* **NLP Processing and MITRE ATT&CK Mapping Script**

Utilizes spaCy to preprocess text, performs Named Entity Recognition (NER), and applies fuzzy matching against MITRE ATT&CK technique datasets to identify and score attack techniques.

import requests

from bs4 import BeautifulSoup

import spacy

import re

import time

# Load spaCy small English model

nlp = spacy.load("en\_core\_web\_sm")

# Tor proxy for onion sites

proxies = {

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

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

}

# List of URLs to monitor (onion and clearnet)

urls = [

"https://ahmia.fi/",

"http://dreadytofatroptsdj6io7l3xptbet6onoyno2yv7jicoxknyazubrad.onion/",

"http://torlib7fmhyvfv2k7s77xigdds3rosio6k6bxnn256xmtzlbgyizduqd.onion/"

]

# Helper to check if URL is onion (needs proxy)

def is\_onion(url):

return ".onion" in url

# Fetch and return page HTML content

def fetch\_page(url):

try:

if is\_onion(url):

response = requests.get(url, proxies=proxies, timeout=30, verify=False)

else:

response = requests.get(url, timeout=30)

response.raise\_for\_status()

return response.text

except Exception as e:

print(f"[ERROR] Could not access {url} → {e}")

return None

# Extract clean text from HTML

def extract\_text(html):

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

# Remove script and style content

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

script\_or\_style.decompose()

text = soup.get\_text(separator=" ")

# Normalize whitespace

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

return text

# Run spaCy NLP to extract named entities

def extract\_entities(text):

doc = nlp(text)

return [(ent.text, ent.label\_) for ent in doc.ents]

def main():

all\_entities = []

print("[\*] Starting Dark Web Marketplace Monitor with NLP")

for url in urls:

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

html = fetch\_page(url)

if html:

text = extract\_text(html)

entities = extract\_entities(text)

print(f" Extracted {len(entities)} named entities from {url}")

for ent\_text, ent\_label in entities:

print(f" - {ent\_text} [{ent\_label}]")

all\_entities.extend(entities)

else:

print(f" Skipping NLP due to fetch failure for {url}")

time.sleep(5) # Polite delay between requests

# Save all extracted entities to a text file

with open("extracted\_entities.txt", "w", encoding="utf-8") as f:

for ent\_text, ent\_label in all\_entities:

f.write(f"{ent\_text}\t{ent\_label}\n")

print(f"[✔] Completed monitoring. Total entities extracted: {len(all\_entities)}")

print("[✔] Entities saved to extracted\_entities.txt")

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

main()