โค้ดปรับปรุงล่าสุด

1. **เพิ่มการรองรับ IP Domain ในการตรวจสอบ URL: ระบบจะตรวจสอบว่า URL ที่ป้อนมานั้นเป็นชื่อโดเมนหรือ IP Address แล้วทำการประมวลผลได้**
2. **เพิ่มการตรวจจับช่องโหว่ SQL Injection: เพิ่มการตรวจสอบการประกอบคำสั่ง SQL ที่ไม่ปลอดภัย**
3. **เพิ่มการตรวจจับช่องโหว่ CSRF: เพิ่มฟังก์ชันการสแกนเพื่อตรวจจับช่องโหว่ CSRF จากฟอร์มที่อยู่ในเว็บไซต์**
4. **ปรับปรุง GUI ให้สวยงามขึ้น: ปรับปรุงรูปแบบ GUI ให้ดูเรียบง่ายและใช้งานสะดวกขึ้น**

โค้ดที่ปรับปรุงแล้ว:

```
import tkinter as tk
from tkinter import messagebox
import re
import requests
import os
from urllib.parse import urlparse

# ตรวจสอบว่าเป็น IP Address หรือไม่
def is_ip_address(url):
    return re.match(r"^\d{1,3}(\.\d{1,3}){3}$", url) is not None

# 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"
```

```
# ฟังก์ชันสำหรับตรวจจับช่องโหว่ CSRF
def detect_csrf(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            if "csrf" in response.text.lower():
                return "พบช่องโหว่ CSRF"
            else:
                return "ไม่พบช่องโหว่ CSRF"
        else:
            return "ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้"
    except requests.RequestException:
        return "ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้"

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

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

    if is_ip_address(urlparse(url).netloc):
        result_text.insert(tk.END, "กำลังสแกน IP Address...\n")
```

```
    else:
        result_text.insert(tk.END, "กำลังสแกนโดเมน...\n")

    try:
        response = requests.get(url)
        if response.status_code == 200:
            result_text.insert(tk.END, f"การเชื่อมต่อสำเร็จ: {url}\n")
            csrf_result = detect_csrf(url)
            result_text.insert(tk.END, csrf_result + "\n")
        else:
            result_text.insert(tk.END, "การเชื่อมต่อไม่สำเร็จ\n")
    except requests.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 CSRF ภายในเว็บไซต์")

# สร้างหน้าต่างหลัก
root = tk.Tk()
root.title("Vulnerability Scanner Tool")
root.geometry("700x500")

# ปรับแต่งสไตล์ของ GUI ให้สวยงาม
root.configure(bg="#f0f0f0")

# กรอบ URL
url_frame = tk.Frame(root, bg="#f0f0f0")
url_frame.pack(pady=10)

url_label = tk.Label(url_frame, text="ใส่ URL หรือ IP:", bg="#f0f0f0")
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, bg="#f0f0f0")
sql_frame.pack(pady=10)


code_label = tk.Label(sql_frame, text="ใส่โค้ดที่ต้องการสแกน SQL:", bg="#f0f0f0")
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, bg="#f0f0f0")
result_frame.pack(pady=10)


result_label = tk.Label(result_frame, text="ผลการสแกน:", bg="#f0f0f0")
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)
```

อธิบายการปรับปรุง:

1. **การเพิ่มฟังก์ชันตรวจสอบ IP Domain:** ฟังก์ชัน `is_ip_address` จะตรวจสอบว่า URL ที่ผู้ใช้ป้อนเป็น IP Address หรือไม่ หากเป็น จะมีการแสดงข้อความว่ากำลังสแกน IP Address
2. **เพิ่มการตรวจจับ CSRF:** ฟังก์ชัน `detect_csrf` จะตรวจสอบเว็บไซต์ว่ามีช่องโหว่ CSRF หรือไม่ โดยการค้นหาคำว่า "csrf" ใน HTML ของหน้าเว็บ
3. **การปรับปรุง GUI:** เพิ่มการตกแต่ง GUI ด้วยสีพื้นหลังและการจัดเรียงองค์ประกอบให้สวยงาม

โค้ดนี้ทำให้โปรแกรมสแกนช่องโหว่มีความสามารถในการตรวจจับทั้ง SQL Injection และ CSRF รวมถึงสามารถสแกนเว็บไซต์ที่เป็น IP Address ได้ด้วย

โค้ดที่ปรับปรุงนี้มีการทำงานที่หลากหลาย โดยเพิ่มการตรวจจับทั้ง SQL Injection, CSRF, และการรองรับ URL ที่เป็น IP Address รวมถึงการปรับปรุง GUI ให้สวยงามขึ้น

ขั้นตอนการทำงานของโค้ด:

1. **รับ URL หรือ IP จากผู้ใช้:**
   - **ผู้ใช้จะกรอก URL หรือ IP Address ลงในช่องที่กำหนดใน GUI**
   - **ระบบจะตรวจสอบว่า URL ที่ป้อนมาเป็นชื่อโดเมนหรือ IP Address โดยใช้ฟังก์ชัน is_ip_address เพื่อแยกแยะระหว่าง IP Address และโดเมน**

```
def is_ip_address(url):
    return re.match(r"^\d{1,3}(\.\d{1,3}){3}$", url) is not None
```

2.การสแกน URL:

- **เมื่อผู้ใช้กดปุ่ม "สแกน URL" ระบบจะเริ่มตรวจสอบการเชื่อมต่อกับเว็บไซต์ที่ป้อนมา**
- **ระบบจะใช้ requests.get(url) เพื่อตรวจสอบว่าการเชื่อมต่อสำเร็จหรือไม่**
- **หาก URL เป็น IP Address ระบบจะแสดงข้อความว่า "กำลังสแกน IP Address"**
- **หาก URL เป็นโดเมน ระบบจะแสดงข้อความว่า "กำลังสแกนโดเมน"**

```
def scan_url():
    url = url_entry.get().strip()
    if not (url.startswith("http://") or url.startswith("https://")):
        url = "https://" + url

    if is_ip_address(urlparse(url).netloc):
        result_text.insert(tk.END, "กำลังสแกน IP Address...\n")
    else:
        result_text.insert(tk.END, "กำลังสแกนโดเมน...\n")

    try:
        response = requests.get(url)
        if response.status_code == 200:
            result_text.insert(tk.END, f"การเชื่อมต่อสำเร็จ: {url}\n")
            csrf_result = detect_csrf(url)
            result_text.insert(tk.END, csrf_result + "\n")
        else:
            result_text.insert(tk.END, "การเชื่อมต่อไม่สำเร็จ\n")
    except requests.RequestException:
        result_text.insert(tk.END, "ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้\n")
```

3.การตรวจจับช่องโหว่ CSRF:

- **ฟังก์ชัน detect_csrf จะทำการตรวจสอบ HTML ของหน้าเว็บที่โหลดขึ้นมาจาก URL ว่ามีคำว่า "csrf" อยู่หรือไม่ ซึ่งบ่งบอกถึงการป้องกันหรือช่องโหว่ของ CSRF**
- **หากพบคำว่า "csrf" ใน HTML จะถือว่ามีช่องโหว่ CSRF และจะแสดงข้อความว่า "พบช่องโหว่ CSRF"**

```
def detect_csrf(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            if "csrf" in response.text.lower():
                return "พบช่องโหว่ CSRF"
            else:
                return "ไม่พบช่องโหว่ CSRF"
        else:
            return "ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้"
    except requests.RequestException:
        return "ไม่สามารถเชื่อมต่อกับเว็บไซต์ได้"
```

4.การตรวจจับ SQL Injection:

- **ผู้ใช้สามารถป้อนโค้ด SQL ลงในช่องที่กำหนด จากนั้นเมื่อกดปุ่ม "สแกน SQL Injection" ระบบจะทำการวิเคราะห์คำสั่ง SQL ที่ป้อน**
- **ระบบใช้ Regular Expressions เพื่อตรวจหาคำสั่ง SQL ที่อาจเป็นอันตราย เช่น SELECT, INSERT, UPDATE, DELETE, และ WHERE**

- หากพบรูปแบบคำสั่ง SQL ที่น่าสงสัย ระบบจะแสดงผลว่า "พบความเป็นไปได้ของ 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"
```

การแสดงผลลัพธ์:

- หลังจากที่ผู้ใช้ทำการสแกน URL หรือ SQL Injection ผลลัพธ์จะแสดงในกรอบผลลัพธ์ ซึ่งจะแสดงผลการตรวจสอบการเชื่อมต่อ การตรวจจับ CSRF หรือการตรวจจับ SQL Injection
- หากต้องการล้างผลลัพธ์ ผู้ใช้สามารถกดปุ่ม "ล้างผลลัพธ์" เพื่อเคลียร์ข้อมูลที่แสดง

การปรับปรุงหน้าต่าง GUI:

- ระบบ GUI ใช้ Tkinter ในการสร้างหน้าต่างหลักที่มีองค์ประกอบต่างๆ เช่น ช่องกรอกข้อมูล ปุ่มสแกน ปุ่มล้าง และเมนูช่วยเหลือ

- เพิ่มการตกแต่งสีพื้นหลังและการจัดวางองค์ประกอบให้สวยงามและเรียบง่าย

```
root = tk.Tk()

root.title("Vulnerability Scanner Tool")

root.geometry("700x500")

root.configure(bg="#f0f0f0")
```

สรุปขั้นตอนการทำงาน:

1. ผู้ใช้กรอก URL หรือ IP และกดปุ่มสแกน ระบบจะตรวจสอบการเชื่อมต่อและสแกนหา CSRF
2. หากผู้ใช้ต้องการตรวจสอบ SQL Injection สามารถป้อนโค้ด SQL ลงในช่องที่กำหนดและกดปุ่มสแกน
3. ผลลัพธ์จะแสดงในหน้าต่าง GUI และสามารถล้างผลลัพธ์ได้