* **Text Extraction and CSV Data Logging**

Extracts cleaned textual content from onion pages and logs URL, timestamp, and text to CSV for further analysis.

import requests

from bs4 import BeautifulSoup

import re

import time

from datetime import datetime

import csv

from requests.adapters import HTTPAdapter

from requests.packages.urllib3.util.retry import Retry

import urllib3

# Disable insecure warnings (for self-signed certs on onion sites)

urllib3.disable\_warnings(urllib3.exceptions.InsecureRequestWarning)

# Tor SOCKS5 proxy config (adjust port if needed)

PROXIES = {

'http': 'socks5h://127.0.0.1:9150',

'https': 'socks5h://127.0.0.1:9150'

}

HEADERS = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101 Firefox/89.0'

}

# Seed URLs to start crawling (clearnet + onion)

SEED\_URLS = [

"https://ahmia.fi",

"http://dreadytofatroptsdj6io7l3xptbet6onoyno2yv7jicoxknyazubrad.onion",

"http://torlib7fmhyvfv2k7s77xigdds3rosio6k6bxnn256xmtzlbgyizduqd.onion"

]

OUTPUT\_CSV = "darkweb\_raw\_data.csv"

def get\_requests\_session(max\_retries=3, backoff\_factor=1):

session = requests.Session()

retries = Retry(

total=max\_retries,

backoff\_factor=backoff\_factor,

status\_forcelist=[500, 502, 503, 504],

allowed\_methods=["HEAD", "GET", "OPTIONS"]

)

adapter = HTTPAdapter(max\_retries=retries)

session.mount("http://", adapter)

session.mount("https://", adapter)

return session

def fetch\_page(session, url, timeout=30):

try:

print(f"[+] Fetching: {url}")

response = session.get(url, proxies=PROXIES, headers=HEADERS, timeout=timeout, verify=False)

response.raise\_for\_status()

return response.text

except requests.exceptions.RequestException as e:

print(f" [✖] Failed to fetch {url}: {e}")

return None

def extract\_onion\_links(html):

soup = BeautifulSoup(html, "html.parser")

links = set()

for a in soup.find\_all("a", href=True):

href = a['href']

if ".onion" in href:

match = re.search(r'(http[s]?://[^"\']+\.onion)', href)

if match:

links.add(match.group(1))

return list(links)

def extract\_clean\_text(html):

soup = BeautifulSoup(html, "html.parser")

for script\_or\_style in soup(["script", "style", "noscript"]):

script\_or\_style.decompose()

text = soup.get\_text(separator=" ", strip=True)

text = re.sub(r'\s+', ' ', text)

return text

def save\_to\_csv(data\_rows):

with open(OUTPUT\_CSV, "w", newline="", encoding="utf-8") as csvfile:

writer = csv.writer(csvfile)

writer.writerow(["URL", "Timestamp", "RawText"])

for row in data\_rows:

writer.writerow(row)

print(f"[+] Saved {len(data\_rows)} records to {OUTPUT\_CSV}")

def main():

print("[\*] Starting Dark Web Data Extraction with retries")

session = get\_requests\_session(max\_retries=3, backoff\_factor=2)

discovered\_onion\_links = set()

all\_data = []

# Step 1: Crawl each seed URL to discover onion links

for seed\_url in SEED\_URLS:

seed\_html = fetch\_page(session, seed\_url)

if not seed\_html:

continue

links = extract\_onion\_links(seed\_html)

print(f"[+] Found {len(links)} onion links on {seed\_url}")

discovered\_onion\_links.update(links)

time.sleep(5) # polite delay

print(f"\n[+] Total unique onion links discovered: {len(discovered\_onion\_links)}")

# Step 2: Visit each discovered onion link to extract raw text

for onion\_url in discovered\_onion\_links:

page\_html = fetch\_page(session, onion\_url)

if not page\_html:

continue

clean\_text = extract\_clean\_text(page\_html)

timestamp = datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S UTC")

all\_data.append([onion\_url, timestamp, clean\_text])

print(f" [✔] Extracted text from {onion\_url} ({len(clean\_text)} chars)")

time.sleep(5) # polite delay

# Step 3: Save all extracted raw text data

if all\_data:

save\_to\_csv(all\_data)

else:

print("[!] No data extracted from onion links.")

if \_\_name\_\_ == "\_\_main\_\_":

main()