SOAR Developer Async Test

Goal

Build a tiny SOAR pipeline from scratch that:

1. Ingests an alert (JSON file input)
2. Enriches IOCs using local mock TI data (no internet calls)
3. Triages with deterministic rules (severity, suppression, tagging, MITRE ATT&CK)
4. Responds by simulating device isolation (write to a log)
5. Outputs a normalized incident JSON and a Jinja2 Markdown analyst summary.

Deliverables

A public GitHub repository containing:

1. Source code
2. README.md (how to set up, run, and test)
3. Configs (YAML) for allowlists, MITRE map, connector bases
4. Local TI mocks (JSON files)
5. Two sample alerts (JSON files)
6. Output folders (JSON, .md and .log file)

Requirements

Ingestion

- CLI invocation (choose one):
    - python main.py alerts/sentinel.json
    - or make run ALERT=alerts/sentinel.json
- Your program reads a single alert JSON file and processes it end-to-end.

Normalization

- Convert source-specific alert fields into a normalized internal shape:
    - Keep original alert in incident.source_alert.
    - Normalize indicators into a list: [{ "type": "ipv4|domains|urls|sha256", "value": "…" }].
    - Keep asset.device_id, asset.hostname, asset.ip if present.

Enrichment

- For each IOC (indicator of compromise), read mock TI JSON files from disk (no network).
- Support three “providers”: Defender TI, ReversingLabs, Anomali.
- Produce a merged risk per indicator:
    - "risk": { "verdict": "malicious|suspicious|clean|unknown", "score": 0-100, "sources": ["defender_ti", ...] }

Triage (severity, suppression, tagging, MITRE)

- Base severity by alert.type (fallback Unknown=40):
    - Malware=70, Phishing=60, Beaconing=65, CredentialAccess=75, C2=80

- Intel boosts:
    - +20 if any IOC verdict == malicious
    - +10 if any IOC verdict == suspicious
    - +5 per extra flagged IOC (malicious/suspicious) beyond the first, cap +20

- Allowlist suppression (from YAML):
    - If an IOC is allowlisted: subtract 25 and add tag allowlisted
    - If all IOCs are allowlisted → severity=0, add tag suppressed=true, skip response

- Clamp and bucket:
    - Clamp to 0..100
    - Buckets: 0=Suppressed, 1–39 Low, 40–69 Medium, 70–89 High, 90–100 Critical

- MITRE ATT&CK tagging:
    - Map alert types via YAML to techniques (e.g., ["T1056", "T1555"]);
    - Use defaults if type not mapped.

Outputs

- If final severity ≥ 70 and asset.device_id present and not allowlisted:
    - Append a line to out/isolation.log:
        - <ISO-TS> isolate device_id=<ID> incident=<INCIDENT_ID> result=isolated
- Incident JSON → out/incidents/<incident_id>.json (must include):

```
JSON
{
```

```
  "incident_id": "...",
  "source_alert": { "...original..." },
  "asset": { "device_id": "...", "hostname": "...", "ip": "..." },
  "indicators": [
    { "type":"ipv4","value":"1.2.3.4","risk":{},"allowlisted": false }
  ],
  "triage": { "severity": 0-100, "bucket":"Low|Medium|High|Critical|Suppressed", "tags":
["..."], "suppressed": true|false },
  "mitre": { "techniques": ["T1059", ...] },
  "actions": [ { "type":"isolate","target":"device:<ID>","result":"isolated","ts":"..."} ],
  "timeline": [ { "stage":"ingest|enrich|triage|respond", "ts":"...", "details":"..." } ]
}
```

Analyst summary (Jinja2 → Markdown) → out/summaries/<incident_id>.md:

- Incident, indicators table, severity & tags, ATT&CK techniques, actions taken.

Samples and mock files:
https://drive.google.com/drive/u/0/folders/1hPPQ4Php_3YdA4_vlDksD7qHS90lQTVZ