Traff Group

MVP of the Cloak Service

Test Assignment

Author: Kyryl Kvas
Date: May 7, 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 /detect`
  - 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, such as reputation checks and heuristics.

- 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
- Extensibility: filter pipeline is pluggable via dependency injection.