Here's a forward-leaning, practical design for an **initial dataset pipeline** that pulls from these open sources to power your Incident Response MVP. This pipeline will ingest, normalize, enrich, and store data to showcase detection, triage, and automated response capabilities.

Incident Response MVP Data Pipeline Architecture

1. Data Ingestion Layer

This layer fetches raw data from diverse open sources, using APIs, scheduled downloads, or web scraping where necessary.

Source	Data Type	Ingestion Method	Update Frequency
MITRE ATT&CK Framework	JSON threat TTP definitions	Direct API or JSON downloads	Weekly/On update
CISA Known Exploited Vulnerabilities	CSV/JSON vulnerability lists	Scheduled downloads from CISA website	Weekly
National Vulnerability Database (NVD)	CVE vulnerability metadata	NVD API or bulk JSON/XML download	Daily/Weekly
OpenPhish Threat Intelligence	Phishing URLs	API fetch or RSS feed	Hourly/Daily
Malware Traffic Analysis	PCAP files, network logs	HTTP download from public repositories	Ad hoc
AlienVault OTX	Indicators of Compromise (IOCs)	OTX REST API	Hourly
PhishTank	Phishing URLs	API or bulk data download	Daily
US-CERT Public Advisories	Incident reports & advisories	RSS feeds or HTTP scraping	Daily
Open Source SIEM Logs (e.g., CICIDS 2017)	Event logs	Bulk download	One-time or periodic

---

2. Data Normalization & Parsing Layer

- Use **ETL scripts** (Python recommended) to parse each data format into a **common schema**, e.g.,
  - `timestamp`
  - `indicator_type` (IP, URL, file hash, CVE, TTP)
  - `indicator_value`
  - `source`
  - `threat_category`

    - `severity_score` (CVSS or derived)
    - `description`
    - `related_assets` (if any)
- Normalize timestamps to UTC, clean malformed entries, and unify naming conventions.
- For structured sources like MITRE ATT&CK, extract technique IDs, tactics, and example procedures.

---

3. Data Enrichment Layer

- Cross-reference indicators with multiple sources for validation (e.g., an IP flagged both by AlienVault and OpenPhish).
- Pull CVSS scores from NVD for vulnerabilities and map them to CISA vulnerabilities.
- Tag phishing URLs with reputation scores and historical occurrence data.
- Annotate MITRE TTPs with severity and likelihood based on recent incident reports.

---

4. Storage Layer

- Store normalized and enriched data in a **scalable database** optimized for fast queries. Options:
    - **NoSQL DB** (e.g., MongoDB, Elasticsearch) for flexible, schema-less data storage and powerful search capabilities
    - **Relational DB** (e.g., PostgreSQL) with JSON support for structured query and reporting
- Implement **data versioning and retention policies** to simulate incident timelines.

---

5. API & Access Layer

- Develop a RESTful API to expose normalized threat intelligence data for your Incident Response app.
- API endpoints could include:
    - `/indicators` with filters by type, severity, date
    - `/incidents` simulating ongoing or historical events
    - `/attack_techniques` returning MITRE TTP data
    - `/phishing_urls` for real-time phishing alerts

---

6. Integration with Incident Response MVP

- The app ingests data via API and uses AI models to:

- Prioritize incidents based on severity and context
- Match incoming logs or alerts to known TTPs from MITRE ATT&CK
- Automate alert generation and remediation playbook triggers based on enriched threat intel
- Support dashboards showing live threat landscape and incident status

---

7. Automation & Scheduling

- Use **cron jobs or cloud schedulers** to automate data fetch and processing pipelines at defined intervals.
- Incorporate monitoring/logging to track data freshness, pipeline health, and alert on ingestion failures.

Example Tech Stack for MVP Pipeline

Component	Suggested Tool/Library
Data Ingestion	Python (requests, urllib), Airflow for scheduling
Parsing & ETL	Python (pandas, json, xml.etree.ElementTree)
Storage	Elasticsearch or MongoDB (NoSQL) or PostgreSQL (SQL)
API Layer	Flask/FastAPI (Python)
Scheduling	Cron (Linux), Airflow, or AWS Lambda + CloudWatch Events
Monitoring	Prometheus + Grafana, or built-in cloud monitoring tools

---

Next Steps to Implementation

- Build a **prototype ETL script** for one or two sources (e.g., MITRE ATT&CK + CISA vulnerabilities) to validate normalization approach.
- Set up storage and a simple API exposing normalized data.
- Develop initial AI/ML models consuming this data to demonstrate incident triage and prioritization.
- Create demo dashboards visualizing threat indicators mapped to agency-relevant metrics.

This pipeline approach is scalable and modular—perfect for a startup or mid-size federal contractor looking to demonstrate a polished MVP with real-world data. It aligns with best practices for data-driven cybersecurity apps and positions you to quickly iterate and expand as agency needs evolve.