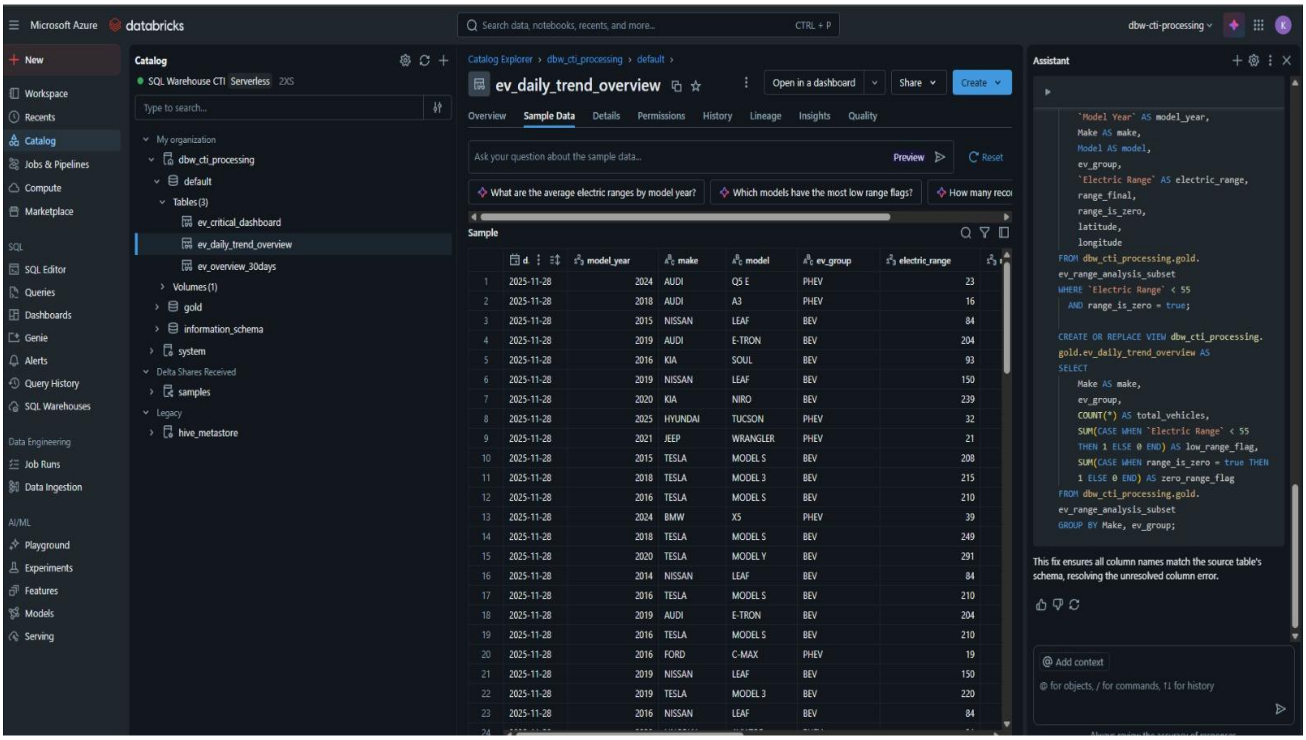

FIG- Sample Gold layer table in Databricks showing Delta Lake structure

The real CTI pipeline takes the threat intelligence pulses that contain IoCs in corresponding Delta Lake tables. The data structure consists of threat metadata (pulse ID, name, author, timestamps), indicators (IPs, domains, hashes, URLs), threat classifications (tags, TLP levels, severity), and reference links (stored in Bronze, Silver and Gold layers).

7. Description of Data Processing

The data processing implements the medallion architecture pattern with three layers. All processing code is available in the GitLab repository under databricks/notebooks/.

7.1 Bronze Layer Processing

Purpose

The Bronze layer serves as the immutable landing zone for raw data. It preserves complete data lineage while enabling efficient querying through Delta Lake format.

Processing Steps

- **Data Ingestion:** Read raw files from Azure Data Lake Storage Gen2
- **Schema Extraction:** Infer or apply schema to the incoming data
- **Metadata Addition:** Add ingestion_timestamp, source, and partition columns
- **CDC Implementation:** MERGE operation ensures idempotency and tracks changes
- **Delta Lake Storage:** Write to partitioned Delta table with Change Data Feed enabled

7.2 Silver Layer: IoC Normalization

Data Quality and Enrichment

- **Data Cleansing:** Remove duplicates, handle nulls, standardize formats
- **IoC Parsing:** Extract and categorize indicators by type (IPv4, domain, hash, URL)
- **Threat Categorization:** Classify by threat type (Malware, Phishing, APT, Ransomware)
- **Deduplication:** Identify and merge duplicate IoCs across pulses
- **Validation:** Schema enforcement, IP format validation, hash verification

7.3 Gold Layer: Threat Intelligence Analytics

Business Intelligence Operations

- **Critical IoC Watchlist:** High-severity threats requiring immediate attention
- **Daily Threat Summary:** Aggregated threat counts by category and severity
- **IoC Trending:** Identify emerging threats and trending IoCs
- **Threat Actor Profiles:** Consolidated view of threat actor activities
- **CDC Change Tracking:** Track IoC modifications and threat evolution

8. Development of the Pipeline

This section details the implementation of each component in the CTI data pipeline.

8.1 Data Extraction Tool: Azure Data Factory

Pipeline Configuration

Component	Configuration
Source	HTTP REST connector to OTX API
Authentication	API Key in X-OTX-API-KEY header (stored in Key Vault)
Destination	ADLS Gen2: /mnt/cti/raw/otx/
Schedule	Every 15 minutes (tumbling window trigger)
Error Handling	3 retries with exponential backoff, failure alerts

Key Features

Feature	Implementation
Authentication	Managed Identity or Service Principal
Monitoring	Built-in monitoring with Azure Monitor integration

8.2 Data Processing Tool: Azure Databricks

Cluster Configuration

Setting	Value
Runtime	DBR 14.x LTS with Delta Lake 3.x
Node Type	Standard_DS3_v2 (4 cores, 14 GB RAM)
Auto-scaling	Enabled (2-8 workers)

Setting	Value
Auto-termination	15 minutes of inactivity
Storage Mount	ADLS Gen2 mounted to /mnt/cti

Notebook Execution Flow

1. **Bronze Layer:** Raw ingestion with CDC MERGE
2. **Silver Layer:** IoC normalization and threat categorization
3. **Gold Layer:** Analytics aggregations and watchlist creation

Jobs & Pipelines Execution

Databricks Jobs orchestrate the notebook execution with scheduled runs. The screenshot below shows the job execution history:

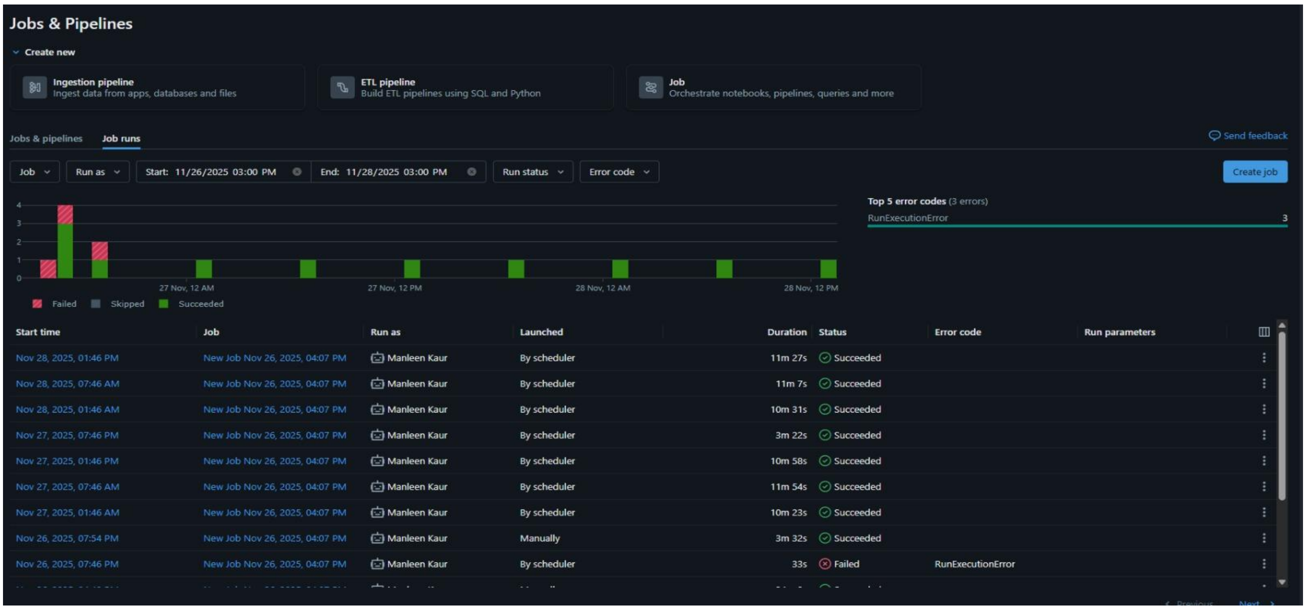

FIG-Databricks Jobs & Pipelines showing scheduled job runs with success/failure status

Job Execution Details:

- **Job Types:** Ingestion pipeline, ETL pipeline, Orchestration jobs
- **Scheduling:** Automated scheduler triggers (by scheduler)

- **Success Rate:** High success rate with green indicators, minimal failures
- **Duration:** Jobs complete in 3-11 minutes (3m 22s to 11m 54s)
- **Error Handling:** Failed jobs show RunExecutionError for troubleshooting

8.3 Interface to View Results

Results can be accessed and monitored through multiple interfaces:

- **Databricks SQL Warehouse:** Query Gold layer tables with real-time monitoring
- **Databricks Notebooks:** Interactive analysis using PySpark
- **Azure Logic Apps Email Alerts:** Automated email notifications for critical threats
- **SQL Endpoint:** External integration for SIEM/SOAR systems

SQL Warehouse Monitoring

The Databricks SQL Warehouse provides real-time query monitoring and performance metrics. The screenshot below shows the monitoring dashboard with query execution history:

FIG-SQL Warehouse CTI monitoring dashboard showing query history and cluster activity

Key Metrics Visible:

- **Query Execution:** SELECT statements from dbw_cti_processing.gold.ev_range_analysis_subset
- **Performance:** Query durations (7-9 seconds), fetch times (59-115 ms)
- **Source:** Azure-logic-apps integration
- **User Activity:** Automated queries from kaurmanleen93

Email Alert Implementation

Azure Logic Apps are configured to monitor the pipeline and send email alerts through Azure Monitor. The screenshots below show the actual email alerts received:

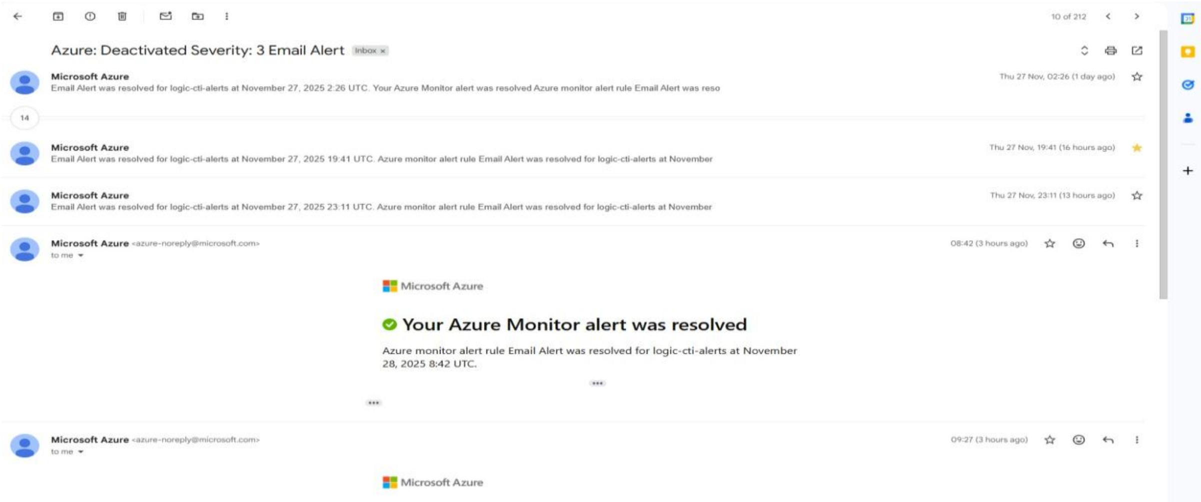

Figure 4: Azure Monitor email alerts inbox showing multiple alert notifications

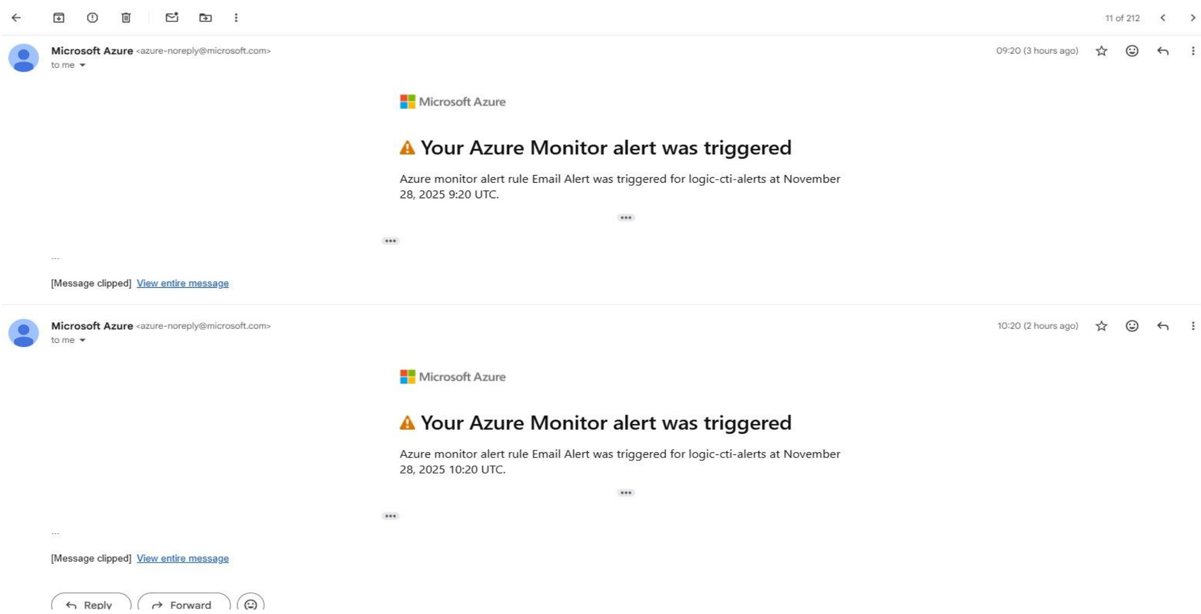

FIG-Detailed alert showing 'Azure Monitor alert was triggered' for logic-cti-alerts

Alert Configuration Details

Parameter	Configuration
Alert Rule Name	Email Alert (logic-cti-alerts)
Trigger Condition	Pipeline completion, critical threats detected, data quality issues
Notification Channel	Email via Azure Monitor
Severity Levels	Informational (resolved), Warning (triggered)

Power BI dashboard and Microsoft Teams alerts were not implemented in this version. The primary monitoring interface is email-based alerting via Azure Logic Apps and Azure Monitor as shown in the screenshots above.

9. Challenges and Lessons Learned

9.1 Technical Challenges

Challenge 1: Delta Lake CDC Implementation

Problem: Implementing MERGE operation with proper CDC tracking for IoCs required understanding Delta Lake internals. Initial attempts resulted in duplicate records when threat pulses were updated.

Solution: Implemented timestamp-based MERGE condition (source.modified > target.modified) to update only newer data. Enabled Change Data Feed to track all INSERT and UPDATE operations for comprehensive audit trail.

Challenge 2: IoC Deduplication Logic

Problem: Same IoCs appeared in multiple threat pulses from different sources, causing data duplication in Silver layer. Need to identify unique IoCs while preserving context from all sources.

Solution: Created composite deduplication key using hash of (indicator_type + indicator_value_clean). Aggregated pulse IDs into arrays to maintain linkage to all source pulses while storing unique IoCs.

Challenge 3: API Rate Limiting and Scheduling

Problem: OTX API has rate limits. Initial 5-minute polling schedule hit rate limits and caused pipeline failures. Need to balance freshness vs. API constraints.

Solution: Adjusted ADF pipeline to 15-minute intervals with exponential backoff retry logic. Implemented error handling and alerting for rate limit violations via Azure Monitor email alerts. This ensures reliable ingestion while respecting API limits.

9.2 Lessons Learned

- **CDC is Essential to Threat Intelligence:** Changes in IoC with time were found to be very valuable in the evolution of threats and the maintenance of proper watchlists.
- **Test with Real Data Early:** Delays in starting test with actual threat feeds led to rework. Immediate testing of Bronze layer using live data showed schema problems.
- **Deduplication Complexity:** IoC deduplication is more complicated than would be expected because it comes in different formats and contexts. Silver layer Standardization is necessary.
- **Medallion Architecture Value:** Bronze (raw), Silver (clean) and Gold (analytics) were separated well thus debugging and validation was much easier.
- **Follow API Costs:** API and cloud resources cost very fast. Scheduling and auto-termination is very important in cost control.
- **Incremental Approach:** The layer-by-layer approach and validation allowed eliminating compound errors and ensured that the troubleshooting was reasonable. Logic Apps email alerts were used to provide instant feedback on the health of pipelines.
- **Delta Lake Benefits:** ACID transactions and time-travel were valuable in data reliability and debugging. These are the characteristics that distinguish Delta Lake and common storage formats.

10. References

Documentation and Technical Resources:

- AlienVault OTX API Documentation: https://otx.alienvault.com/api
- Azure Data Factory Documentation: https://docs.microsoft.com/en-us/azure/data-factory/
- Azure Databricks Documentation: https://docs.microsoft.com/en-us/azure/databricks/
- Delta Lake Documentation: https://docs.delta.io/
- Delta Lake CDC Guide: https://docs.delta.io/latest/delta-change-data-feed.html
- Azure Logic Apps Documentation: https://docs.microsoft.com/en-us/azure/logic-apps/
- Azure Monitor Alerts: https://docs.microsoft.com/en-us/azure/azure-monitor/alerts/

Online Guides and Tutorials:

- Databricks Academy - Data Engineering with Databricks
- Microsoft Learn - Azure Data Engineer Learning Path
- Medallion Architecture Best Practices: https://www.databricks.com/glossary/medallion-architecture
- Threat Intelligence Sharing Guide: MISP Project Documentation

Academic Papers:

- [1] A. Aleksiyants, O. Borisenko, D. Turdakov, A. Sher, and S. Kuznetsov, “Implementing Apache Spark jobs execution and Apache Spark cluster creation for Openstack Sahara,” Proceedings of the Institute for System Programming of RAS, vol. 27, no. 5, pp. 35–48, 2015, doi: https://doi.org/10.15514/ispras-2015-27(5)-3.
- [1] P. Nainar Balasubramanian, “ML-Driven Threat Detection with Azure Security Center,” 2025, doi: https://doi.org/10.2139/ssrn.5393916.