Integration Guide: Log Pipeline (Vector → Wazuh)

1. Architecture Overview

The data flow has been optimized for high performance by consolidating data cleaning and **ECS (Elastic Common Schema)** formatting into a single transformation step, bypassing unnecessary enrichment stages. The following configuration is specifically for the datasets used for the test.

Note: If you want to bypass the SLP Enrichment component, you will need to do the following steps before you deploy de platform or before you execute the scritp init.sh.

Important: Vector dependencies

Go to Vector docker-compose.yaml and comment or delete the dependencies with enrichment:

```
  depends_on:
    resilmesh-ap-silentpush:
      condition: service_healthy
      restart: true
```

Before you deploy the platform, go to Docker-compose/docker-compose-Full-platform.yaml or the one you are going to use for the deployment and comment the enrichment line:

```
include:
  - ./Vector/docker-compose.yaml
  - ./NATS/docker-compose.yaml
 # - ./Enrichment/docker-compose.yaml
  - ./MISP_client/docker-compose.yaml
```

2. Vector Configuration

A. Dynamic Normalization (4_csv_normalizer.yaml)

This step identifies the dataset type based on the column count and sanitizes the raw CSV strings (removing extra quotes and whitespace).

Transforms/ 4_csv_normalizer.yaml:

```
type: "remap"

inputs: ["3_deduped_csv_events"]

source: |

 if starts_with(string!(.message), "#") || strlen(string!(.message)) == 0 { abort }

 row, err = parse_csv(string!(.message))

 if err == null {

  col_count = length(row)

  if col_count == 6 {

   # Dataset 1

   .source_ip = strip_whitespace!(replace!(string!(row[7]), "\"", ""))

   .attack_category = "Reputation List"

   .attack_name = "Known Malicious IP"

   .destination_ip = "0.0.0.0"

  } else if col_count >= 11 {

   # Dataset 2

   .last_time = strip_whitespace!(replace!(string!(row[3]), "\"", ""))

   .attack_category = strip_whitespace!(replace!(string!(row[4]), "\"", ""))

   .source_ip = strip_whitespace!(replace!(string!(row[7]), "\"", ""))

   .destination_ip = strip_whitespace!(replace!(string!(row[9]), "\"", ""))

   .attack_name = strip_whitespace!(replace!(string!(row[11]), "\"", ""))

  }
```

B. ECS Formatting & Cleanup (7_format_ecs)

This transforms the cleaned data into the hierarchical ECS structure and purges technical metadata to optimize network traffic.

Transforms/7_format_ecs.yaml:

```
type: "remap"

inputs: ["6_normalized_events", "0_ad_events"]

source: |

 message = parse_json!(.message)

 .@timestamp = now()

 .ecs.version = "1.6.0"

 .threat.enrichments.indicator.last_seen = message.last_time

 .threat.group.name = message.attack_category

 .source.ip = message.source_ip

 .destination.ip = message.destination_ip

 .threat.enrichments.indicator.reference = message.attack_name

 del(.source_type)

 del(.subject)

 del(.message)

 del(.timestamp))
```

C. Output

Direct TCP connection to the Wazuh Manager's rsyslog port.

Sinks/11_rsyslog:

```
 11_rsyslog:

type: "socket"

inputs: ["7_format_ecs"]
```

```
address: "${RSYSLOG_HOST}:10514"
mode: "tcp"
encoding:
  codec: "json"
```

---

3. Wazuh Configuration

A. Decoders (/var/ossec/etc/decoders/local_decoder.xml)

Enables Wazuh to break down the nested JSON object sent by Vector.

Create local_decoder.xml:

```
<decoder name="Resilmesh_decoders">
  <program_name>resilmesh-ap-vector.resilmesh_network</program_name>
</decoder>


<decoder name="Resilmesh_json">
  <parent>Resilmesh_decoders</parent>
  <plugin_decoder>JSON_Decoder</plugin_decoder>
</decoder>
```

B. Rules (/var/ossec/etc/rules/local_rules.xml)

Rules are updated to look for nested ECS field names (using dot notation).

Create local_rules.xml:

```
<group name="resilmesh,">
<rule id="100600" level="3">
<decoded_as>Resilmesh_decoders</decoded_as>
<description>Resilmesh: Event received</description>
</rule>
<rule id="100601" level="10">
```

```
<if_sid>100600</if_sid>

<field name="threat.group.name">^.+$</field>

<description>Resilmesh: Attack type $(threat.group.name) detected</description>

</rule>

<rule id="100602" level="5">

<if_sid>100600</if_sid>

<field name="source.ip">\.+</field>

<description>Resilmesh: Activity from IP $(source.ip)</description>

</rule>

</group>
```

---

4. Testing the flow:

Test 1: execute the following code in the server where CESNET's datasets are used.

```
$ docker exec -u 0 resilmesh-ap-vector bash -c 'tail -n50
/etc/vector/datasets/CESNET/bad_ips.csv >> /etc/vector/datasets/CESNET/bad_ips.csv'
```

Test 2: execute the following code in the server where UMU's datasets are used.

```
$ docker exec -u 0 resilmesh-ap-vector bash -c 'tail -n50 /etc/vector/datasets/UMU/NUSW-
NB15_GT.csv >> /etc/vector/datasets/UMU/NUSW-NB15_GT.csv'
```

---

5. Maintenance & Troubleshooting

1. **Monitor Raw Logs:** in wazuh manager, run tail -f /var/log/Resilmesh.log to ensure rsyslog is receiving data.
2. **Wazuh Logtest:** Always test new logs using /var/ossec/bin/wazuh-logtest. Ensure **Phase 3** correctly identifies the Rule ID and Level.