Microservice Requirement Document: Email Analysis

Version: 1.0
Date: 22-08-2025
Author: Aditya Pal

1. Introduction & Purpose

This document specifies the requirements for the **Email Analysis Microservice**. This service is a foundational component of the OSINT application, designed to uncover the digital footprint and potential vulnerabilities associated with an email address.

The primary purpose of this microservice is to take an email address as input and query various public and specialized sources to determine its validity, exposure in data breaches, associated domain details, and links to online accounts. The service will be built using **Python** and the **FastAPI** framework.

2. Functional Requirements

2.1. Supported Input Identifiers

The service will accept a single input type:

- **Email Address:** A syntactically valid email address (e.g., example@domain.com).

2.2. Target Data Sources & Analysis

The microservice must perform checks against the following sources:

- **Data Breach Databases:**
  - Integrate with the **Have I Been Pwned (HIBP)** API to check if the email address has appeared in any known public data breaches.
- **Social Media Account Existence:**
  - Programmatically check the "Forgot Password" or account sign-up pages of major platforms (Facebook, X, Instagram, LinkedIn) to determine if an account is registered with the email address, without triggering a password reset.
- **Domain Analysis:**
  - Perform a **WHOIS lookup** on the email's domain to retrieve registration details.
  - Check the domain's **MX (Mail Exchange) records** to verify it is configured to receive emails.
- **Gravatar Lookup:**
  - Check for a publicly associated Gravatar profile image linked to the email address.

2.3. Required Output Data Points

The service must aggregate all findings into a single, structured JSON response.

- email_address: The input email address.
- is_valid_format: A boolean indicating if the email syntax is correct.
- is_reachable: A status indicating if the domain's mail servers are reachable ("valid", "invalid", "unknown").
- data_breaches:
  - is_breached: A boolean indicating if the email was found in any breach.
  - breaches_count: The total number of breaches found.
  - breach_details: A list of objects, each containing the name of the breached site and the breach_date.
- social_accounts:
  - A list of objects, each containing the platform name (e.g., "Facebook") and an is_registered boolean.
- domain_details:
  - domain_name: The domain of the email.
  - whois_info: A JSON object containing key WHOIS data (e.g., registrar, creation_date, registrant_contact). This should be null for common free email providers (gmail.com, outlook.com, etc.).
- gravatar_profile:
  - has_gravatar: A boolean.
  - profile_image_url: The URL to the Gravatar image, if one exists.

3. API Endpoints Specification

The service will use the standard asynchronous task-based model.

3.1. POST /analyze

Initiates the analysis of an email address.

- **Method:** POST
- **Description:** Submits an email analysis task.
- **Request Body:**

```
{
  "email": "test.subject@example.com"
}
```

- Success Response (202 - Accepted):
  Returns a task_id.

```
{
  "message": "Email analysis task accepted.",
  "task_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234"
}
```

3.2. GET /results/{task_id}

Retrieves the results of the analysis.

- **Method:** GET
- **Response:** Follows the standard PENDING/COMPLETE status structure, with the data field containing the output specified in section 2.4 when the task is complete.

4. Non-Functional Requirements

- **Security:**
  - The API must be secured with an API key (X-API-KEY header).
  - All external API keys (e.g., for HIBP) must be stored securely and not exposed in the codebase.
  - Input must be sanitized to prevent command injection or other attacks.
- **Performance:**
  - The POST /analyze endpoint must respond in under **200ms**.
  - A full analysis should be completed in under 30 seconds.
- **Rate Limiting:**
  - The service must respect the rate limits of all third-party APIs it consumes (e.g., HIBP, WHOIS services). Implement appropriate backoff and retry logic.
- **Scalability:**
  - The service must be containerized (Docker) for easy deployment.
  - It should be capable of handling a high volume of concurrent analysis requests.
- **Logging:**
  - Log each request (without logging the email itself, to protect PII). Use the task_id for tracking.
  - Log errors from third-party APIs to monitor service health and integration issues.

5. Technology Stack

- **Language:** Python 3.9+
- **Framework:** FastAPI
- **Asynchronous Tasks:** Celery with Redis
- **HTTP Requests:** httpx
- **WHOIS Lookup:** python-whois library.
- **DNS Checks:** dnspython for MX record lookups.
- **Deployment:** Docker