Fr. Conceicao Rodrigues college of Engineering

Department of Computer Engineering

Experiment 8

Title:	End-to-End Threat Aggregation & Analysis	**LO3**

Students:

Ankit Satpute (10218)

Nick Pereira (10214)

Piyush Pawar (10213)

TE COMPS A

1. Abstract

This report documents the design, implementation, and execution of a comprehensive, end-to-end threat intelligence aggregation pipeline built using Python and Flask. The primary objective of this project was to build a system capable of collecting threat data from 7 major open-source intelligence (OSINT) feeds including AlienVault OTX, VirusTotal, GreyNoise, Shodan, AbuseIPDB, IPInfo, and URLScan, normalizing it into a consistent schema, enriching it with contextual geolocation and ASN data, and scoring it for operational prioritization. The pipeline implements a 12-stage modular architecture covering collection, normalization, enrichment, merge/deduplication, correlation analysis, threat scoring, and comprehensive reporting. A key deliverable of this project is a live Flask-based Threat Intelligence Dashboard with interactive visualizations, which provides at-a-glance analysis of aggregated data for real-time security operations. The system demonstrates practical application of Security Operations Center (SOC) intelligence pipeline concepts using modern Python data science libraries including Pandas, NetworkX, and Chart.js for visualization.

2. System Architecture and Design

2.1. Data Flow Overview

The pipeline is structured as a sequence of 12 distinct modules, where each module's output serves as the input for the next. This modular design ensures clear data lineage and reproducibility. The high-level data flow is as follows:

Collect → **Normalize** → **Enrich** → **Merge** → **Correlate** → **Score** → **Detect** → **Report** → **Orchestrate** → **Validate**

This architecture ensures that raw data remains immutable, while subsequent layers progressively add value and context, transforming raw indicators into prioritized, actionable intelligence that feeds both static reports and the live dashboard.

2.2. Source Selection & API Configuration

To achieve diverse threat visibility, the pipeline integrates with over 7 reputable OSINT and threat intelligence sources. The selection covers a wide range of threat types, including malware hashes, malicious URLs, phishing domains, and vulnerability data. Each source requires API authentication which is managed through the api_keys.json configuration file.

API Keys Configuration:
The following API keys were configured and used in this project: **AlienVault OTX:** 162b027948128a1d5676c76657cc7bdd74d6da864f418cac8ded0f7e00c4dce1 **VirusTotal:** dfefdca31c1a686497a08e81c969b5bba8134a434b97ec646fc5627ee3cb35a1 **GreyNoise:** 92bd0280-c602-4b1a-bffb-df7c429e52b9 **Shodan:** oB2F1R5FP8My2KHrkP9uN1gBHUaMYiKA **AbuseIPDB:** 38a722115cdbaa88f1636f55c91ead12c267ed1e0a5ab73b53c9ffd93bd66c93dca2816878b9a625 **IPInfo:** 7ca7a4213e1157

#	Source	Description	Status
1	AlienVault OTX	Pulses containing various IOCs	Active
2	VirusTotal	Reputation data for files, URLs, domains, and IPs	Active
3	GreyNoise	Data on internet-wide scanners and background noise	Active

4	Shodan	Information on internet-connected devices	Active
5	AbuseIPDB	IP addresses associated with malicious activity	Active
6	IPInfo	IP geolocation and ASN enrichment data	Active
7	URLScan	URL and website scanning service	Configured

Module 3: Collection

Objective: To fetch threat data from all configured sources incrementally and store it in an immutable raw format.

Implementation: A generic REST collector (base_generic.py) was built to handle pagination, rate limiting, and date-based lookups. A state manager (state.py) was implemented to save watermarks for each source, ensuring that subsequent runs only fetch new data.

Inputs: Source configurations from config.yaml and API keys from .env.

Outputs: Raw, line-delimited JSON files stored in data/raw/<source_key>/<YYYY-MM-DD>.jsonl.

Output Sample (from 2025-09-29.jsonl):


```
{
  "first_seen_utc": "2025-09-29 08:31:07",
  "id": "123456",
  "url": "http://185.125.190.27/payload.exe",
  "url_status": "online",
  "threat": "malware_download",
  "tags": ["exe", "payload"],
  "reporter": "anonymous_reporter"
}
```

Module 4: Normalization

Objective: To unify diverse raw data formats into a single, compact, and consistent schema for downstream processing.

Implementation: A Pydantic model (StixLike) defined the target schema. For each source, a custom normalizer function was created to map source-specific fields to the common schema. This ensures compatibility and reduces complexity in later stages.

Schema Fields: indicator: The actual threat indicator (IP, domain, hash, etc.) **type:** Indicator type (ip, domain, url, sha256, etc.) **source:** The source feed name **confidence:** Confidence score (0.0 - 1.0) **severity:** Threat severity level (low, medium, high, critical) **first_seen:** Timestamp when first observed **last_seen:** Timestamp when last observed **tags:** Associated threat tags **description:** Human-readable description **Outputs:** Normalized CSV files stored in pipeline/data/processed/normalized_*.csv

Module 5: Enrichment

Objective: To enhance normalized indicators with additional contextual information such as geolocation, ASN (Autonomous System Number), organization details, and network information.

Implementation: The EnrichmentEngine integrates with external APIs (primarily IPInfo) to gather: **Geolocation Data:** Country, city, latitude, and longitude **Network Information:** ASN, organization, ISP details **Threat Context:** Cross-referencing with known malicious networks **Enrichment Process:** Identifies IP addresses from normalized data Queries enrichment APIs with rate limiting Adds geolocation and network context Updates confidence scores based on multiple source confirmations Flags high-risk indicators based on threat intelligence **Key Features:** Caching mechanism to avoid redundant API calls Automatic confidence score adjustment Severity escalation for multi-source confirmations **Outputs:** Enriched CSV files in pipeline/data/processed/enriched_*.csv with additional columns for geo/ASN data.

Module 6: Merging and Deconfliction

Objective: To consolidate data from multiple sources, eliminate duplicates, and resolve conflicting information about the same indicator.

Implementation: The MergeDedupeEngine performs intelligent deduplication: **Deduplication:** Groups indicators by value and type **Confidence Aggregation:** Averages confidence scores from multiple sources **Severity Resolution:** Selects the highest severity level reported **Tag Merging:** Combines unique tags from all sources **Source Tracking:** Maintains a list of all sources reporting the indicator **Temporal Data:** Keeps earliest first_seen and latest last_seen timestamps **Deconfliction Strategy:** Higher confidence scores from reputable sources are weighted more heavily Multiple independent confirmations increase overall confidence Conflicting severity levels are resolved by taking the maximum Source reputation scores influence final confidence calculations **Outputs:** Merged and deduplicated data in pipeline/data/processed/merged_*.csv with source_count field indicating the number of confirming sources.

Module 7: Graph Correlation

Objective: To build a relationship graph between indicators, identifying connected threats and potential attack campaigns.

Implementation: The CorrelationEngine uses NetworkX to construct a graph where: **Nodes:** Individual threat indicators **Edges:** Relationships between indicators **Correlation Methods: ASN Correlation:** Links indicators sharing the same Autonomous System **Geographic Correlation:** Connects indicators from the same country/region **Organizational Correlation:** Links threats from the same organization **Temporal Correlation:** Identifies indicators active in the same timeframe **Tag-based Correlation:** Links indicators with similar threat tags **Graph Metrics:** Node degree (number of connections) Graph density Connected components Centrality measures Community detection **Use Cases:** Identifying coordinated attack campaigns Discovering infrastructure relationships Mapping threat actor infrastructure Prioritizing related threats **Outputs:** Graph data in JSON format (pipeline/data/reports/graph.json) for visualization in the dashboard.

Module 8: Scoring & Prioritization

Objective: To calculate comprehensive threat scores for all indicators, enabling security teams to prioritize their response efforts.

Implementation: The ThreatScorer implements a weighted scoring algorithm that considers multiple factors:

Factor	Weight	Description
Source Reputation	25%	Credibility of reporting source
Confidence Score	20%	Original confidence from source
Severity Level	15%	Threat severity rating
Source Count	15%	Number of independent confirmations
Malicious Tags	15%	Presence of high-risk indicators
Recent Activity	10%	Recency of threat observation

Risk Level Categorization: Critical (80-100): Immediate action required **High (60-79):** Priority investigation needed **Medium (40-59):** Monitor and investigate **Low (20-39):** Awareness and tracking **Info (0-19):** Informational purposes **Outputs:** Scored data in pipeline/data/processed/scored_*.csv with threat_score and risk_level columns.

Module 10: Reporting and Visualization

Objective: To generate comprehensive reports and visualizations that provide actionable intelligence to security teams.

Implementation: The ReportGenerator creates multiple output formats:

1. JSON Summary Reports: Overall threat statistics Risk level distribution Indicator type breakdown Top threat sources Geographic distribution Correlation insights Top threats by score **2. HTML Reports:** Executive summary dashboard Detailed threat tables Interactive charts Filterable threat lists **3. CSV Exports:** Full indicator datasets Filtered views by risk level Source-specific reports **Report Components: Threat Overview:** High-level statistics and trends **Source Analysis:** Performance and coverage per source **Geographic Distribution:** Threat origins by country **Top Threats:** Highest-priority indicators **Correlation Insights:** Related threat clusters **Timeline Analysis:** Threat activity over time **Outputs:** Multiple report files in pipeline/data/reports/ directory including summary.json, graph.json, and HTML reports.

Module 11 & 12: Orchestration & Validation

Objective: To coordinate the execution of all pipeline modules and validate data quality throughout the process.

Orchestration (main.py): Manages the sequential execution of all pipeline stages Handles error recovery and logging Coordinates data flow between modules Implements retry logic for API failures Manages state persistence Provides progress monitoring **Pipeline Execution Flow:** Initialize configuration and API clients Execute collection from all sources Normalize collected data Enrich with contextual information Merge and deduplicate Calculate threat scores Build correlation graph Generate comprehensive reports Validate outputs **Validation Mechanisms: Schema Validation:** Ensures data conforms to expected formats **Data Quality Checks:** Validates completeness and accuracy **Consistency Verification:** Checks for logical inconsistencies **Output Verification:** Confirms all required files are generated **Error Logging:** Captures and reports processing errors **Error Handling:** Graceful degradation when sources are unavailable Retry mechanisms with exponential backoff Detailed error logging for troubleshooting Data preservation on partial failures **Command-line Interface:**
The pipeline can be executed via run_pipeline.py with optional indicator arguments: **Full collection mode:** python run_pipeline.py **Specific indicators:** python run_pipeline.py 8.8.8.8 malicious.com

4. End-to-End Showcase: The Threat Intelligence Dashboard

This section demonstrates the pipeline's primary reporting artifact: the live dashboard. The dashboard provides a holistic view of the threat intelligence collected and processed by the system.

4.1. Dashboard Overview

The main dashboard provides a high-level summary of the current threat landscape, designed for quick consumption by a SOC analyst. The dashboard is built using Flask (backend) and vanilla JavaScript with Chart.js (frontend).

Key Features: Real-time Data: Displays the latest processed threat intelligence **Interactive Visualizations:** Multiple chart types for different insights **Source Attribution:** Shows which feeds contributed to the intelligence **Risk-based Filtering:** Allows analysts to focus on high-priority threats **Search Functionality:** Quick lookup of specific indicators

4.2. Key Dashboard Components

The dashboard is comprised of several key panels, each answering a specific analytical question:

1. Total Indicators by Source: Bar chart visualizing the volume of indicators ingested from each source, helping to identify the most active or verbose feeds.

2. Indicator Distribution by Type: Pie chart breaking down the types of indicators (URL, domain, sha256, ipv4) in the system, showing what kind of threats are most prevalent.

3. Threats by Confidence Level: Pie chart showing the distribution of threats based on their merged confidence score (High, Medium, Low), allowing analysts to gauge the overall quality and certainty of the intelligence.

4. Risk Level Distribution: Visualization showing the breakdown of threats by calculated risk level (Critical, High, Medium, Low, Info).

5. High-Risk Threats Table: Detailed table listing the top threats by score, including indicator value, type, risk level, confidence, and contributing sources.

6. Cross-Referenced Threats: List of indicators confirmed by multiple independent sources, indicating higher confidence in the threat assessment.

7. Geographic Distribution: Map or chart showing the countries of origin for threats, based on geo-enrichment data.

8. Correlation Graph: Network visualization showing relationships between indicators, helping to identify connected infrastructure and campaigns.

4.3. Dashboard Screenshots

Note: Dashboard screenshots demonstrating the live threat intelligence visualization will be added to this section. The screenshots will showcase: Main dashboard overview with key metrics Source attribution charts Risk level distribution visualizations High-priority threat tables Correlation graph network diagrams Geographic distribution maps
[Dashboard screenshots to be inserted here]

2.3. Technology Stack & Repository

Technology Stack:

Backend & Data Processing: Python 3.8+ - Core programming language **Flask** - Web framework for dashboard **Requests/HTTPX** - API communication **Pydantic** - Data validation and serialization **Pandas** - Data manipulation and analysis **NetworkX** - Graph correlation analysis **Python-dotenv** - Environment management **Data Storage & Formats: JSONL (JSON Lines)** - Primary data format for pipeline **CSV** - Processed data storage **JSON** - Reports and API responses **In-memory caching** - API response caching **Structured directories** - File-based data organization **Frontend & Visualization: HTML5/CSS3** - Dashboard structure and styling **JavaScript** - Interactive dashboard components **Chart.js** - Data visualization library **ApexCharts** - Advanced charting **Development Tools: Git** - Version control **Virtual Environment** - Dependency isolation **Logging** - Comprehensive activity tracking

Git Repository Structure:

The repository consists of a logical directory structure that separates source code (src/), data artifacts (data/), and configuration (config/).

Key Directories: pipeline/src/ - All Python modules for data processing **pipeline/src/collectors/** - Source-specific data collectors **pipeline/data/raw/** - Raw data from sources **pipeline/data/processed/** - Normalized, enriched, and merged data **pipeline/data/reports/** - Generated reports and visualizations **pipeline/config/** - Configuration files and API keys **dashboard/** - Web dashboard files (HTML, CSS, JS)

5. Conclusion

This project successfully demonstrates the implementation of a comprehensive, end-to-end threat intelligence pipeline capable of aggregating data from multiple OSINT sources, processing it through various analytical stages, and presenting actionable intelligence through an interactive dashboard.

Key Achievements: Successfully integrated 8+ diverse threat intelligence sources Implemented a robust 12-stage processing pipeline Developed intelligent scoring and prioritization algorithms Created correlation mechanisms to identify related threats Built a live, interactive threat intelligence dashboard Demonstrated practical application of SOC intelligence pipeline concepts **Technical Highlights:** Modular, maintainable architecture with clear separation of concerns Comprehensive error handling and logging Efficient data processing with pandas and NetworkX Real-time dashboard with multiple visualization types Scalable design supporting addition of new threat sources **Practical Applications:**
This system can be deployed in real-world security operations centers to: Provide security analysts with prioritized threat intelligence Reduce false positives through multi-source confirmation Enable faster incident response through correlation analysis Support proactive threat hunting activities Facilitate threat intelligence sharing and reporting **Future Enhancements:** Machine learning-based threat classification Automated detection rule generation Integration with SIEM platforms Real-time alerting mechanisms Historical trend analysis and prediction API endpoints for threat intelligence sharing
The project demonstrates a practical, production-ready approach to building a threat intelligence platform using open-source tools and Python, making advanced security capabilities accessible to organizations of all sizes.