PII Detection and Redaction System

Using Microsoft Presidio

*Automated Text File Processing with Self-Improving Feedback Loop*

# Overview

This system automatically detects and redacts personally identifiable information (PII) from text files using Microsoft Presidio, an open-source data protection framework. The architecture uses a two-pass validation approach and continuously improves through automated feedback and model retraining.

## About Presidio

Presidio is a production-ready, open-source PII detection and anonymization SDK developed by Microsoft. It combines multiple detection techniques including pattern matching, named entity recognition, and machine learning models to identify and redact sensitive information.

## Key Capabilities

* Multi-layer detection using regex patterns, NER models, and custom recognizers
* Flexible anonymization strategies including masking, replacement, and encryption
* Configurable confidence thresholds for precision tuning
* Context-aware detection for improved accuracy
* Support for custom entity types and domain-specific patterns
* Model training and fine-tuning with custom datasets

# System Architecture

The system processes text files through multiple stages, validates redaction quality, and uses failures to improve detection accuracy over time.

## Processing Workflow

The complete system operates in stages, with Phase 1 providing core redaction and Phase 2 adding validation and continuous improvement.

**Phase 1: Primary Redaction**

1. **File Ingestion**

Raw text files are uploaded to S3 (Raw bucket). An EventBridge rule detects new uploads and triggers the redaction process via AWS Batch.

1. **Primary Detection and Redaction**

A Batch job (RedactPII) runs on Fargate and uses Presidio's AnalyzerEngine to detect PII with entity-specific confidence thresholds. Detected entities are redacted using AnonymizerEngine with deterministic HMAC-based tokens. Redacted files are stored in S3 (Redacted bucket).

**Phase 2: Validation and Feedback Loop**

1. **Validation Pass**

Redacted files are scanned by a second Presidio pass configured with lower confidence thresholds (0.4+) and additional recognizers. This validation pass catches residual PII that may have been missed. If PII is detected, findings are published to EventBridge (PII Leaks).

1. **Quarantine**

When validation detects PII leaks, EventBridge triggers a Batch job (CopyToQuarantine) that copies the original raw files to S3 (Quarantine bucket). An SNS alert notifies the team.

1. **Pattern Analysis**

Presidio analyzes quarantined files with detailed logging. Results are published to EventBridge (Quarantine) to trigger pattern extraction.

1. **Training Data Extraction**

A Batch job (ExtractPatterns) processes findings from quarantined files and generates structured training examples. These examples are written to S3 (Training Dataset bucket).

1. **Model Retraining**

Presidio's custom recognizers and NER models are periodically retrained using the training dataset. This closes the feedback loop and improves detection accuracy over time.

# Technical Configuration

## Two-Pass Strategy

The system uses the same Presidio SDK with different configurations for detection and validation.

### Primary Pass Configuration

* Confidence threshold: 0.7+ for high precision
* Core recognizers: Names, emails, phone numbers, SSNs, credit cards
* Anonymization: Replace with generic tokens

### Validation Pass Configuration

* Confidence threshold: 0.4+ for higher recall
* All available recognizers including experimental ones
* Context-aware detection enabled
* Additional regex patterns for structured PII

## Infrastructure

| **Component** | **Technology** |
| --- | --- |
| **Storage** | Amazon S3 (Raw, Redacted, Quarantine, Training Dataset buckets) |
| **Compute** | AWS Batch on Fargate (serverless container execution) |
| **Orchestration** | Amazon EventBridge (event-driven workflow triggers) |
| **Monitoring** | CloudWatch (logs and metrics), SNS (alerting) |
| **Container** | Docker with Presidio, spaCy models, and custom recognizers |

# Implementation Plan

**The implementation follows a two-phase approach. Phase 1 establishes the core redaction capability. Phase 2 adds validation and feedback loop for continuous improvement.**

## Phase 1: Primary Redaction (Weeks 1-2)

Focus on building a production-ready PII detection and redaction pipeline with security best practices.

### Week 1: Development and Testing

**Days 1-2: Setup**

* Install and configure Presidio locally
* Create Docker container with Presidio, spaCy models, and dependencies
* Implement structured logging framework with PII redaction
* Test with sample text files and tune recognizer settings

**Days 3-5: Primary Detection and Redaction**

* Build RedactPII Batch job with Presidio AnalyzerEngine and AnonymizerEngine
* Implement per-entity confidence thresholds (SSN: 0.99, Credit Card: 0.95, etc.)
* Implement deterministic anonymization using HMAC-based tokens
* Implement S3 integration for raw and redacted buckets
* Add comprehensive error handling and redacted logging
* Create unit tests and integration tests
* Test end-to-end with sample files of varying sizes

### Week 2: Security Hardening and Deployment

**Days 6-9: Infrastructure Deployment**

* Build and push Docker image to ECR
* Create AWS Batch job definitions and Fargate compute environments
* Configure EventBridge rules with versioned schemas for file upload events
* Set up CloudWatch dashboards with key metrics
* Configure SNS alerts for job failures and errors
* Deploy to staging environment

**Days 10-12: Testing and Production Launch**

* Run end-to-end tests in staging with production-scale data
* Perform security validation: verify VPC isolation, encryption, log redaction
* Load testing and performance optimization
* Deploy to production environment
* Document operational runbooks and troubleshooting procedures

### Phase 1 Deliverables

* Production-ready PII redaction pipeline
* Monitoring dashboards and alerting
* Operational documentation

### Phase 1 Success Criteria

* RedactPII Batch job executes successfully without errors
* PII detection meets baseline expectations with per-entity thresholds
* No PII leakage in CloudWatch logs (verified through log audits)
* Security controls verified: VPC-only access, separate KMS keys, encryption
* System handles production volume without performance issues
* Monitoring dashboards operational with alerting configured

## Phase 2: Validation and Feedback Loop (Weeks 3-4)

Add a validation layer to catch missed PII and implement an automated feedback loop for continuous improvement.

### Week 3: Validation Pass Development

**Days 1-2: Validation Pass Implementation**

* Build validation Batch job with lower confidence thresholds
* Configure all available Presidio recognizers for comprehensive detection
* Implement fail-closed logic to block file release when leaks detected
* Configure EventBridge integration for PII leak events
* Test validation pass against intentionally incomplete redactions

**Days 3-5: Quarantine and Pattern Extraction**

* Create Quarantine S3 bucket with separate KMS key
* Build CopyToQuarantine Batch job triggered by leak detection events
* Configure S3 lifecycle policies for Quarantine bucket (90-day TTL)
* Build ExtractPatterns job to analyze quarantined files
* Create Training Dataset S3 bucket and output format
* Test complete quarantine and extraction workflow

### Week 4: Model Retraining and Production Deployment

**Days 6-7: Model Retraining Pipeline**

* Develop model retraining script for Presidio custom recognizers
* Implement recognizer versioning system
* Create validation test suite for model quality checks
* Test retraining with sample training data

**Days 8-10: Integration and Deployment**

* Deploy validation components to staging
* Run end-to-end tests of complete feedback loop
* Update monitoring dashboards with validation metrics
* Deploy validation pass to production
* Monitor leakage rate and quarantine throughput
* Perform first manual model retraining with accumulated training data
* Document model retraining procedures

### Phase 2 Deliverables

* Validation layer with fail-closed protection
* Automated quarantine and pattern extraction
* Model retraining pipeline
* Self-improving feedback loop
* Enhanced monitoring for validation metrics

### Phase 2 Success Criteria

* Validation pass successfully identifies residual PII in redacted files
* Fail-closed logic blocks file release when leaks detected
* Quarantine workflow captures and processes failed redactions
* Pattern extraction generates usable training data
* Model retraining pipeline successfully improves detection accuracy
* Leakage rate metric shows measurable improvement over time