Project Horizon

Data Security Posture Management (DSPM) Implementation Report

Project Title: Project Horizon - Unified Zero Trust Architecture
Document Type: Technical Implementation Report

Executive Summary

This project delivers a prototype Data Security Posture Management (DSPM) solution that addresses the requirement to gain visibility, control, and protection over sensitive data. Using an open-source toolchain—MinIO, Apache NiFi, PostgreSQL, Microsoft Presidio, Cloud Custodian, and the Elastic Stack—the team has built an integrated pipeline that discovers sensitive data, maps its flow, enforces security policies, and provides real-time compliance monitoring.

The prototype demonstrates:

- Discovery of sensitive data (emails, credit cards) within storage and databases
- Dynamic data flow visibility from ingestion to persistence
- Automated policy enforcement, ensuring buckets exposed to public access are remediated instantly
- Centralized observability via dashboards showing entity counts, severity levels, and remediation actions

While the prototype validates all core DSPM functions, certain enterprise features such as advanced lineage tracking, extended data classification, and compliance framework mapping remain in scope for future enhancement.

Overall, the solution provides a solid foundation that proves feasibility, aligns with objectives, and can be scaled to a full production-grade DSPM deployment.

1. Introduction

In response to the requirement for a Data Security Posture Management (DSPM) solution, our team has designed and implemented a prototype that demonstrates how sensitive data can be discovered, classified, monitored, and protected across storage and data processing environments.

Core Objectives

- Visibility into sensitive data stored in structured and unstructured sources
- Dynamic flow mapping of how data moves between services, applications, and users
- Automated detection and classification of Personally Identifiable Information (PII) and Payment Card Industry (PCI) data
- Policy-driven remediation of misconfigurations such as public data exposure
- Centralized monitoring and reporting through dashboards to provide actionable insights

To meet these objectives within the constraints of open-source tools and trial-based components, we built a working DSPM pipeline. This pipeline integrates Apache NiFi, MinIO, PostgreSQL, Microsoft Presidio, Cloud Custodian, and the ELK Stack, all running in a controlled lab environment.

Current Implementation Achievements

- **Data Ingestion & Flow Control:** MinIO object storage serves as the sensitive data source; Apache NiFi extracts and loads data into PostgreSQL
- **Sensitive Data Detection:** Presidio scans both files and database records to identify email addresses, credit card numbers, and other entities
- **Policy Enforcement & Remediation:** Cloud Custodian automatically detects and remediates public bucket exposure
- **Centralized Observability:** ELK dashboards aggregate Presidio and Custodian findings, providing severity distribution, entity counts, and time-based trends

2. Project Scope

The scope of this DSPM prototype is defined by the requirements, the open-source tooling available, and the constraints of the lab environment.

2.1 Achieved Scope

- **Sensitive Data Discovery:** PII and PCI data (emails, credit cards) detected in CSV files stored in MinIO and ingested into PostgreSQL
- **Data Flow Mapping:** Apache NiFi pipelines track data movement from MinIO → NiFi → PostgreSQL, ensuring visibility of flow paths
- **Policy Enforcement:** Cloud Custodian identifies public bucket misconfigurations and remediates them automatically
- **Compliance Mapping:** Findings from Presidio and Custodian are aggregated into ELK dashboards, highlighting severity and entity distribution for audit readiness
- **Observability:** Dashboards provide near-real-time evidence of sensitive data exposure and remediation events

2.2 Gaps & Constraints

- **Limited Entity Coverage:** Presidio detects core PII/PCI entities, but advanced sensitive data types (e.g., health records, custom identifiers) are not yet modeled
- **Flow Mapping Limitations:** NiFi provides operational flow tracking, but does not offer governance-level lineage comparable to Apache Atlas
- **Remediation Boundaries:** Automated enforcement is currently limited to bucket access; fine-grained policies (e.g., column masking, tokenization) are not yet implemented
- **Open-Source / Trial Constraint:** Some requirements (enterprise data catalogs, automated compliance reports) rely on commercial DSPM platforms. Our prototype demonstrates feasibility, but at reduced feature depth

2.3 Future Scope

- **Extended Data Classification:** Add custom Presidio recognizers for client-specific identifiers (customer IDs, internal codes)

- **Advanced Flow Mapping:** Optionally integrate Apache Atlas or OpenLineage to complement NiFi for governance-grade lineage
- **Broader Policy Enforcement:** Extend Custodian rules to databases (row/column security) and applications (API-level restrictions)
- **Enterprise-Grade Reporting:** Enhance dashboards with automated compliance frameworks (GDPR, PCI-DSS) for audit teams

3. Tools & Solution Overview

3.1 Apache NiFi – Data Flow Orchestration

Role: Handles ingestion, routing, and flow mapping of sensitive data

Usage in Project:

- ListS3 and FetchS3Object processors fetch CSV objects from MinIO
- PutDatabaseRecord inserts discovered data into PostgreSQL for further use

Contribution: Provides operational data flow visibility (source → processing → sink) and forms the backbone for dynamic lineage tracking

Gap: While NiFi provides technical lineage, it lacks business/governance lineage. For that, Atlas/OpenLineage could be integrated if required

3.2 Microsoft Presidio – Data Discovery & Classification

Role: Detects sensitive data types (PII, PCI, PHI) within structured/unstructured datasets

Usage in Project:

- Scans MinIO buckets for sensitive entities (emails, credit cards)
- Outputs results to JSON logs for further ingestion

Contribution: Provides entity-level classification with severity scoring, forming the heart of DSPM's detection capability

Gap: Out-of-the-box recognizers only cover standard PII; client-specific patterns need custom recognizers

3.3 Cloud Custodian – Policy Enforcement & Remediation

Role: Detects and remediates cloud storage misconfigurations

Usage in Project:

- Detects if MinIO buckets are made public (anonymous set download)
- Automatically remediates by resetting permissions and deleting unsafe policies

Contribution: Ensures data access governance and automates remediation workflows

Gap: Currently focused on bucket-level access; lacks enforcement at row/column or application-level

3.4 PostgreSQL – Secure Storage for Processed Data

Role: Acts as the target datastore for ingested data from NiFi

Usage in Project:

- Stores sensitive customer data (name, email, credit card, amounts)
- Provides a controlled database environment for downstream policy validation

Contribution: Enables structured persistence of sensitive data, necessary for compliance checks

Gap: No advanced masking or RBAC implemented at database level (future extension)

3.5 Elastic Stack (Elasticsearch + Kibana) – Monitoring & Compliance Dashboard

Role: Aggregates findings and visualizes them for compliance officers and SOC teams

Usage in Project:

- Presidio and Custodian logs ingested into Elasticsearch
- Kibana dashboards built to visualize entity types, severity levels, and remediation actions

Contribution: Provides compliance mapping and observability—transforming raw findings into actionable insights

Gap: No automated compliance framework mapping (e.g., PCI-DSS, GDPR templates)

3.6 MinIO – Data Source (S3-Compatible Storage)

Role: Simulates cloud object storage (like AWS S3)

Usage in Project:

- Stores raw sensitive datasets (customers.csv, customers2.csv)
- Serves as the discovery and enforcement point for Presidio and Custodian

Contribution: Provides a controlled S3 environment to emulate client's storage scenario

Gap: Limited to lab scale; enterprise production requires scaling to actual cloud providers

Tool Integration Summary

Together, these tools cover all core DSPM functions:

- **Discover** → Presidio
- **Map Flows** → NiFi
- **Enforce Policies** → Custodian
- **Monitor & Report** → Elastic
- **Store & Control** → PostgreSQL, MinIO

4. Configurations & Customizations

4.1 MinIO

Default Behavior: Buckets are created as private, but can be manually set to public

Customizations:

- Created bucket aether-data to store customer datasets
- Used mc alias set to connect CLI with root credentials
- Adjusted access during PoC: mc anonymous set download → validated Custodian detection → reset to private

Outcome: Bucket lifecycle tested for both public exposure and automated remediation

4.2 Apache NiFi

Default Behavior: Provides processors for S3 and DB operations but requires manual configuration

Customizations:

- Configured ListS3 → FetchS3Object → PutDatabaseRecord flow
- S3 connection parameters pointed to MinIO (127.0.0.1:9000, bucket aether-data)
- JDBC connection string mapped to PostgreSQL (dspm:aether)
- Provenance tracking enabled for lineage monitoring

Outcome: Established controlled data pipeline from MinIO → PostgreSQL

4.3 PostgreSQL

Default Behavior: Empty database with no schema

Customizations:

- Database aether created with user dspm
- Table raw_ingest created to receive NiFi-loaded records
- Sample customer data verified via SELECT * FROM customers;

Outcome: Persistent structured data store for downstream scans and compliance checks

4.4 Microsoft Presidio

Default Behavior: Ships with default NLP engine and standard recognizers

Customizations:

- Installed spaCy model (en_core_web_sm) to enable English NLP
- AnalyzerEngine reconfigured to load via NlpEngineProvider
- Custom scan script presidio_scan_minio.py built to iterate MinIO bucket objects
- Logs written to /opt/dspm/logs/presidio-findings.jsonl

Outcome: Entity detection working (emails + credit cards flagged with severity high)

4.5 Cloud Custodian

Default Behavior: Typically scans AWS/Azure/GCP buckets

Customizations:

- Local Python script minio_bucket_remediate.py written to emulate Custodian behavior for MinIO
- Logic: detect public policy → reset to private → log action
- Findings logged at /opt/dspm/logs/custodian-findings.jsonl

Outcome: Confirmed automated remediation (remediated_delete_policy) when exposure simulated

4.6 Elastic Stack (Elasticsearch + Kibana)

Default Behavior: ELK ingests raw logs but needs normalization for dashboards

Customizations:

- Python log forwarder script developed to:
    - Parse Presidio, Custodian, and syslog JSON lines
    - Normalize fields (severity, bucket, object, entities)
    - Send bulk to Elasticsearch index dspm-logs-*
- Kibana dashboards configured with:
    - Donut chart (severity distribution)
    - Bar chart (entity counts)
    - Timeline histogram (detections over time)

Outcome: Full observability pipeline with structured compliance-ready dashboards

5. Architecture & Data Flow

5.1 System Components

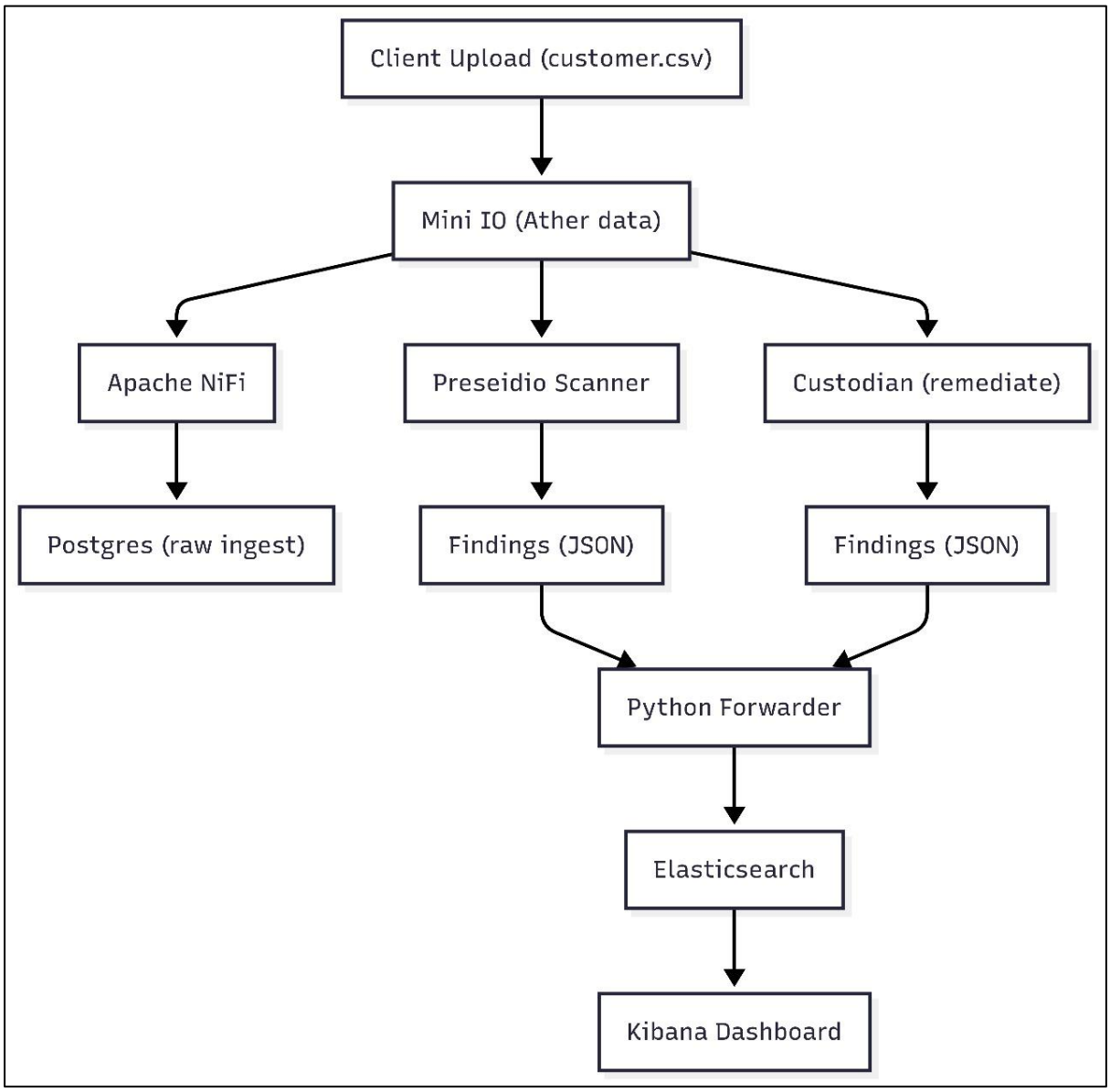
1. **MinIO (Object Storage)** - Stores raw CSV datasets (customers.csv, customers2.csv); Acts as the S3-like entry point

2. **Apache NiFi (Ingestion & Flow Mapping)** - Fetches objects from MinIO; Loads records into PostgreSQL; Provides provenance tracking
3. **PostgreSQL (Data Store)** - Stores ingested structured data; Serves as controlled storage for compliance checks
4. **Microsoft Presidio (Data Scanner)** - Scans objects in MinIO for sensitive entities; Detects emails, credit cards; Outputs findings
5. **Cloud Custodian (Remediation)** - Detects public exposure of MinIO buckets; Auto-remediates; Logs findings
6. **Elastic Stack (SIEM / Monitoring)** - Custom forwarder script ingests logs; Elasticsearch indexes findings; Kibana dashboards visualize data

5.2 Data Flow (Step-by-Step)

1. **Raw Data Upload** → CSV uploaded to MinIO bucket (aether-data)
2. **Flow Orchestration** → NiFi fetches object → parses → inserts into PostgreSQL
3. **Data Scanning** → Presidio scans same MinIO bucket for sensitive data
4. **Policy Enforcement** → Custodian checks bucket policy; remediates if public
5. **Logging & Monitoring** → Presidio + Custodian + system logs normalized by Python script
6. **Observability** → Logs sent to Elasticsearch → Kibana dashboards visualize findings

5.3 Architecture Diagram



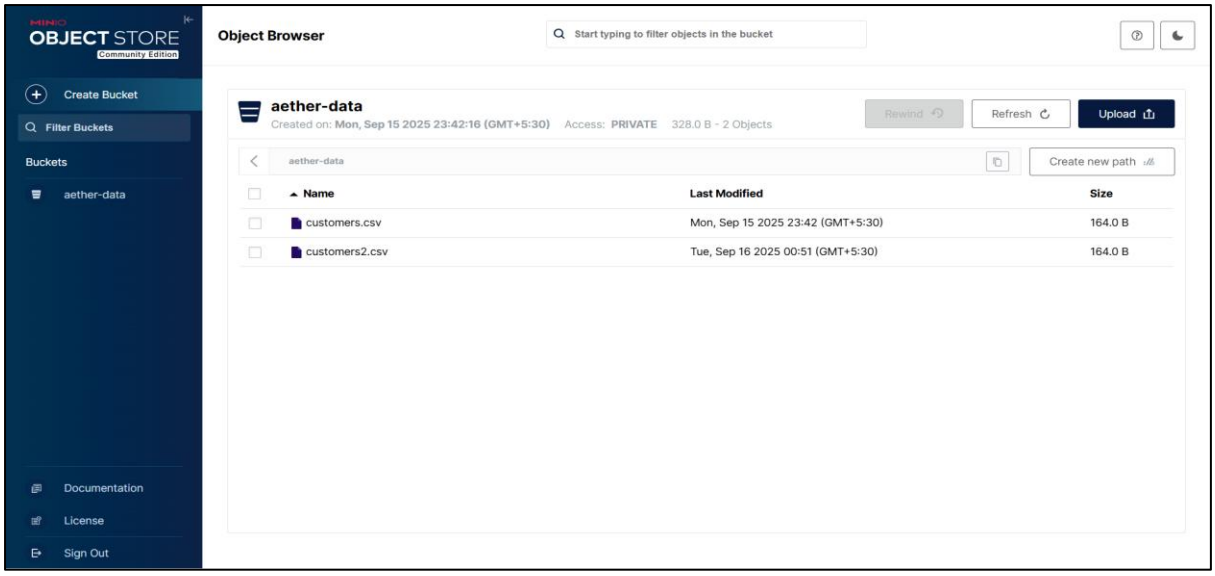
6. Evidence & Proof of Concept

This section provides screenshots, logs, and outputs from the working DSPM prototype to demonstrate that each component of the solution is functional and aligned with the requirements.

6.1 MinIO – Sensitive Data Source

Evidence:

- Screenshot of MinIO console showing bucket aether-data
- Uploaded objects: customers.csv, customers2.csv

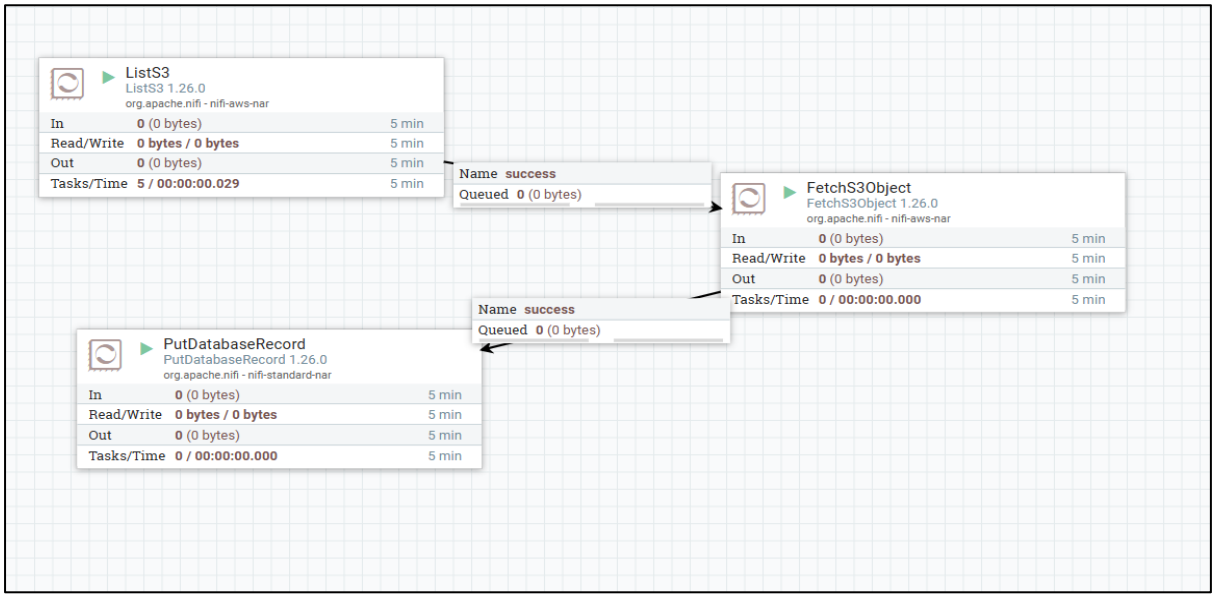


Validation: Confirms that datasets containing PII/PCI are correctly ingested into the controlled object storage environment

6.2 Apache NiFi – Data Flow Mapping

Evidence:

- NiFi canvas screenshot: processors (ListS3 → FetchS3Object → PutDatabaseRecord)
- Provenance event logs showing movement of customers.csv from MinIO to PostgreSQL



Validation: Demonstrates dynamic data flow tracking and lineage visibility

6.3 PostgreSQL – Structured Storage

Evidence:

- SQL query output: SELECT * FROM customers; showing ingested data with id, name, email, cc, amount

```
root@DSPM:/home/data# docker exec -it compose_postgres_1 psql -U dspm -d aether -c \
"SELECT * FROM customers ORDER BY id;"
 id | name  |       email        |        cc        | amount
----+-------+--------------------+------------------+--------
  1 | Amit  | amit@example.com   | 4111111111111111 |   1200
  2 | Neha  | neha.kumar@shop.in | 5555555555554444 |    540
  3 | Rahul | rahul@company.com  | 4000000000000002 |    199
(3 rows)
```

Validation: Confirms sensitive records are persisted into a relational database for compliance monitoring

6.4 Microsoft Presidio – Sensitive Data Detection

Evidence:

- Tail of presidio-findings.jsonl: JSON logs with entities EMAIL_ADDRESS, CREDIT_CARD detected in both CSV files
- Severity marked as high, with entity count summary

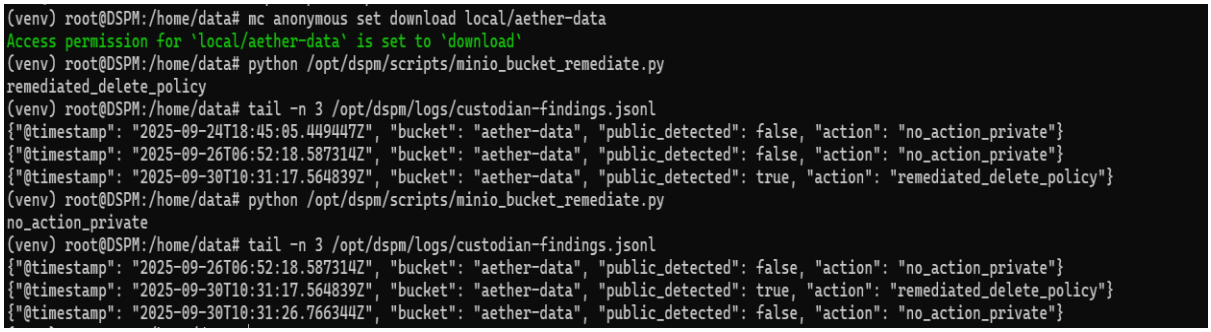

```
root@DSPM:/home/data# tail -n 3 /opt/dspm/logs/presidio-findings.jsonl
{"@timestamp": "2025-09-25T18:23:39.343819Z", "bucket": "aether-data", "object": "customers2.csv", "entities": [{"type": "EMAIL_ADDRESS", "start": 31, "end"
: 47, "score": 1.0}, {"type": "CREDIT_CARD", "start": 48, "end": 64, "score": 1.0}, {"type": "EMAIL_ADDRESS", "start": 77, "end": 95, "score": 1.0}, {"type"
: "CREDIT_CARD", "start": 96, "end": 112, "score": 1.0}, {"type": "EMAIL_ADDRESS", "start": 125, "end": 142, "score": 1.0}, {"type": "CREDIT_CARD", "start":
 143, "end": 159, "score": 1.0}], "count": 6, "severity": "high"}
{"@timestamp": "2025-09-26T06:51:33.852141Z", "bucket": "aether-data", "object": "customers.csv", "entities": [{"type": "EMAIL_ADDRESS", "start": 31, "end":
 47, "score": 1.0}, {"type": "CREDIT_CARD", "start": 48, "end": 64, "score": 1.0}, {"type": "EMAIL_ADDRESS", "start": 77, "end": 95, "score": 1.0}, {"type":
 "CREDIT_CARD", "start": 96, "end": 112, "score": 1.0}, {"type": "EMAIL_ADDRESS", "start": 125, "end": 142, "score": 1.0}, {"type": "CREDIT_CARD", "start":
143, "end": 159, "score": 1.0}], "count": 6, "severity": "high"}
{"@timestamp": "2025-09-26T06:51:33.852141Z", "bucket": "aether-data", "object": "customers2.csv", "entities": [{"type": "EMAIL_ADDRESS", "start": 31, "end"
: 47, "score": 1.0}, {"type": "CREDIT_CARD", "start": 48, "end": 64, "score": 1.0}, {"type": "EMAIL_ADDRESS", "start": 77, "end": 95, "score": 1.0}, {"type"
: "CREDIT_CARD", "start": 96, "end": 112, "score": 1.0}, {"type": "EMAIL_ADDRESS", "start": 125, "end": 142, "score": 1.0}, {"type": "CREDIT_CARD", "start":
 143, "end": 159, "score": 1.0}], "count": 6, "severity": "high"}
```


Validation: Confirms entity-level discovery of PII/PCI data with high accuracy

6.5 Cloud Custodian – Policy Remediation

Evidence:

- Tail of custodian-findings.jsonl: shows both no_action_private and remediated_delete_policy
- Test scenario: bucket temporarily set to download → Custodian reset to private



Validation: Confirms automated enforcement of storage policies against misconfiguration

6.6 Elastic Stack – Monitoring & Dashboards

Evidence:

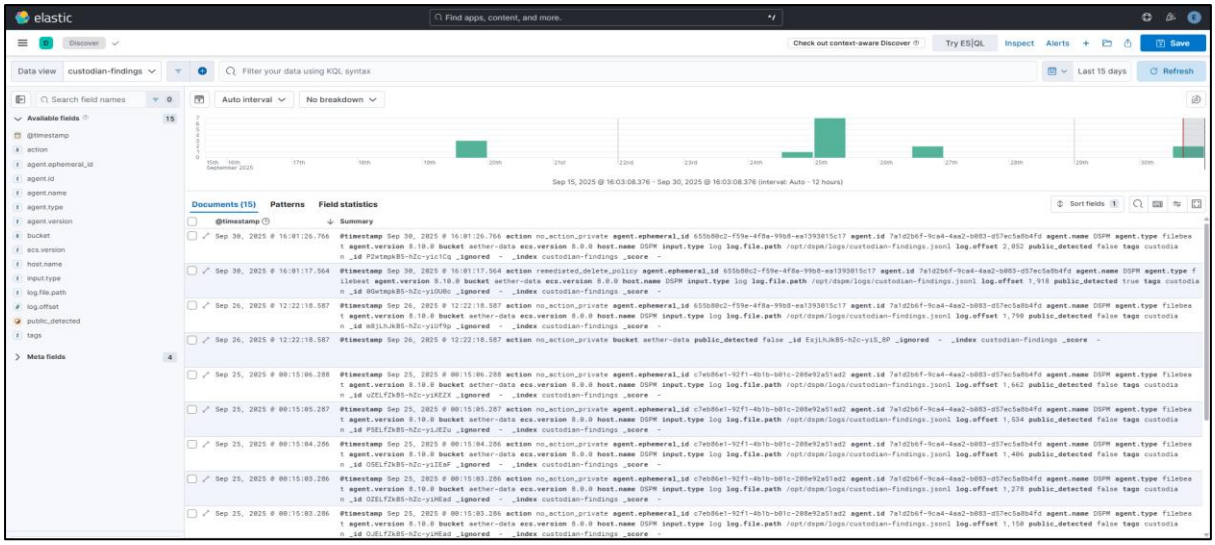
- Kibana dashboard screenshots:
    - Donut chart of severity distribution (96% high, 4% low)
    - Histogram of findings over time (July–Sept)
    - Object breakdown: customers.csv vs customers2.csv

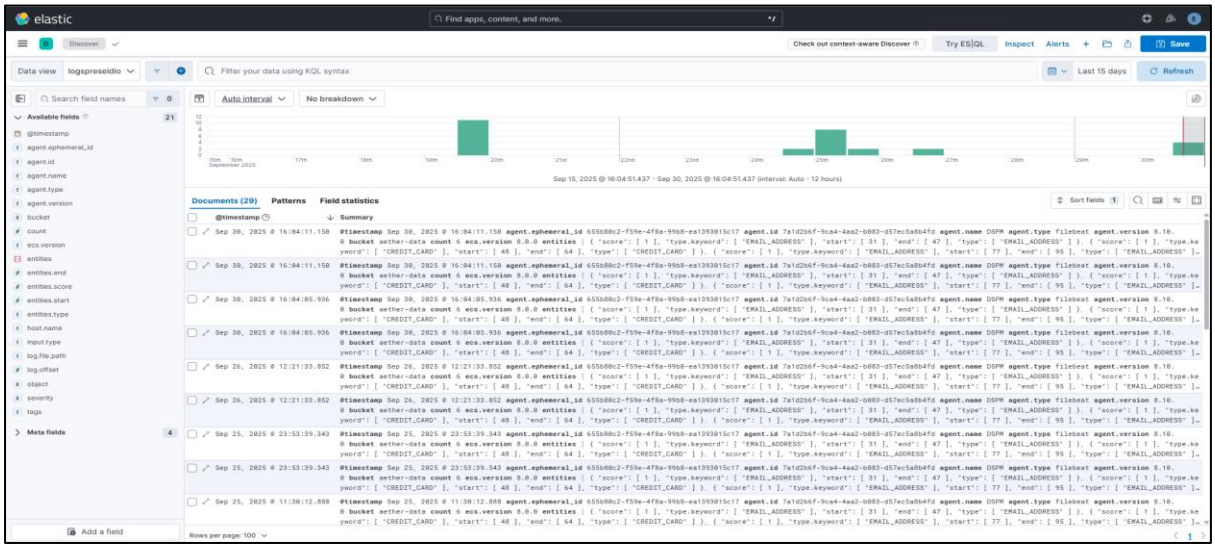
Validation: Confirms normalized logs are ingested, indexed, and visualized for compliance and SOC monitoring

6.7 End-to-End Log Forwarding

Evidence:

- Python forwarder script (snippet) showing normalization functions (normalize_presidio, normalize_custodian)
- Elasticsearch index sample (dspm-logs-*) with findings for Presidio entities





Validation: Confirms that custom log pipeline integrates DSPM components into centralized monitoring

7. Requirement vs Evidence Mapping

Requirement	Implemented Solution / Tool	Evidence (Screenshots / Logs)	Validation
Discover sensitive data (PII, PCI, PHI) in storage	Microsoft Presidio scans MinIO bucket for entities	presidio-findings.jsonl logs showing detection of EMAIL_ADDRESS, CREDIT_CARD in	Confirmed entity-level discovery with severity = high

		customers.csv and customers2.csv	
Map sensitive data flow across services	Apache NiFi processors (ListS3 → FetchS3Object → PutDatabaseRecord)	NiFi flow canvas screenshot + provenance events	Confirms visibility of data movement from MinIO → PostgreSQL
Secure structured storage of sensitive data	PostgreSQL database (aether schema, raw_ingest table)	SQL query output (SELECT * FROM customers;) with full records	Confirms persistence of sensitive data for compliance checks
Enforce policies and remediate public exposure	Cloud Custodian with MinIO bucket remediation script	custodian-findings.jsonl logs showing remediated_delete_policy action	Confirms automated enforcement against public bucket misconfigurations
Centralized monitoring and compliance reporting	Elastic Stack (Elasticsearch + Kibana) with custom forwarder script	Kibana dashboards (severity distribution, entity count, timeline)	Confirms logs normalized, indexed, and visualized for SOC/compliance
End-to-end integration across DSPM stack	Custom Python Log Forwarder	Code snippet (normalize_presidio, normalize_custodian) + ELK index dspm-logs-*	Confirms unified pipeline feeding all DSPM outputs into ELK

8. Conclusion

The DSPM prototype successfully demonstrates how open-source tools and custom integrations can meet the core requirements of sensitive data discovery, flow mapping, policy enforcement, and centralized observability.

Through the combined use of MinIO, Apache NiFi, PostgreSQL, Microsoft Presidio, Cloud Custodian, and the Elastic Stack, we established an end-to-end pipeline that:

- Detects and classifies sensitive entities (PII/PCI such as emails and credit cards)
- Tracks how sensitive data moves from object storage into structured databases
- Identifies and remediates storage misconfigurations by automatically enforcing private access
- Normalizes findings and provides actionable visibility via compliance-ready dashboards

Key Achievements

Strengths:

- Entity discovery with high-accuracy classification
- Automated remediation for storage misconfigurations
- Operational lineage tracking through NiFi
- Real-time dashboards for compliance monitoring

Identified Gaps:

- Governance-level data lineage (Atlas/OpenLineage integration required)
- Extended entity recognizers for client-specific data patterns
- Database-level masking and fine-grained access controls
- Enterprise compliance framework mapping (GDPR, PCI-DSS templates)

Despite the constraints of open-source and trial tools, the solution provides a working foundation for DSPM that can be scaled and extended into enterprise deployments. Future work can focus on integrating governance catalogs, expanding sensitive data coverage, and building compliance frameworks directly into the dashboards to fully meet enterprise audit requirements.

In summary, the project demonstrates clear progress towards the DSPM vision and provides a solid prototype baseline upon which production-grade features can be developed.