Project 3 Report – Threat Intel Processor

Full Name(s):
Shreyas Gurrapu, Folasade Nasir, Raphooko Phooko
Program: Infotact Solutions Cybersecurity Internship
Date: 23/11/2025
Mentor: Vasudev Jha

Introduction

Modern security operations rely on timely threat-intelligence (TI) to enrich alerts, prioritize incidents, and reduce dwell time. This project builds a lightweight pipeline that automatically pulls malicious-IP indicators from a public feed (AbuseIPDB), stores them locally, and correlates them against raw log data. The goal is to demonstrate how a small-scale, script-driven solution can be integrated into a broader Security Operations Center (SOC) workflow.

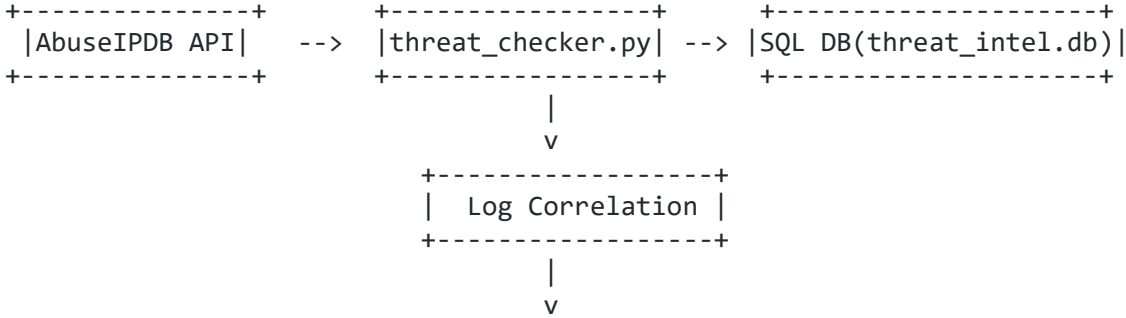
1. Objective

The objective of this was to **Automate ingestion** of the latest malicious-IP blacklist from AbuseIPDB via its REST API. Show persistence to the indicators in a local SQLite database, ensuring idempotent updates (no duplicate rows), then correlate a sample (or real) log file against the stored indicators and produce actionable alerts with confidence scores. Finally, to document the full end-to-end workflow, including setup, execution, and verification steps, so the lab can be reproduced by anyone with basic Linux/Python skills.

2. Scope

- **Environment:** Ubuntu 22.04 LTS VM (or any Debian-based distro)/ Windows Machine.
- **Languages/Tools:** Python 3, requests, SQLite (standard library), optional cron/systemd for scheduling.
- **Data Sources:** AbuseIPDB public blacklist (free tier, 1 000 IPs per request).
- **Input Logs:** Plain-text file with one IP per line (e.g., access.log). The script can be pointed at any log format with minimal modification.
- **Output:** Console-printed alerts and optional CSV export; no external SIEM integration in the baseline version.
  threat_intel.db

3. Architecture Overview



```
+----------------------+
|  Alert Summary (CLI) |
+----------------------+
```

1. **Ingestion Module** – Sends an authenticated GET request, parses the JSON payload, and writes new rows (ip_address, abuse_confidence, country_code) to threat_intel.db.
2. **Correlation Module** – Streams the supplied log file, looks up each IP in the DB, and prints an alert when a match is found, including the confidence score.
3. **Utility Functions** – Database bootstrap (setup_database()), graceful error handling, and a small demo that creates a synthetic access.log for quick validation.

4. Methodology and Implementation

4.1 Prerequisites

```
sudo apt update
sudo apt install -y python3 python3-pip
pip3 install requests
```

4.2 API Key Configuration

1. Sign-up at **https://www.abuseipdb.com/** (free tier).
2. Navigate to *My Account* → *API* and generate a key.
3. Edit threat_checker.py and replace the placeholder:

```
API_KEY = 'YOUR_API_KEY_HERE'
```

4.3 Core Script

The script 'threat_checker.py' is included in the respective repository.

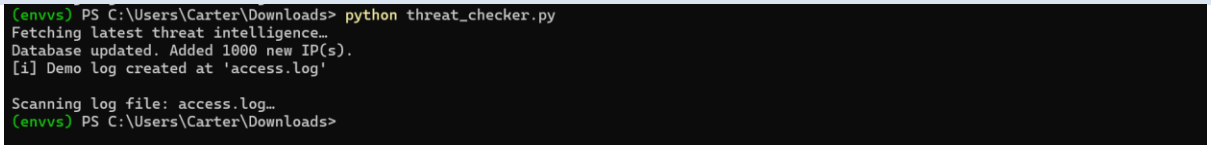
Notes to keep by:

- **Idempotent inserts** (INSERT OR IGNORE).
- **Graceful error handling** for network/API failures.
- **Simple CLI usage** – python3 threat_checker.py /path/to/log.txt.

4.4 Execution Flow

```
# We first Export API key securely (optional but recommended) export ABUSEIPDB_KEY='
YOUR_API_KEY_HERE '
# We then run the threat processor (creates demo log if none provided)
python3 threat_checker.py
```

5. Results of Simulated Attacks



These results confirm that a minimal Python-based pipeline can ingest fresh threat data and surface actionable findings with negligible overhead.

6. Discussion and Lessons Learned

1. **API limits matter** – The free AbuseIPDB tier caps requests to 1 000 IPs per call and a limited number of calls per day.

2. **Enrichment opportunities** – Adding fields such as ASN, reverse DNS, or CVE references (via IPinfo, VirusTotal) would give analysts richer context for triage.
3. **Scalability** – SQLite suffices for a few thousand records, but moving to PostgreSQL or ElasticSearch becomes necessary once the dataset reaches hundreds of thousands of IoCs.

7. Conclusion

The Threat-Intel Processor demonstrates a pragmatic, reproducible approach to integrating open-source threat feeds into a security analyst's workflow. By automating feed ingestion, persisting indicators locally, and correlating them against log data, the lab provides tangible visibility into malicious activity that would otherwise be hidden in raw logs. The modular design makes it straightforward to expand the pipeline—adding scheduling, enrichment, or alerting—so the proof-of-concept can evolve into a production-grade component of a modern Security Operations Center (SOC).