CENTRO UNIVERSITÁRIO DE BRASÍLIA
CIÊNCIA DA COMPUTAÇÃO

FELIPE BARCELOS DE CARVALHO (RA: 22350044)
MARCOS VINICIUS ROCHA (RA: 22352865)
JOÃO MARCELO GUIMARÃES DOURADO (RA: 22350653)
EDUARDO ARAÚJO UCHOA (RA: 22353207)
DAVI MAIA (RA: 22305561)

PROJECT AEGIS:
PAINEL DE INTELIGÊNCIA DE AMEAÇAS

BRASÍLIA
2025

PROJECT AEGIS: GUIA TÉCNICO DE EXECUÇÃO E MANIPULAÇÃO DE DADOS (ETL)

Versão: 1.0 (Release Candidate)

Repositório: https://github.com/vrsmarcos26/Project-Aegis

---

1. Visão Geral da Arquitetura

O Project Aegis utiliza uma arquitetura *Serverless* (sem servidor dedicado) para coleta, processamento e apresentação de dados de inteligência de ameaças. O pipeline de dados (ETL) é totalmente automatizado via GitHub Actions, eliminando a necessidade de intervenção manual e custos de infraestrutura.

O fluxo de dados segue as seguintes etapas:

1. **Extração (E):** Scripts Python consultam APIs públicas (NVD, OTX, HIBP, AbuseIPDB).
2. **Transformação (T):** Dados brutos são limpos, normalizados e enriquecidos (ex: geolocalização de IPs, classificação de CVEs).
3. **Carga (L):** Os dados processados são salvos como arquivos JSON estáticos na pasta /site, servindo como base de dados para o frontend.

---

2. Pré-requisitos de Ambiente (Local)

Para executar ou modificar os scripts de manipulação de dados localmente, o ambiente de desenvolvimento deve atender aos seguintes requisitos:

- **Linguagem:** Python 3.10 ou superior.
- **Gerenciador de Pacotes:** pip.
- **Bibliotecas Necessárias:**

Bash

pip install pandas requests

---

3. Configuração de Credenciais e Segurança

O projeto utiliza chaves de API (API Keys) para acessar serviços de terceiros. Por motivos de segurança, **nenhuma chave é armazenada no código-fonte**.

3.1. Variáveis de Ambiente (Environment Variables)

Os scripts Python (src/scripts/*.py) são projetados para ler credenciais exclusivamente através de variáveis de ambiente.

Variável	Descrição	Serviço / Fonte
API_NVD_CVE	Chave de acesso à base de vulnerabilidades (NIST).	NVD NIST
API_OTX	Chave de acesso à inteligência de ameaças.	AlienVault OTX
API_ABUSEIPDB	Chave para consulta de reputação e geolocalização de IPs.	AbuseIPDB

3.2. Configuração no GitHub Actions (Produção)

No ambiente de produção (nuvem), as chaves são gerenciadas via **GitHub Secrets**:

1. Acesse o repositório > *Settings* > *Secrets and variables* > *Actions*.
2. As chaves são injetadas no ambiente de execução apenas durante o tempo de vida do Workflow.

---

4. Execução dos Scripts de Manipulação (ETL)

Abaixo, detalhamos o comando de execução e a função de cada script responsável pela manipulação dos dados.

4.1. Coletor de Vulnerabilidades (CVEs)

- **Arquivo:** src/scripts/coletor_cve.py
- **Comando:** python src/scripts/coletor_cve.py
- **Função:** Consulta as vulnerabilidades publicadas nas últimas 24h a 7 dias. Aplica lógica de categorização textual para identificar tipos de falha (ex: "SQL Injection", "XSS") e simplifica o score CVSS.
- **Saída:** site/cve_kpis.json

4.2. Coletor de Vazamentos de Dados (Breaches)

- **Arquivo:** src/scripts/coletor_vazamentos.py
- **Comando:** python src/scripts/coletor_vazamentos.py
- **Função:** Consulta a base do *Have I Been Pwned*. Ordena os vazamentos por data de divulgação, formata números inteiros e anonimiza campos sensíveis.
- **Saída:** site/hibp_kpis.json

4.3. Coletor de Ameaças Globais (OTX)

- **Arquivo:** src/scripts/coletor_otx.py
- **Comando:** python src/scripts/coletor_otx.py

- **Função:** Agrega "pulsos" de ameaças globais, remove duplicatas e extrai tags de categorização (ex: Phishing, Malware) para análise de tendências.
- **Saída:** site/otx_kpis.json

4.4. Coletor de Geolocalização de IPs

- **Arquivo:** src/scripts/coletor_paises.py
- **Comando:** python src/scripts/coletor_paises.py
- **Função:** Consulta IPs maliciosos reportados recentemente, converte o endereço IP em País de Origem (GeoIP) e agrega os totais por nação.
- **Saída:** site/paises_kpis.json

---

5. Tratamento de Erros e Logs

Para garantir a robustez do processo automatizado, todos os scripts implementam tratamento de exceções (try-except) para os seguintes cenários:

1. **Falha de API (HTTP 5xx):** O script registra o erro no log e encerra a execução sem sobrescrever os arquivos JSON antigos (preservando os dados anteriores no site).
2. **Limite de Requisições (HTTP 429 - Rate Limit):** Implementação de time.sleep() estratégico entre requisições para respeitar as cotas gratuitas das APIs.
3. **Dados Vazios:** Se uma API retornar zero resultados, o script gera um JSON vazio válido [] ou mantém o arquivo anterior, evitando quebra do Frontend (JavaScript).

---

6. Automação e Agendamento (CI/CD)

A orquestração desses scripts é definida no arquivo de workflow .github/workflows/deploy_diario.yml.

- **Frequência:** Execução automática agendada para **06:00** e **18:00** (Horário de Brasília).
- **Gatilho Manual:** Suporte a execução sob demanda via *workflow_dispatch* para testes ou atualizações emergenciais.
- **Deploy:** Após a geração bem-sucedida dos novos arquivos JSON na pasta /site, o GitHub Actions realiza o deploy automático da nova versão para o GitHub Pages.