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

Entregável 2: Scripts de ETL (Extração, Transformação e Carga)

Documentação dos Processos de ETL

1. Visão Geral do Pipeline O processo de ETL é executado automaticamente via GitHub Actions. Os scripts Python atuam como *crawlers* de APIs públicas, normalizam os dados e salvam o resultado em arquivos estáticos prontos para consumo pelo Frontend.

2. Script: Coletor CVE (coletor_cve.py)

- **Fonte:** NVD NIST API (National Vulnerability Database).
- **Transformação:** Filtra vulnerabilidades recentes; classifica automaticamente o tipo de falha (ex: SQL Injection, RCE) com base na descrição técnica e simplifica o score CVSS.
- **Carga:** Gera o arquivo site/cve_kpis.json.

database > coletor_cve.py > ...

```python
# Importar bibliotecas necessárias
import pandas as pd
import requests
from datetime import datetime, timedelta
import time
import json
import os

# ================================================================================
# 0. CONFIGURAÇÃO DA API (NVD)
# ================================================================================
API_KEY = "49e988e1-7235-47d9-8f56-a2e7e4cae151"
API_URL = "https://services.nvd.nist.gov/rest/json/cves/2.0"
headers = {'apiKey': API_KEY}

# --- FUNÇÃO DE CLASSIFICAÇÃO ---
def classificar_vulnerabilidade(descricao):
    """
    Lê a descrição técnica e retorna uma categoria simplificada.
    """
    if not descricao:
        return 'Outros'

    desc = descricao.lower()

    if 'remote code execution' in desc or 'execute arbitrary code' in desc:
        return 'RCE (Execução Remota)'
    elif 'denial of service' in desc or 'dos' in desc:
        return 'DoS (Negação de Serviço)'
    elif 'sql injection' in desc:
        return 'SQL Injection'
    elif 'cross-site scripting' in desc or 'xss' in desc:
        return 'XSS'
    elif 'privilege escalation' in desc or 'gain privileges' in desc:
        return 'Escalada de Privilégio'
    elif 'buffer overflow' in desc:
        return 'Buffer Overflow'
    else:
        return 'Outros'

print("="*80)
print("ATRIBUIÇÃO: Este produto usa dados da NVD API, mas não é endossado ou certificado pela NVD.")
print("="*80 + "\n")
```

3. Script: Coletor de Vazamentos (coletor_vazamentos.py)

- **Fonte:** Have I Been Pwned API.
- **Transformação:** Ordena os vazamentos por data de divulgação; formata números grandes de contas afetadas; anonimiza dados sensíveis e padroniza datas.
- **Carga:** Gera o arquivo site/hibp_kpis.json.

database > coletor_vazamentos.py > ...

```python
# Importar bibliotecas necessárias
import pandas as pd
import requests
import time
import json
import warnings

# Suprimir avisos de "UserWarning"
warnings.filterwarnings('ignore', category=UserWarning)

# ==============================================================================
# 0. CONFIGURAÇÃO DA API (Have I Been Pwned - HIBP)
# ==============================================================================
headers = {
    'User-Agent': 'Project-Aegis-Coletor-Academico'
}
API_URL = "https://haveibeenpwned.com/api/v3/breaches"
SLEEP_TIME = 2

print("="*80)
print("ATRIBUIÇÃO: Usando a Base de Dados 'Have I Been Pwned' (HIBP) v3.")
print("="*80 + "\n")

# ==============================================================================
# 1. DICIONÁRIO DE DADOS (NOSSO ALVO)
# ==============================================================================
dicionario_dados_vazamento = {
    "nome_vazamento": "String (Nome da empresa/site, ex: 'SocialNet')",
    "data_vazamento": "Date (Data em que o vazamento ocorreu)",
    "contas_afetadas": "Integer (Número total de contas vazadas)",
    "setor": "String (Indústria da empresa, ex: 'Technology')"
}
print("--- 1. ESTRUTURA DO DICIONÁRIO DE DADOS (NOSSO ALVO) ---")
for chave, valor in dicionario_dados_vazamento.items():
    print(f"{chave}: {valor}")
print("\n" + "="*50 + "\n")

```

4. Script: Coletor de Geolocalização (coletor_paises.py)

- **Fonte:** AbuseIPDB (Blacklist de IPs maliciosos).
- **Transformação:** Consulta a lista de IPs reportados recentemente, realiza a geolocalização (IP -> País) e agrega a contagem de ameaças por nação para alimentar o gráfico.
- **Carga:** Gera o arquivo site/paises_kpis.json.

database > coletor_paises.py > ...

```python
# Importar bibliotecas necessárias
import pandas as pd
import requests
import json
import time
from datetime import datetime

# ================================================================================
# 0. CONFIGURAÇÃO DA API (AbuseIPDB)
# ================================================================================
API_KEY = "4947aff3d5a68c6db3a3d38206e061b8df8033400f2c14c332acfc48c2ae8bc0b0924798870703a3"

if "SUA_CHAVE" in API_KEY:
    print("ERRO: Adicione sua chave de API.")
    exit()

headers = {
    'Key': API_KEY,
    'Accept': 'application/json',
}
API_URL = "https://api.abuseipdb.com/api/v2/blacklist"
# Limitamos a 10.000 IPs (máximo gratuito)
PARAMS = {
    'confidenceMinimum': 50,
    'limit': 10000
}

# --- DICIONÁRIO DE MAPEAMENTO (Country Code -> Country Name) ---
COUNTRY_MAP = {
    "US": "United States", "CN": "China", "RU": "Russia", "DE": "Germany",
    "NL": "Netherlands", "FR": "France", "GB": "United Kingdom",
    "IN": "India", "BR": "Brazil", "VN": "Vietnam", "UA": "Ukraine",
    "KR": "South Korea", "JP": "Japan", "CA": "Canada", "AU": "Australia",
    "PL": "Poland", "TR": "Turkey", "HK": "Hong Kong", "SG": "Singapore",
    "TW": "Taiwan", "IR": "Iran", "RO": "Romania", "CZ": "Czech Republic",
    "ID": "Indonesia", "ES": "Spain", "IT": "Italy", "TH": "Thailand"
}

print("="*80)
print("ATRIBUIÇÃO: Usando a Base de Dados 'AbuseIPDB'.")
print("="*80 + "\n")

```

5. Script: Coletor OTX (coletor_otx.py)

- **Fonte:** AlienVault Open Threat Exchange.
- **Transformação:** Agrega "Pulsos" de ameaças globais; remove duplicatas e extrai tags de categorização (ex: Phishing, Malware) para análise de tendências.
- **Carga:** Gera o arquivo site/otx_kpis.json.

database > coletor_otx.py > ...

```python
# Importar bibliotecas necessárias
import pandas as pd
import requests
import json
from datetime import datetime, timedelta
import time

# ==============================================================================
# 0. CONFIGURAÇÃO DA API (AlienVault OTX)
# ==============================================================================
API_KEY = "6dc4519f2fe20cb267b376d6f626b0ee64243693f27d30109f3e6b1802e78c2d"

if "SUA_CHAVE_OTX_AQUI" in API_KEY:
    print("ERRO: Adicione sua chave de API da OTX.")
    exit()

headers = {
    'X-OTX-API-KEY': API_KEY
}

# Usando /subscribed pois mostrou-se mais estável que /activity
API_URL = "https://otx.alienvault.com/api/v1/pulses/subscribed"

# Configurações de Coleta
RESULTS_PER_PAGE = 50
MAX_PAGES = 200 # Limite de segurança para não rodar infinito (200 * 50 = 10.000 pulsos)
DIAS_PARA_BUSCAR = 365 # Queremos 1 ano de dados

print("="*80)
print("ATRIBUIÇÃO: Usando a Base de Dados 'AlienVault OTX' (Open Threat Exchange).")
print("="*80 + "\n")

# ==============================================================================
# 1. COLETA REAL DE DADOS (ETL - 365 DIAS)
# ==============================================================================
print("--- 2. COLETANDO DADOS REAIS DA API DA OTX (Janela de 1 Ano) ---")

# Calcular a data limite (1 ano atrás)
data_limite = datetime.now() - timedelta(days=DIAS_PARA_BUSCAR)
print(f"Buscando dados a partir de: {data_limite.strftime('%Y-%m-%d')}")

dados_coletados = []
parar_coleta = False
```