Traff Group

MVP of the Cloak Service

Test Assignment

Author: Kyryl Kvas
Date: May 9, 2025

Contents

1 Introduction

Modern web services are constantly under threat from automated bots that scrape data, execute credential-stuffing attacks, or overload APIs with malicious traffic. The goal of our Cloak Service is to provide a simple yet logical filter pipeline that classifies each request as either “bot” or “not bot” via a RESTful API, thereby protecting downstream systems from unwanted automated access.

The original assignment reads:

> Коротко описати суть проблеми, яку вирішують даним інструментом. Наша "клоака"повинна приймати дані від користувача через RESTful API та повертати відповідь: "бот"чи "не бот". Суть не в кількості фільтрів, а в логіці їхньої роботи. MVP може бути простим, але ТЗ має показати, що кандидат розуміється на тому, що робить. Ось чим користуємося наразі ми, його документацію можна взяти як приклад https://vpnapi.io/. Наголосимо, не потрібно описувати ось прям все, суть роботи інструменту, авторизація нас не цікавить — як приклад модуля робота над яким не буде врахована.

This document defines the Minimal Viable Product (MVP) for the Cloak Service, focusing on the essential requirements, API contract, and filter logic without covering authentication or auxiliary modules. It demonstrates a clear understanding of RESTful design and decision-based filtering, matching the expectations of the Traff Group test assignment.

2 Requirements

2.1 Functional Requirements

- Single HTTPS endpoint: `POST /bot-detection/verdict`
  - Request body (JSON):
    ```
    {
      "headers": { ... }
    }
    ```
  - Response body (JSON):
    ```
    {
      "verdict": "bot" | "not_bot",
      "score": <float>
    }
    ```
    where `score` $\in [0, 1]$ represents a confidence level, with 0 being "not bot" and 1 being "bot". The score is calculated based on a series of filters applied to the request headers
- Deterministic verdict on each call.
- Persist each request and verdict in MongoDB:
  ```
  {
    "timestamp": "2025-05-07T12:34:56Z",
    "headers": { ... },
    "verdict": "bot",
    "score": <float>
  }
  ```
- Cache third-party lookups in MongoDB with TTL.

2.2 Non-Functional Requirements

- Observability: structured JSON logs.
- Tech Stack:
  - Docker
  - Node.js v22 (NestJS)
  - MongoDB

3 Filter Logic

The Cloak Service employs a layered approach to bot detection, where certain filters immediately flag a request as suspicious, assigning a verdict of "bot" with a score of 1, while others provide a more heuristic-based evaluation. This approach ensures both speed and accuracy in identifying automated traffic.
Note: Server-side detection alone is inherently limited—malicious clients can mimic legitimate requests, and HTTP headers can be forged or replayed. For high-assurance scenarios, additional client-side challenges (e.g., JavaScript puzzles or TLS fingerprinting) may be required.
Below is a detailed breakdown of the filter logic.

3.1 Filters

- **Header Integrity:** Bots often omit or falsify critical headers or supply implausible User-Agent (UA) strings. We perform a two-part analysis and then merge the results:

  1. *UA validity and pattern check:*

  $$S_{\text{UA}} = \begin{cases} 1 & \text{if UA is missing or matches patterns from } \texttt{crawler-user-agents} \text{ or } \texttt{user-agents}, \\ 0 & \text{otherwise} \end{cases}$$

  2. *Header presence and order:* Let $\mathcal{H} = \{\texttt{Host}, \texttt{Accept}, \texttt{Referer}, \texttt{Origin}, \texttt{Accept-Language}, \ldots\}$. For each $h \in \mathcal{H}$:

  $$H_h = \begin{cases} 1 & \text{if header } h \text{ is present and matches expected patterns} \\ 0 & \text{otherwise} \end{cases}$$

  Define the canonical order indicator:

  $$H_{\text{ord}} = \begin{cases} 1 & \text{if the headers appear in the typical browser order (allowing minor variations)} \\ 0 & \text{otherwise} \end{cases}$$

  Then combine into:

  $$H_c = \begin{cases} 0 & \exists\, h : H_h = 0 \ \vee \ H_{\text{ord}} = 0, \\ 1 & \text{otherwise.} \end{cases}$$

  Finally, merge both checks:

  $$H = \begin{cases} 0 & S_{\text{UA}} = 1 \ \vee \ H_{\text{c}} = 0, \\ 1 & \text{otherwise.} \end{cases}$$

  If $H = 0$, the request is immediately classified as **bot** (score = 1).

- **Request Frequency:** Bots tend to send requests much more frequently than legitimate users. If the number of requests from the same IP exceeds a predefined threshold within a short time window, it is flagged as a bot.

Let $N(t)$ represent the number of requests made within the time window of length t seconds. If $N(t) > T$, where T is the threshold, the request is classified as a "bot" with a score of 1:

$$N(t) > T \quad \Rightarrow \quad F = 1$$

In this case, the request is classified as a "bot" with a score of 1.

- **IP Reputation:** The IP reputation check is a heuristic filter based on the score from an external service. We use free-tier IP reputation services to assess the likelihood that the incoming request's IP address is associated with bot activity. The following external services are available for IP reputation checks:
  - `https://ipqualityscore.com/` - Provides IP reputation data, fraud scoring, and bot detection.
  - `https://www.abuseipdb.com/` - Offers risk scores based on reported malicious activity from IPs.
  - `https://www.virustotal.com/` - Checks IP reputation based on threat intelligence from multiple sources.

  Each of these services provides a reputation score R_{IP}, normalized between 0 and 1, where a higher score indicates a higher likelihood of the request originating from a bot. The first available service will be used to retrieve the reputation score for the incoming request's IP address. If the first service is unavailable or the response is invalid, we proceed to the next available service, and so on, until a valid score is obtained.

  However, if all the external services fail (e.g., due to network issues or API limits), we fall back to using a list of blocked IP addresses provided by the https://github.com/stamparm/ipsum/tree/master repository. The **ipsum** repository contains a list of IPs known to be associated with malicious activities. If the IP address matches any in this list (we can use child process for this), we consider it a bot.

3.2 Final Verdict

Once all filters are applied, the final verdict is determined based on the results of the checks. If any of the immediate red flag filters (Header Integrity, or Request Frequency) are triggered, the request is classified as "bot" with a score of 1. Otherwise, the IP reputation score is considered, and the request is classified as a bot if the IP reputation score exceeds the threshold of the external service or if the score is 1.

The final verdict can be expressed as:

$$\text{verdict} = \begin{cases} \text{bot} & \text{if } H = 0 \text{ or } F = 1 \text{ or } R_{IP} > T_{\text{service}} \\ \text{not\_bot} & \text{otherwise} \end{cases}$$

Where:
- H is the header integrity check result.
- F is the request frequency check result.
- R_{IP} is the reputation score from the external service, compared with T_{service}.