จากการวิเคราะห์โค้ดที่คุณส่งมา โค้ดนี้เป็นเครื่องมือสแกนหาช่องโหว่ที่มีฟังก์ชันหลายอย่าง เช่น การติดตามลิงก์ URL, การวิเคราะห์ JavaScript, และการทำงานร่วมกับ OpenAI API เพื่อช่วยในการประมวลผล นอกจากนี้ยังมีการจัดการไฟล์สำหรับเก็บผลลัพธ์และทำความสะอาดไฟล์ที่ไม่จำเป็นหลังจากการประมวลผล

ข้อเสนอแนะในการเพิ่มความสามารถในการตรวจจับช่องโหว่ของ SQL Injection

1. การใช้ Pattern Matching เพื่อตรวจจับคำสั่ง SQL ที่อาจเป็นอันตราย: คุณสามารถใช้ Regular Expressions (regex) เพื่อตรวจหาคำสั่ง SQL ที่พบได้บ่อย เช่น SELECT, INSERT, UPDATE, DELETE หรือคำสั่ง WHERE ภายในโค้ด โดยตรวจสอบว่ามีการส่งผ่าน input ที่ไม่ได้รับการป้องกันเข้าสู่คำสั่งเหล่านี้หรือไม่ ซึ่งอาจนำไปสู่ SQL Injection

ตัวอย่างโค้ด:

```
sql_patterns = [
    r'\bSELECT\b.*\bFROM\b',
    r'\bINSERT\b.*\bINTO\b',
    r'\bUPDATE\b.*\bSET\b',
    r'\bDELETE\b.*\bFROM\b',
    r'\bWHERE\b.*=.*'
]
def detect_sql_injection(code):
    for pattern in sql_patterns:
        if re.search(pattern, code, re.IGNORECASE):
            print(f"พบความเป็นไปได้ของ SQL Injection: {pattern}")
```

2.ตรวจสอบการใช้ Parameter Binding: แนะนำให้เพิ่มการตรวจสอบการใช้ Parameter Binding ใน SQL Queries ซึ่งเป็นวิธีหนึ่งในการป้องกัน SQL Injection โดยการตรวจหาโค้ดที่ใช้การเชื่อมต่อข้อความด้วยวิธีที่ไม่ปลอดภัย เช่นการใช้ + หรือ .format() เพื่อประกอบคำสั่ง SQL

ตัวอย่างโค้ด:

```
def check_for_unsafe_concatenation(sql_code):
    if "" in sql_code or """ in sql_code:
```

```
        print("พบการเชื่อมต่อข้อความที่ไม่ปลอดภัย")
```

3.ปรับปรุงการสแกน JavaScript: ถ้า JavaScript ถูกใช้ในการสร้างคำสั่ง SQL ฝั่งไคลเอนต์ (เช่นผ่าน WebSQL หรือไลบรารี SQL อื่นๆ) เครื่องมือนี้ควรจะสามารถสแกนหาช่องโหว่ในกรณีนี้ได้ โดยการเพิ่มฟังก์ชันที่ตรวจจับการใช้คำสั่ง SQL ใน JavaScript

4.การใช้ OpenAI เพื่อวิเคราะห์คำสั่ง SQL: เนื่องจากโค้ดมีการเชื่อมต่อกับ OpenAI API คุณสามารถใช้ API นี้เพื่อตรวจสอบและวิเคราะห์คำสั่ง SQL และพิจารณาว่าคำสั่งเหล่านั้นมีความเสี่ยงต่อ SQL Injection หรือไม่

ตัวอย่างโค้ด:

```
def use_openai_for_sql_detection(sql_code):
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=f"วิเคราะห์คำสั่ง SQL นี้ว่ามีความเสี่ยงต่อการถูกโจมตี SQL Injection หรือไม่: {sql_code}",
        max_tokens=100
    )
    print(response.choices[0].text.strip())
```

สรุป:

คุณสามารถเพิ่มฟังก์ชันในการตรวจจับช่องโหว่ SQL ได้โดยการใช้ Regular Expressions เพื่อค้นหาคำสั่ง SQL ที่น่าสงสัย การตรวจสอบการใช้ Parameter Binding และการวิเคราะห์คำสั่ง SQL โดยใช้ OpenAI API ซึ่งจะช่วยให้เครื่องมือนี้มีความสามารถในการตรวจจับ SQL Injection ที่ดีขึ้น

โค้ดสร้าง GUI ด้วย Tkinter :

```
import tkinter as tk
from tkinter import messagebox
import re
import requests
```

```
import os

# ฟังก์ชันสำหรับสแกนช่องโหว่ SQL Injection
def detect_sql_injection(code):
    sql_patterns = [
        r'\bSELECT\b.*\bFROM\b',
        r'\bINSERT\b.*\bINTO\b',
        r'\bUPDATE\b.*\bSET\b',
        r'\bDELETE\b.*\bFROM\b',
        r'\bWHERE\b.*=.*'
    ]
    for pattern in sql_patterns:
        if re.search(pattern, code, re.IGNORECASE):
            return f"พบความเป็นไปได้ของ SQL Injection: {pattern}"
    return "ไม่พบ SQL Injection"

# ฟังก์ชันสำหรับสแกน URL
def scan_url():
    url = url_entry.get().strip()

    if not (url.startswith("http://") or url.startswith("https://")):
        url = "https://" + url

    try:
        response = requests.get(url)
        if response.status_code == 200:
```

```
            result_text.insert(tk.END, f"การเชื่อมต่อสำเร็จ: {url}\n")
            # ตรงนี้สามารถเพิ่มการติดตามลิงก์ หรือทำงานสแกนลึกๆ ได้ตามฟังก์ชันที่คุณมี
            result_text.insert(tk.END, "การสแกน URL เริ่มขึ้น...\n")
        else:
            result_text.insert(tk.END, "การเชื่อมต่อไม่สำเร็จ\n")
    except requests.exceptions.RequestException:
        result_text.insert(tk.END, "ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้\n")

# ฟังก์ชันสำหรับการสแกน SQL Injection
def scan_sql():
    code_to_scan = code_entry.get("1.0", tk.END)
    result = detect_sql_injection(code_to_scan)
    result_text.insert(tk.END, result + "\n")

# ฟังก์ชันสำหรับการล้างผลลัพธ์
def clear_results():
    result_text.delete("1.0", tk.END)

# ฟังก์ชันสำหรับการแสดงข้อความ About
def show_about():
    messagebox.showinfo("About", "โปรแกรมนี้เป็นเครื่องมือสำหรับสแกนหาช่องโหว่ SQL Injection และ URL ภายในเว็บไซต์")

# สร้างหน้าต่างหลัก
root = tk.Tk()
root.title("Vulnerability Scanner Tool")
root.geometry("600x400")
```

```
# กรอบ URL
url_frame = tk.Frame(root)
url_frame.pack(pady=10)

url_label = tk.Label(url_frame, text="ใส่ URL:")
url_label.pack(side=tk.LEFT)

url_entry = tk.Entry(url_frame, width=50)
url_entry.pack(side=tk.LEFT, padx=5)

scan_url_button = tk.Button(url_frame, text="สแกน URL", command=scan_url)
scan_url_button.pack(side=tk.LEFT)

# กรอบ SQL Injection
sql_frame = tk.Frame(root)
sql_frame.pack(pady=10)

code_label = tk.Label(sql_frame, text="ใส่โค้ดที่ต้องการสแกน SQL:")
code_label.pack()

code_entry = tk.Text(sql_frame, height=5, width=50)
code_entry.pack()

scan_sql_button = tk.Button(sql_frame, text="สแกน SQL Injection",
command=scan_sql)
scan_sql_button.pack(pady=5)
```

```
# แสดงผลการสแกน
result_frame = tk.Frame(root)
result_frame.pack(pady=10)

result_label = tk.Label(result_frame, text="ผลการสแกน:")
result_label.pack()

result_text = tk.Text(result_frame, height=10, width=70)
result_text.pack()

# ปุ่มล้างผลลัพธ์
clear_button = tk.Button(root, text="ล้างผลลัพธ์", command=clear_results)
clear_button.pack(pady=5)

# เมนู About
menu = tk.Menu(root)
root.config(menu=menu)

help_menu = tk.Menu(menu)
menu.add_cascade(label="Help", menu=help_menu)
help_menu.add_command(label="About", command=show_about)

root.mainloop()
```

อธิบายการใช้งาน:

1. ส่วนกรอก URL:
   - ที่ด้านบนของหน้าต่าง คุณจะเห็นช่องให้ใส่ URL ของเว็บไซต์ที่คุณต้องการสแกน ช่องนี้รองรับ URL ที่เริ่มต้นด้วย http:// หรือ https:// ถ้าไม่ใส่ ตัวโปรแกรมจะเพิ่มให้โดยอัตโนมัติ
   - เมื่อใส่ URL เสร็จแล้ว กดปุ่ม "สแกน URL" เพื่อให้ระบบสแกนการเชื่อมต่อไปยัง URL นั้น โดยจะแสดงผลในส่วน "ผลการสแกน"
2. ส่วนกรอกโค้ด SQL:
   - ใต้ช่อง URL จะมีกรอบให้คุณกรอกโค้ด SQL หรือโค้ดที่ต้องการตรวจสอบสำหรับ SQL Injection
   - เมื่อกรอกเสร็จ ให้กดปุ่ม "สแกน SQL Injection" เพื่อให้ระบบตรวจสอบว่ามีช่องโหว่ SQL Injection ในโค้ดที่ใส่หรือไม่ โดยผลจะถูกแสดงในช่อง "ผลการสแกน"
3. ผลการสแกน:
   - หลังจากที่คุณทำการสแกน ไม่ว่าจะเป็น URL หรือ SQL Injection ผลลัพธ์จะถูกแสดงในช่องข้อความด้านล่าง
   - คุณสามารถกดปุ่ม "ล้างผลลัพธ์" เพื่อล้างข้อความทั้งหมด
4. เมนู About:
   - คุณสามารถคลิกที่เมนู "Help" ด้านบนและเลือก "About" เพื่อแสดงข้อมูลเกี่ยวกับโปรแกรม

ฟังก์ชันการทำงาน:

- ระบบจะทำการสแกนการเชื่อมต่อ URL และตรวจสอบว่าเว็บไซต์ตอบกลับหรือไม่
- สำหรับการสแกน SQL Injection ระบบจะใช้ Regular Expressions เพื่อตรวจหาคำสั่ง SQL ที่อาจเป็นอันตราย