ส่วนประกอบหลักของโค้ด

1. การตั้งค่าตัวแปร (Set_Variable1):
   - ระบบจะขอให้ผู้ใช้ป้อน URL ของเว็บไซต์หรือ IP ที่ต้องการสแกน จากนั้นระบบจะตรวจสอบการเชื่อมต่อไปยัง URL ที่ให้มา หากการเชื่อมต่อสำเร็จ ระบบจะแสดงข้อความแจ้งและขอให้ผู้ใช้กำหนดระดับการสแกน (1 ถึง 3 ระดับ) ซึ่งหมายถึงจำนวนชั้นของลิงก์ที่จะติดตาม
2. การกำหนดชื่อไฟล์ (file_name_i):
   - ระบบมีการกำหนดชื่อไฟล์ที่ใช้สำหรับการบันทึกข้อมูลที่สแกนได้ โดยมีการปรับแต่งชื่อไฟล์ เช่น การลบส่วนขยายหรือการแปลง Subdomain เป็น / ซึ่งอาจเป็นการจัดระเบียบโครงสร้างไฟล์ที่จัดเก็บผลลัพธ์จากการสแกน
3. ฟังก์ชันหลักสำหรับการสแกน URL (URL_Finder):
   - ฟังก์ชันนี้จะทำงานโดยสแกนลิงก์ทั้งหมดภายในเว็บไซต์ตามระดับการสแกนที่กำหนดไว้ เพื่อเก็บข้อมูล URL ภายในเว็บไซต์ที่เชื่อมโยงกัน โดยการใช้ BeautifulSoup เพื่อดึงข้อมูลลิงก์ใน HTML และติดตามลิงก์ที่ค้นพบ
4. การวิเคราะห์ JavaScript (Javascript):
   - มีการวิเคราะห์ไฟล์ JavaScript ที่ค้นพบจากการสแกนเว็บไซต์ โดยใช้ไฟล์ที่ระบุชื่อไว้ใน JS_scanner_file_name และข้อมูลการสแกน JavaScript จะถูกส่งไปยังฟังก์ชันที่อาจจะเชื่อมต่อกับ OpenAI API
5. การใช้งาน OpenAI API :
   - ระบบใช้ OpenAI API สำหรับการวิเคราะห์ข้อมูลเพิ่มเติมที่ได้จาก JavaScript การวิเคราะห์นี้อาจเป็นการขอความช่วยเหลือจาก OpenAI ในการตรวจสอบช่องโหว่หรือการสร้างรายงานเชิงลึก
6. การฟิลเตอร์ผลลัพธ์ของ OpenAI (JS_Output_Filtering):
   - ผลลัพธ์ที่ได้จากการวิเคราะห์ผ่าน OpenAI จะถูกฟิลเตอร์และจัดรูปแบบให้อ่านง่ายขึ้น โดยใช้ฟังก์ชัน JS_Output_Filtering
7. การทำความสะอาดไฟล์ (clean_up_files):
   - หลังจากการสแกนและการวิเคราะห์เสร็จสิ้น ระบบจะทำการลบไฟล์ที่ไม่จำเป็นออก (ไฟล์ .txt) เพื่อป้องกันการเกิดไฟล์ขยะ โดยยกเว้นไฟล์สำคัญ เช่น API_Key.txt หรือไฟล์ที่กำลังใช้งานอยู่

ขั้นตอนการทำงานของโค้ด

1. ผู้ใช้ป้อน URL หรือ IP ของเว็บไซต์ที่ต้องการสแกน
2. ระบบตรวจสอบการเชื่อมต่อกับเว็บไซต์
3. ผู้ใช้กำหนดระดับการสแกน (1-3 ระดับ)
4. ระบบทำการสแกนลิงก์ภายในเว็บไซต์ตามระดับที่กำหนด
5. ระบบค้นหาและวิเคราะห์ไฟล์ JavaScript จากเว็บไซต์
6. ใช้ OpenAI API เพื่อทำการวิเคราะห์เพิ่มเติมจากผลลัพธ์ที่ได้
7. ผลลัพธ์จาก OpenAI จะถูกฟิลเตอร์และแสดงผล
8. ระบบทำความสะอาดไฟล์ชั่วคราวที่ไม่จำเป็น

จุดเด่น:

- มีการวิเคราะห์ JavaScript เพื่อตรวจหาช่องโหว่
- การใช้ OpenAI API เพื่อวิเคราะห์ข้อมูลเชิงลึก
- ระบบมีการจัดการไฟล์อย่างมีประสิทธิภาพ

1. การตั้งค่าตัวแปร (Set_Variable1)

ในขั้นตอนนี้ ระบบจะรับข้อมูลจากผู้ใช้ เช่น URL และระดับการสแกน จากนั้นจะตรวจสอบการเชื่อมต่อไปยังเว็บไซต์

```
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
```

อธิบาย:

- **รับ URL จากผู้ใช้ และเพิ่ม https:// หากไม่ได้ใส่มา**
- **ตรวจสอบการเชื่อมต่อไปยัง URL นั้น หากเชื่อมต่อสำเร็จ จะให้ผู้ใช้เลือกระดับการสแกน (1-3)**

---

2. การตั้งค่าชื่อไฟล์ (file_name_i)

ฟังก์ชันนี้มีการตั้งชื่อไฟล์สำหรับการบันทึกผลลัพธ์ที่ได้จากการสแกน โดยจะลบส่วนขยายของไฟล์และจัดรูปแบบชื่อไฟล์ให้สอดคล้อง

```
class file_name_i():

    # ตั้งชื่อไฟล์และจัดรูปแบบ

    file_name = urlparse(url).netloc.replace('.', '_') + ".txt"
```

อธิบาย:

- **ระบบจะสร้างชื่อไฟล์จาก URL โดยใช้ส่วน netloc ของ URL และแทนที่เครื่องหมาย . ด้วย _ เพื่อเป็นชื่อไฟล์ .txt**

---

3. ฟังก์ชันสแกน URL (URL_Finder)

ฟังก์ชันนี้ทำหน้าที่สแกนลิงก์ภายในเว็บไซต์ตามระดับที่กำหนด

```
def URL_Finder(url, max_recursion_level, file_name):
    def get_links(soup, base_url):
        links = []
        for a_tag in soup.find_all('a', href=True):
            link = a_tag['href']
            if not link.startswith('http'):
                link = requests.compat.urljoin(base_url, link)
            links.append(link)
        return links

    to_visit = [url]
    visited = set()

    with open(file_name, 'w') as f:
        recursion_level = 0
        while to_visit and recursion_level < max_recursion_level:
            current_url = to_visit.pop(0)
            if current_url not in visited:
                visited.add(current_url)
                try:
                    response = requests.get(current_url)
                    soup = BeautifulSoup(response.text, 'html.parser')
                    f.write(current_url + '\n')
                    new_links = get_links(soup, current_url)
                    to_visit.extend(new_links)
                    recursion_level += 1
```

```
            except requests.RequestException:
                pass
```

อธิบาย:

- **ฟังก์ชัน URL_Finder จะสแกนลิงก์ภายในเว็บไซต์**
- **ใช้ BeautifulSoup เพื่อดึงลิงก์จาก HTML ของเว็บไซต์ และเพิ่มลิงก์ที่พบลงในไฟล์ผลลัพธ์**
- **ใช้ลูปเพื่อทำการสแกนลิงก์ซ้ำจนกว่าจะถึงระดับการสแกนที่กำหนด**

---

4. การวิเคราะห์ไฟล์ JavaScript (Javascript)

ในขั้นตอนนี้จะเป็นการสแกนไฟล์ JavaScript ภายในเว็บไซต์เพื่อวิเคราะห์หาช่องโหว่

```
def Javascript(file_name, instructions):
    # เปิดไฟล์ JavaScript ที่ระบุและทำการวิเคราะห์
    with open(file_name, 'r') as js_file:
        js_code = js_file.read()

    # ส่งคำสั่งการวิเคราะห์ไปยัง OpenAI API
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=f"{instructions}\n\n{js_code}",
        max_tokens=1000
    )

    # พิมพ์ผลลัพธ์การวิเคราะห์
    print(response.choices[0].text.strip())
```

อธิบาย:

- **อ่านโค้ดจากไฟล์ JavaScript ที่เก็บไว้ใน file_name**
- **ใช้ OpenAI API เพื่อวิเคราะห์โค้ด โดยส่งโค้ด JavaScript พร้อมคำแนะนำ (instructions) ไปยัง OpenAI**
- **ผลลัพธ์จะถูกพิมพ์ออกมาหรือบันทึกไว้**

---

5. การใช้ OpenAI API

ฟังก์ชันนี้เป็นการใช้ OpenAI API ในการวิเคราะห์ข้อมูลเพิ่มเติม

```
def chatGPT_API(file_name, api_key):
    # ใช้ OpenAI API เพื่อวิเคราะห์ข้อมูลเพิ่มเติมจากไฟล์
    openai.api_key = api_key
    with open(file_name, 'r') as f:
        file_data = f.read()

    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt="Analyze this file for vulnerabilities:\n\n" + file_data,
        max_tokens=1000
    )
    print(response.choices[0].text.strip())
```

อธิบาย:

- **ใช้ API ของ OpenAI เพื่อวิเคราะห์ไฟล์ JavaScript ที่ระบุโดยใช้ API Key ที่ตั้งไว้**
- **ส่งโค้ดในไฟล์ไปยัง OpenAI เพื่อทำการวิเคราะห์หาช่องโหว่ในไฟล์**

---

6. การกรองผลลัพธ์ของ OpenAI (JS_Output_Filtering)

ฟังก์ชันนี้ทำการฟิลเตอร์ผลลัพธ์ที่ได้จาก OpenAI API ให้ดูง่ายขึ้น

```
def JS_Output_Filtering(file_name):
    with open(file_name, 'r') as f:
        data = f.read()

    # ฟิลเตอร์ผลลัพธ์
    filtered_output = "\n".join([line for line in data.splitlines() if "vulnerability" in line])
    print(filtered_output)
```

อธิบาย:

- **อ่านผลลัพธ์จากไฟล์ที่บันทึกไว้ แล้วทำการฟิลเตอร์เฉพาะบรรทัดที่มีข้อมูลเกี่ยวกับช่องโหว่**
- **แสดงผลลัพธ์ที่ฟิลเตอร์แล้วออกมา**

---

7. การลบไฟล์ที่ไม่จำเป็น (clean_up_files)

ฟังก์ชันนี้จะลบไฟล์ที่ไม่จำเป็นออกจากระบบเพื่อทำความสะอาด

```
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)
```

อธิบาย:

- **ฟังก์ชันนี้ทำการลบไฟล์ .txt ทั้งหมดที่ไม่ใช่ไฟล์ API_Key.txt หรือไฟล์ที่กำหนดไว้ใน Clean_up_file_name เพื่อทำความสะอาดไฟล์ที่ไม่จำเป็นออก**

จากระบบ

สรุป:

แต่ละฟังก์ชันในโค้ดนี้ทำงานร่วมกันเพื่อสร้างเครื่องมือสแกนช่องโหว่ในเว็บไซต์ โดยใช้การสแกนลิงก์, การวิเคราะห์ JavaScript และการใช้ OpenAI API ในการวิเคราะห์เชิงลึก