```
import re

import os

import sys

import json

import time

import openai

import textwrap

import requests

import itertools

import concurrent.futures

from bs4 import BeautifulSoup

from urllib.parse import urlparse


print("""

""")


# Set Variables URL และระดับการเรียกซ้ำ ---> (URL_Finder)

class Set_Variable1():
```

```
    url = input('เลือกURL หรือ DomainIP แล้วกด Enter: ').strip()
    print("")
    if not (url.startswith("http://") or url.startswith("https://")):
        url = "https://" + url

    try:
        response = requests.get(url)
        if response.status_code == 200:
            print("สคริปต์จะติดตามลิงก์ภายในบนเว็บไซต์จนถึงระดับการเรียกซ้ำสูงสุดที่ระบุ")
            while True:
                try:
                    max_recursion_level = int(input('ระดับการScan (ระหว่าง 1-3 | ค่าเริ่มต้น = 1): ').strip() or 1)
                    if 1 <= max_recursion_level <= 3:
                        break
                    else:
                        print('อินพุตต้องอยู่ระหว่าง 1 ถึง 3')
                except ValueError:
                    print('อินพุตต้องเป็นตัวเลข')
            print("")
    except requests.exceptions.RequestException:
        print('ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้')
        exit()
    except:
        print('URL ไม่ถูกต้อง')
url = Set_Variable1.url

# ตั้งค่า file_name (Remove any extension, subdomain are /)
class file_name_integrity():
```

```
    # แยกวิเคราะห์ URL
    parsed_url = urlparse(url)
    # รับชื่อโดเมนที่ไม่มีโดเมนย่อย
    domain_name = parsed_url.netloc.split('.')[-2] + '.' + parsed_url.netloc.split('.')[-1]
    # รวมชื่อโดเมน, นามสกุลโดเมน, และนามสกุลไฟล์เพื่อสร้างชื่อไฟล์
    file_name = f"{domain_name}.txt"

file_name = file_name_integrity.file_name

#2
class Set_Variable2():
    JS_scanner_file_name = file_name
    instructions = """การค้นหาช่องโหว่ผ่านซอร์สโค้ดตัวนี้ถูกพัฒนาผ่านมา 20 ปีแล้ว ควรวิเคราะห์โค้ดต่อไปนี้โดยเคารพคำแนะนำ

    Instructions:
    1. A list of javascript code will be provided to you at the bottom, each javascript code is delimited by --- at the start and --- at the end and include a unique identifier (id[X],JS#[X]). (The X here is a place holder)
    2. Analyse the javascript code
    3. DO NOT INCLUDE THE RESPONSE IN THE TEMPLATE IF THE SNIPPET OF CODE IS NOT VULNERABLE
    4. Use the following template to respond using the corresponding id and INCLUDING +++ AT THE START AND +++ AT THE END of each explanations
    5. Do not include ANY other statement after using the template and make sure you have used the template exactly how it supposed to be use

    +++
    (id[X],JS#[X])
    Secure: [Vulnerable or Not Vulnerable]

    Text: [Explain SHORTLY, IF vulnerable, why it's vulnerable]
```

```
    +++

    DONT FORGET TO ADD THE +++ AT THE END

    The following is all the javascript snippet you need to analyse
    """

#3
class Set_Variable3():
    JS_Unique_file_name = "JS_Unique_" + file_name

    api_key_file = 'API_Key.txt'

    if os.path.exists(api_key_file) and os.path.getsize(api_key_file) > 0:
        with open(api_key_file, 'r') as f:
            api_key = f.read().strip()
    else:
        print(''' 1.ไปที่ https://platform.openai.com/account/api-keys
    2.สร้างคีย์ API แล้วดำเนินการวางที่ตำแหน่งนี้
            ''')
        api_key = input("รวมคีย์ OpenAI API ของคุณ: ")
        with open(api_key_file, 'w') as f:
            f.write(api_key)
            print("")

#4
class Set_Variable4():
    ChatGPT_file_name = "chatGPT_" + file_name

#5
```

```
class Set_Variable5():
    JS_Unique_file_name = "JS_Unique_" + file_name
    JS_URL_file_name = "JS_URL_" + file_name

#6
class Set_Variable6():
    Clean_up_file_name = "final_" + file_name

# ============================================================

def URL_Finder(url, max_recursion_level, file_name):

    visited_urls = []

    def crawl_website(url, max_recursion_level, visited_urls):
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0;Win64) AppleWebkit/537.36 (KHTML, like Gecko)
Chrome/89.0.4389.82 Safari/537.36'}
        session = requests.Session()
        response = requests.get(url, headers=headers, allow_redirects=True)
        final_url = response.url
        if response.status_code != 200:
            return []
        soup = BeautifulSoup(response.text, 'html.parser')
        parsed_url = urlparse(final_url)
        same_domain_urls = []
        visited_urls.append(url)
        for link in soup.find_all('a'):
            link_url = link.get('href')
```

```
        parsed_link_url = urlparse(link_url)
        if parsed_link_url.netloc == parsed_url.netloc:
            # ลบเครื่องหมายทับต่อท้ายออกจาก URL หากมี
            link_url = link_url.rstrip('/')
            if max_recursion_level > 0 and link_url not in visited_urls:
                same_domain_urls.append(link_url)
                same_domain_urls.extend(crawl_website(link_url, max_recursion_level - 1, visited_urls))

    # ตรวจสอบแผนผังเว็บไซต์หรือ robots.txt
    sitemap_url = final_url + '/sitemap.xml'
    robots_url = final_url + '/robots.txt'
    for file_url in [sitemap_url, robots_url]:
        try:
            file_response = requests.get(file_url)
            if file_response.status_code != 200:
                continue
            file_soup = BeautifulSoup(file_response.text, 'xml')
            for link in file_soup.find_all('loc'):
                link_url = link.get_text().rstrip('/')
                parsed_link_url = urlparse(link_url)
                if parsed_link_url.netloc == parsed_url.netloc and max_recursion_level > 0 and link_url not in
visited_urls:
                    same_domain_urls.append(link_url)
                    visited_urls.append(link_url)
        except:
            pass
    return same_domain_urls
```

```
def run_animation():
    spinner = itertools.cycle(['⣾', '⣽', '⣻', '⢿', '⡿', '⣟', '⣯', '⣷'])
    while not crawl_future.done():
        sys.stdout.write("\rกำลังรวบรวมข้อมูล " + next(spinner))
        sys.stdout.flush()
        time.sleep(0.5)
    print("")

if __name__ == '__main__':
    max_recursion_level = int(max_recursion_level)
    while True:
        with concurrent.futures.ThreadPoolExecutor() as executor:
            crawl_future = executor.submit(crawl_website, url, max_recursion_level, visited_urls)
            run_animation()
        urls = crawl_future.result()
        urls.append(url)
        urls.sort() # จัดเรียง URL ตามตัวอักษร

        number_of_urls_found = len(urls)
        print("")
        print("สแกนพบ " + str(number_of_urls_found) + " สแกนพบ ↓")
        print("")
        for url in urls:
            print(url)

        try:
            with open(file_name, 'w') as f:
```

```
                for url in urls:
                    f.write(url + '\n')
            except IOError:
                print(f"Unable to write to {file_name}")
            break

def Javascript(JS_scanner_file_name, instructions):

    def search_scripts(urls):
        # ตั้งค่าตัวแปรตัวนับเพื่อสร้างรหัสที่ไม่ซ้ำกัน
        counter = 1

        # สร้าง dictionary เพื่อติดตามแท็กสคริปต์ที่เคยเห็นแล้ว
        seen_scripts = {}

        # ทำซ้ำรายการ URL
        for url in urls:
            # ตั้งค่าตัวแปรตัวนับสำหรับ URL นี้
            script_counter = 1

            # สร้างรหัสที่ไม่ซ้ำกันสำหรับ URL ปัจจุบัน
            id = f"id{counter}"

            # ส่งคำขอไปยัง URL, การเปลี่ยนเส้นทางต่อไปนี้
            response = requests.get(url, allow_redirects=True)

            # แยกวิเคราะห์ HTML ของหน้าเว็บ
            soup = BeautifulSoup(response.text, 'html.parser')

            # ค้นหาแท็กสคริปต์ทั้งหมด
```

```
script_tags = soup.find_all('script')

# สร้างรหัสที่ไม่ซ้ำกันสำหรับ URL ปัจจุบัน
id = f"id{counter}"

# ตั้งค่าตัวแปรตัวนับ
script_counter = 1

# แยกข้อความจากแท็กสคริปต์แต่ละแท็กและพิมพ์ออกมา
for script in script_tags:
    # ข้ามแท็กสคริปต์ที่มีข้อความว่างเปล่า
    if not script.text.strip():
        continue

    # สร้างสตริง xpath
    xpath = ''
    element = script
    while element is not None:
        if xpath:
            xpath = '/' + xpath
        xpath = element.name + xpath
        element = element.parent

    # ลบครั้งแรก '/' อักขระจากสตริง XPath
    xpath = xpath[10:]

    # ตรวจสอบว่าเห็นสคริปต์นี้แล้วหรือไม่
    script_hash = hash(script.text)
    if script_hash in seen_scripts:
```

```
            # หากเห็นสคริปต์แล้ว, เพิ่มลงในพจนานุกรมที่ซ้ำกัน
            duplication = f"({seen_scripts[script_hash]},JS#{script_counter})"
            key = f"{id},JS#{script_counter}"
            with open("JS_URL_" + file_name.replace(".txt", ".txt"), "a") as f:
                duplicates_dict = {key: {"url": url, "duplication": duplication, "xpath": xpath}}
                json.dump(duplicates_dict, f)
                f.write("\n")
        else:
            seen_scripts[script_hash] = id
            key = f"{id},JS#{script_counter}"
            with open("JS_URL_" + file_name.replace(".txt", ".txt"), "a") as f:
                duplicates_dict = {key: {"url": url, "xpath": xpath, }}
                json.dump(duplicates_dict, f)
                f.write("\n")
            with open("JS_Unique_" + file_name, "a") as f:
                f.write('\n')
                f.write("---\n")
                f.write(f"({id},JS#{script_counter}) \n")
                f.write('\n')  # เขียนอักขระบรรทัดใหม่ก่อนรหัสสคริปต์เสมอ
                f.write(script.text + '\n')
                f.write("---\n")

        # เพิ่มตัวนับสคริปต์
        script_counter += 1

    # เพิ่มตัวนับ URL
    counter += 1
```

```
    return True

def run_animation(future_search, wait_time):
    print(f"เวลาในการวิคราะห์คือ {wait_time} วินาที.")
    spinner = itertools.cycle(['⣾', '⣽', '⣻', '⢿', '⡿', '⣟', '⣯', '⣷', '⣾', '⣽'])
    while not future_search.done():
        sys.stdout.write("\rวิเคราะห์องค์ประกอบ Javascript " + next(spinner))
        sys.stdout.flush()
        time.sleep(0.5)
    print("")

if __name__ == '__main__':
    file_name = JS_scanner_file_name

    # อ่าน URL จากไฟล์
    while True:
        file_name = JS_scanner_file_name
        try:
            with open(file_name, "r") as f:
                urls = f.read().splitlines()
                f.seek(0) # รีเซ็ตตัวชี้แฟ้มไปยังจุดเริ่มต้นของแฟ้ม
                wait_time = int(len(f.readlines()) / 2)
            break # ออกจากลูปหากเปิดไฟล์สำเร็จ
        except FileNotFoundError:
            print("Error: ไม่พบไฟล์ โปรดลองอีกครั้ง\n")
            exit()

    # เขียนผลลัพธ์ไปยังไฟล์
```

```
        with open("JS_Unique_" + file_name, "a") as f:
            # คำแนะนำสำหรับ CHAT-GPT
            f.write(textwrap.dedent(instructions))

        # เรียกใช้ฟังก์ชัน search_scripts โดยใช้พร้อมกัน.futures.ThreadPoolExecutor
        with concurrent.futures.ThreadPoolExecutor() as executor:
            future_search = executor.submit(search_scripts, urls)
            run_animation(future_search, wait_time)

def chatGPT_API(JS_Unique_file_name, api_key):
    openai.api_key = api_key

    def makeCall(message_arr):
        completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=message_arr)
        return completion.choices[0].message

    def conversation():
        message_array = []
        while True:
            filename = JS_Unique_file_name
            try:
                with open(filename, 'r') as file:
                    user_input = file.read()
                break # ออกจากลูปหากเปิดไฟล์สำเร็จ
            except FileNotFoundError:
                print("Error: คีย์ API ที่คุณให้ไว้ไม่ถูกต้อง, หรือไฟล์ API_KEY.txt ไม่สามารถเข้าถึงได้ โปรดตรวจสอบคีย์ API แล้วลองอีกครั้ง\n")
                exit()
```

```
        chatGPT_output_filename = filename[10:]

        message_obj = {"role": "user", "content": user_input}
        message_array.append(message_obj)

        resp = makeCall(message_array)
        resp_str = str(resp)
        resp_json = json.loads(resp_str)
        content = resp_json["content"].strip()

        with open('chatGPT_' + chatGPT_output_filename, 'w') as f:
            f.write(content)

    def run_animation(future_conversation):
        spinner = itertools.cycle(['⠋', '⠙', '⠹', '⠸', '⠼', '⠴', '⠦', '⠧', '⠇', '⠏'])
        while not future_conversation.done():
            sys.stdout.write("\rChatGPT กำลังตรวจสอบรหัส Javascript ของคุณ " + next(spinner))
            sys.stdout.flush()
            time.sleep(0.5)
        print("")

    if __name__ == '__main__':

        with concurrent.futures.ThreadPoolExecutor() as executor:
            future_conversation = executor.submit(conversation)
            run_animation(future_conversation)

def JS_Output_Filtering(ChatGPT_file_name):
    if __name__ == '__main__':
        while True:
```

```
        file_name = ChatGPT_file_name
        try:
            with open(file_name, 'r') as file:
                text = file.read()
            break # ออกจากลูปหากเปิดไฟล์สำเร็จ
        except FileNotFoundError:
            print("Error: ขนาดไฟล์เกินความจุของ ChatGPT.\n")
            exit()

    with open(file_name, 'r') as f:
        file_contents = f.read()

    snippet_regex = re.compile(r'\((id\d+),JS#(\d+)\)\nSecure:\s*(Not Vulnerable|Vulnerable)(?:\n\nText: (.*))?')

    snippet_data = {}

    for match in snippet_regex.finditer(file_contents):
        id_value = match.group(1) + ',' + 'JS#' + match.group(2)
        secure_value = match.group(3)
        text_value = match.group(4) if match.group(4) else None
        if secure_value == "Vulnerable":
            snippet_data[id_value] = {"secure": secure_value, "text": text_value}

    # Write the data to a file
    with open('Individual_JS_Vulnerable.txt', 'w') as f:
        for key, value in snippet_data.items():
            f.write("{" + f'"{key}": {value}' + "}\n")

def Interpretation(JS_Unique_file_name, JS_URL_file_name):
```

```
file_name_1 = "Individual_JS_Vulnerable.txt"
file_name_2 = JS_URL_file_name
file_name_3 = JS_Unique_file_name

# Step 1: อ่านเนื้อหาของไฟล์แรกและจัดเก็บแต่ละบรรทัดเป็นสตริงในรายการ
with open(file_name_1, "r") as f:
    file1_lines = f.readlines()

if not file1_lines:
    print("\033[1m\033[32mNo vulnerability found\033[0m")
    return

# Step 2: อ่านเนื้อหาของไฟล์ที่สองและจัดเก็บแต่ละบรรทัดเป็นสตริงในรายการ
with open(file_name_2, "r") as f:
    file2_lines = f.readlines()

# Step 3: อ่านเนื้อหาของไฟล์ที่สามและจัดเก็บแต่ละบรรทัดเป็นสตริงในรายการ
with open(file_name_3, "r") as f:
    file3_lines = f.readlines()

# Step 4: วนซ้ำแต่ละบรรทัดในไฟล์แรกและแยกค่า id และ JS
output = "" # เริ่มต้นตัวแปรเอาต์พุต
for line in file1_lines:
    # แยกค่า id และ JS โดยใช้การจัดการสตริง
    id_js = line.split(":")[0].strip().strip("{").strip('"')
    text = line.split(":")[-1].strip().strip('}').strip().strip('"')
    text = text[1:-1] # ลบอักขระตัวแรกและตัวสุดท้าย
    output += "\n\033[1m\033[31mThe JS code below may contain a vulnerability. ---> " + id_js + "\033[0m"
    output += "\n"
```

```
output += "\nExplanation: " + text
output += "\n"

# Step 6: วนซ้ำผ่านแต่ละบรรทัดในไฟล์ที่สาม และบันทึกบรรทัดระหว่างจุดเริ่มต้นและจุดสิ้นสุดของส่วนย่อย
found_snippet = False
file3_lines_copy = file3_lines.copy() # สร้างสำเนาของรายการเพื่อรักษาตัวทำซ้ำ
for i, line3 in enumerate(file3_lines_copy):
    if ("(" + id_js + ")") in line3:
        found_snippet = True
        snippet_lines = []
    elif found_snippet and line3.strip() != '---':
        snippet_lines.append(line3)
    elif found_snippet:
        found_snippet = False
        snippet_text = "".join(snippet_lines).strip()
        output += "\n---\n"
        output += snippet_text + "\n"
        output += line3.strip() + "\n" # พิมพ์บรรทัดท้ายของข้อมูลโค้ด
        break

# Step 5: วนซ้ำผ่านแต่ละบรรทัดในไฟล์ที่สอง และตรวจสอบว่าค่า id และ JS ปรากฏขึ้นหรือไม่
output += "\n"
output += "The Following URL's are touched by this potential vulnerability"
output += "\n" + '=' * 55 + "\n"
found = False
for line2 in file2_lines:
    if id_js in line2:
        found = True
```

```
            url = line2.split('"url": "')[1].split('"')[0]
            output += url + "\n"

    if not found:
        output += "Not found\n"

    final_name = "final_" + file_name_2[7:]

    # wriเพื่อส่งออกไปยังไฟล์ชื่อ "Final"
    with open(final_name, "w") as f:
        f.write(output)
        print(output)

def clean_up_files(Clean_up_file_name):
    for filename in os.listdir('.'):
        if filename.endswith('.txt') and filename not in ['API_Key.txt', Clean_up_file_name]:
            os.remove(filename)

# ============================================================

# เรียกคืนตัวแปร
max_recursion_level = Set_Variable1.max_recursion_level
url = Set_Variable1.url

JS_scanner_file_name = Set_Variable2.JS_scanner_file_name
instructions = Set_Variable2.instructions

JS_Unique_file_name = Set_Variable3.JS_Unique_file_name
api_key = Set_Variable3.api_key

ChatGPT_file_name = Set_Variable4.ChatGPT_file_name
```

```
JS_URL_file_name = Set_Variable5.JS_URL_file_name

Clean_up_file_name = Set_Variable6.Clean_up_file_name


# เปิด Functions
class start_UI():
    print("=" * 45)
    print("")
URL_Finder(url, max_recursion_level, file_name)
class end_UI():
    print("")
    print("=" * 45)

class start_UI():
    print("")
Javascript(JS_scanner_file_name, instructions)
class end_UI():
    print("")
    print("=" * 45)

class start_UI():
    print("")
chatGPT_API(JS_Unique_file_name, api_key)
class end_UI():
    print("")
    print("=" * 45)

JS_Output_Filtering(ChatGPT_file_name)
```

```
class start_UI():

    print("")

Interpretation(JS_Unique_file_name, JS_URL_file_name)

clean_up_files(Clean_up_file_name)
```