

Documentação da API Rest

Desafio 1

Gleidson Felipe Pereira da Silva

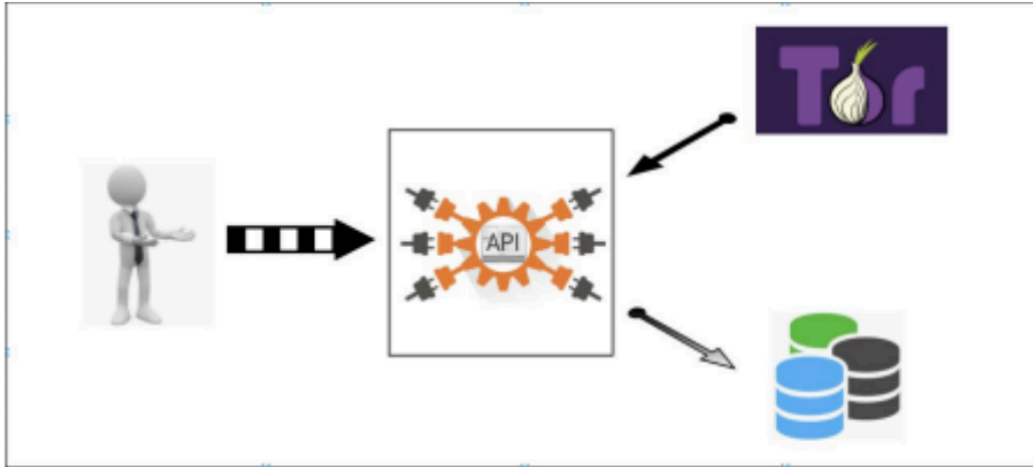
Novembro de 2024



Resumo Executivo

A motivação do desafio vislumbra que todos os membros do time de segurança cibernética tenham um conhecimento básico de Networking, infraestrutura, desenvolvimento e base de dados. Nesse sentido, o desafio proposto tem como objetivo desenvolver um sistema para obter endereços IPs maliciosos, a fim de bloquear o tráfego proveniente desses IPs.

1. Visão Geral



Foi desenvolvida uma API REST em Python que contempla as seguintes funcionalidades:

1. Um endpoint GET que devolve IPs a partir da fonte externa com a seguinte URL:
<https://check.torproject.org/torbulkexitlist>
2. Um endpoint POST que recebe um IP e adiciona este IP na base de dados (postgresql). Nesta base, estão os IPs que não queremos que apareçam no output do endpoint 3
3. Um endpoint GET que retorna os IPs da base de dados.
4. Um endpoint GET que retorna os IPs filtrados, ou seja, retornará a lista de IPs de saída do TOR (fonte externa) e excluindo os IPs adicionados no endpoint 2.
5. Um endpoint DELETE (extra) que faz a remoção de endereço IP.

2. Desenvolvimento

Para o desenvolvimento da aplicação foi usado a linguagem de programação python e o framework FastAPI. Este framework foi escolhido por permitir o desenvolvimento de APIs assíncronas e, consequentemente, ganho de performance nas requisições, bem como disponibilizar e facilitar uma documentação/swagger elegante e intuitiva.

2.1 Execução da Aplicação

Dependências:

- Docker
- Docker compose

Repositório privado da aplicação:

- git clone
https://C4B4N0:ghp_H5XxX8Z4lzlr9lkvfNc3PisO0p6sE61XIMhb@github.com/C4B4N0/API-REST-IPS-TOR.git

Como executar

- Copiar arquivo de env: **cp .env.sample .env**
- Inicie a aplicação e suas dependências: **docker compose up --build -d**
- Instalação inicial do banco de dados: **docker compose run app python -m database.init_db**

2.2 Capturas de telas

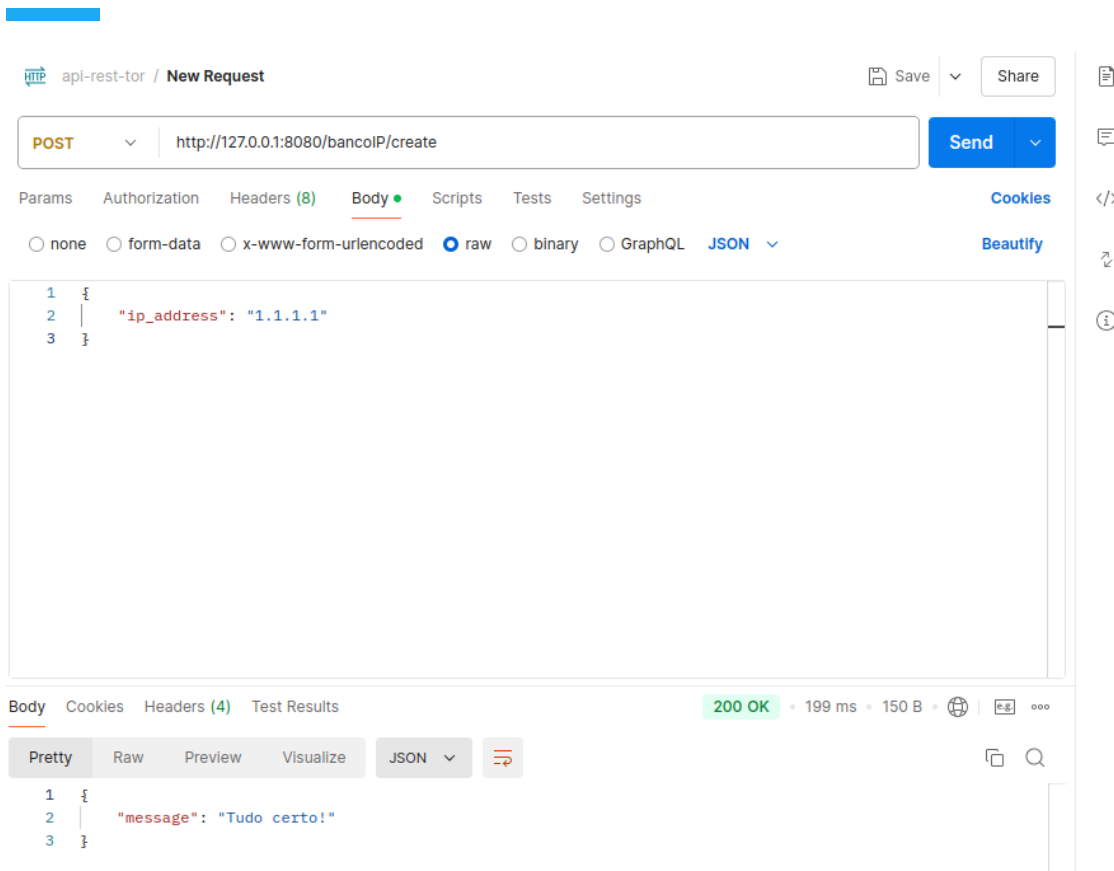
The screenshot displays the API REST client interface. At the top, the URL bar shows a GET request to `http://127.0.0.1:8080/tor-ips`. Below the URL bar, the 'Params' tab is active, showing a table with columns 'Key', 'Value', and 'Description'. The table is currently empty. The 'Body' tab is also visible. The response section at the bottom shows a status of '200 OK' with a response time of 1727 ms and a size of 18.65 KB. The response body is displayed in JSON format, showing a list of IP addresses under the key 'tor_ips'.

```
{  "tor_ips": [    "171.25.193.25",    "80.67.167.81",    "192.42.116.187",    "198.98.51.189",    "89.58.26.216",    "109.70.100.4",    "149.56.22.133",    "5.45.102.93",    "192.42.116.196",    "185.220.101.4",    "45.141.215.62",    "94.102.51.15",    "192.42.116.213",    "45.141.215.80",    "193.26.115.61",    "192.42.116.175",    "149.56.44.47",    "87.118.116.103",    "178.17.171.102",    "192.42.116.208",    "89.58.41.156",    "2.58.56.43",    "104.192.1.138",    "107.189.8.56"  ]}
```

Endpoint 1 - GET

Endpoint GET que devolve IPs a partir da fonte externa com a seguinte URL:

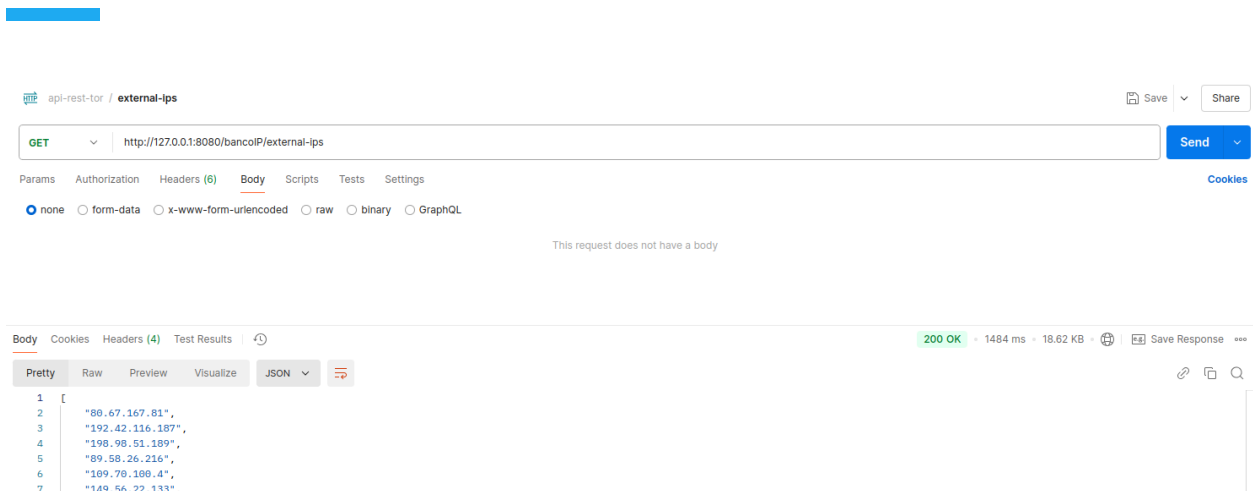
<https://check.torproject.org/torbulkexitlist>



Endpoint 2 - POST

Endpoint POST que recebe um IP e adiciona este IP na base de dados (postgresql). Nesta base, estão os IPs que não queremos que apareçam no output do endpoint 3.

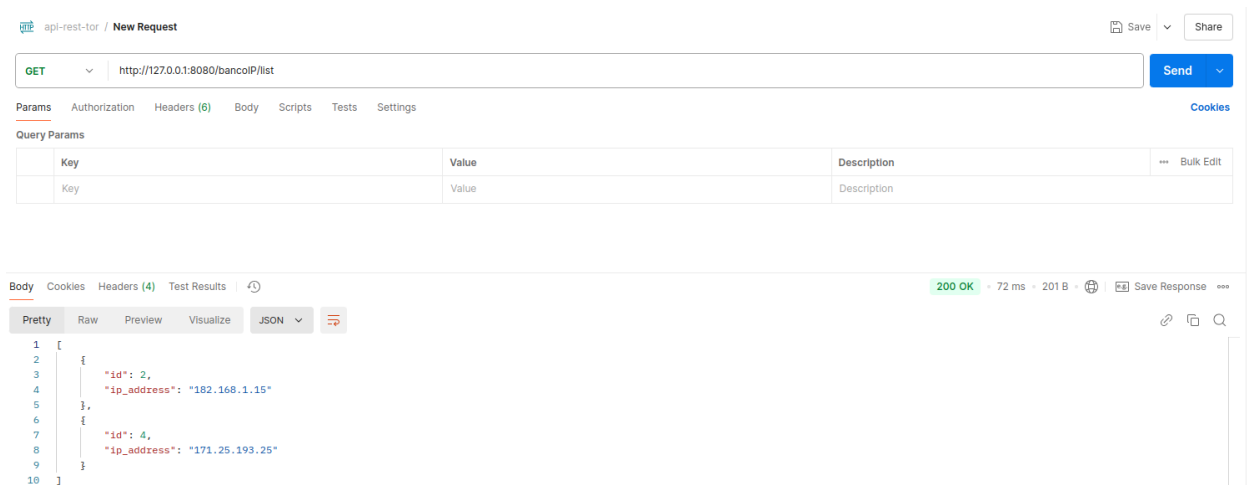
Observação: Não foi criado um script para popular a base de dados. O IP foi adicionado inserindo um JSON via ferramenta POSTMAN, conforme pode ser observado na imagem.



Endpoint 3 - GET

Endpoint GET que retorna os IPs filtrados, ou seja, retornará a lista de IPs de saída do TOR (fonte externa) e excluindo os IPs adicionados no endpoint 2.

Para testar este Endpoint, foi adicionado pelo endpoint 2 o primeiro IP retornado pelo endpoint 1.



Endpoint 4 - GET - Endpoint GET que retorna os IPs da base de dados



api-rest-tor / New Request

SaveShare

DELETE

http://127.0.0.1:8080/bancoIP/delete/3

Send

arams

Authorization

Headers (6)

Body

Scripts

Tests

Settings

Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

dy

Cookies

Headers (4)

Test Results

200 OK · 37 ms · 150 B · Save Response

Pretty

Raw

Preview

Visualize

JSON

1 {

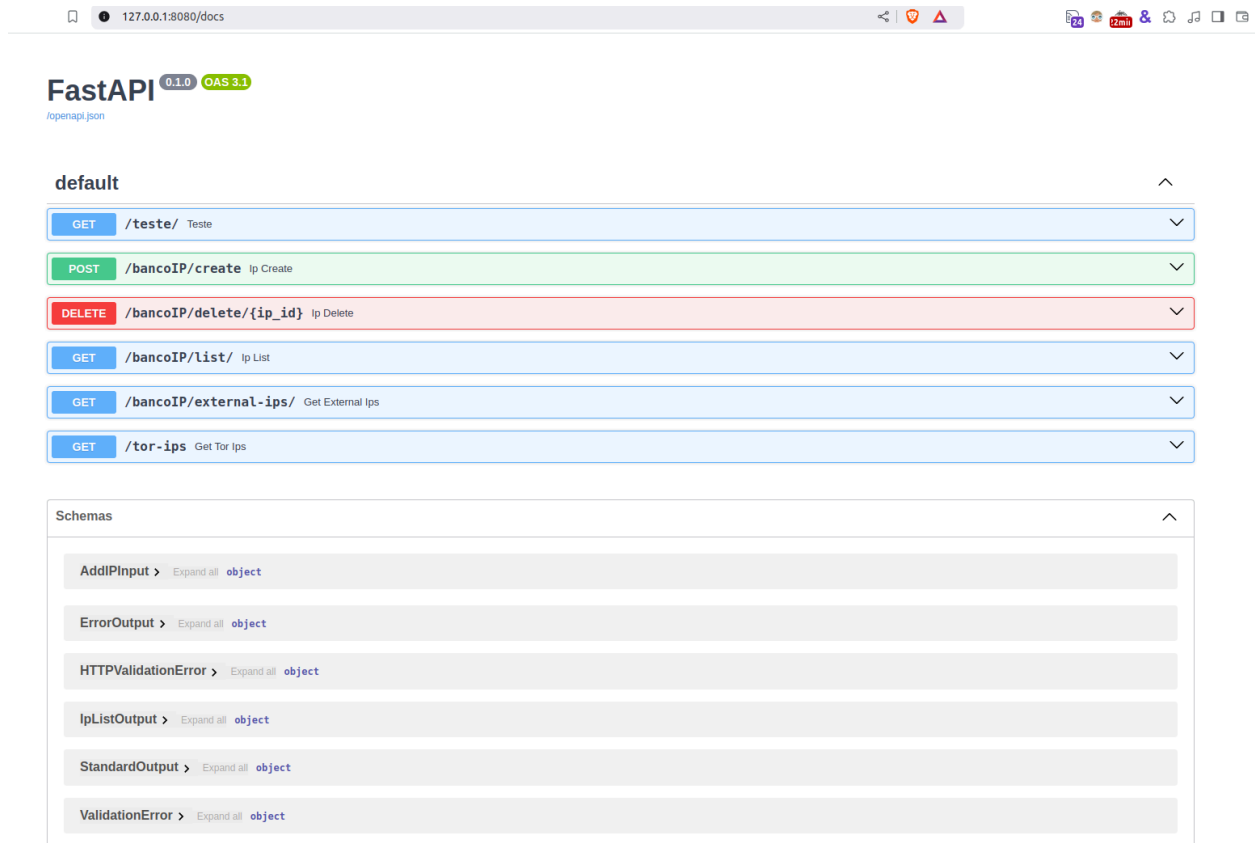
2 | "message": "Tudo certo!"

3 }

Endpoint 5 - DELETE - Endpoint extra que faz a remoção de IP

2.3 Acesso à API

- Interface Swagger: <http://127.0.0.1:4444/docs>
- Interface Swagger Redoc: <http://127.0.0.1:4444/redoc>



Swagger da aplicação

2.4 Deploy em Cloud Público

2.4.1 - Escolha da Cloud Pública: AWS EC2.

2.4.2 - Pré-requisitos:

- Chave SSH
- Ubuntu instalado e atualizado
- Docker e docker compose instalado

2.4.3 - Configuração do Workflow do GitHub Actions

Criar o arquivo `.github/workflows/actions.yml` no repositório com o seguinte conteúdo:

```
name: Deploy FastAPI Application
```

```
on:
```

```
  push:
```

```
    branches:
```

```
      - main
```

```
jobs:
```

```
  deploy:
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      # Checkout do código
```

```
      - name: Checkout code
```

```
        uses: actions/checkout@v3
```



```
# Configuração do ambiente
```

```
- name: Set up Python
```

```
uses: actions/setup-python@v4
```

```
with:
```

```
python-version: '3.9'
```

```
# Instalação de dependências
```

```
- name: Install dependencies
```

```
run: |
```

```
python -m pip install --upgrade pip
```

```
pip install pipenv
```

```
pipenv install
```

```
# Build da aplicação usando Docker
```

```
- name: Build Docker image
```

```
run: |
```

```
docker build -t fastapi-app .
```

```
# Login no servidor remoto via SSH
```

```
- name: Deploy to EC2
```

```
uses: appleboy/ssh-action@v0.1.10
```

```
with:
```

```
host: ${ secrets.EC2_HOST }
```

```
username: ubuntu
```

```
key: ${ secrets.EC2_SSH_KEY }
```

```
port: 22
```



```
script: |
```

```
docker stop fastapi-app || true
```

```
docker rm fastapi-app || true
```

```
docker run -d -p 80:80 --name fastapi-app fastapi-app
```

2.4.4 Configurar Secrets no Github

Adicionar as informações sensíveis no GitHub Action Secrets para segurança:

- EC2_HOST: Endereço IP do servidor EC2
- EC2_SSH_Key: Chave SSH privada usada para acessar o servidor

2.4.5 - Configuração do Servidor EC2

- Instalação no servidor remoto:

```
sudo apt update
```



```
sudo apt install -y docker.io
```



```
sudo systemctl start docker
```



```
sudo systemctl enable docker
```
- Configurar portas: Verificar se a porta 80 está aberta no Security Group do EC2.

2.4.6 Fluxo do Deploy

- Fazer um commit no branch main
- O GitHub Actions será acionado automaticamente
- A aplicação será empacotada em um container Docker e enviada para o servidor EC2
- O container será iniciado no servidor, expondo a API REST na porta 80

2.4.7 Teste do Deploy

Acessar o endereço público do servidor usando o comando curl.



2.5 Dificuldade encontrada

Devido ao fator tempo não foi possível desenvolver a aplicação de forma full assíncrona, bem como os entregáveis extras.