1. การตรวจสอบ Parameter Binding:

เราจะตรวจสอบการสร้างคำสั่ง SQL ว่ามีการใช้วิธีการประกอบคำสั่งที่ปลอดภัย เช่น การใช้ Parameter Binding หรือตรวจสอบการใช้การเชื่อมข้อความที่ไม่ปลอดภัยใน SQL Queries

2. การใช้ OpenAI API:

เราจะเพิ่มการวิเคราะห์คำสั่ง SQL โดยการส่งคำสั่ง SQL ไปที่ OpenAI API เพื่อให้โมเดลวิเคราะห์และบอกว่ามีช่องโหว่ SQL Injection หรือไม่

ตัวอย่างโค้ดที่เพิ่มทั้งสองฟังก์ชัน:

```
import tkinter as tk

from tkinter import messagebox

import re

import requests

import openai  # ต้องติดตั้ง openai ผ่าน pip

import os


# ฟังก์ชันสำหรับตรวจสอบการใช้ Parameter Binding

def check_for_unsafe_concatenation(sql_code):

    # ตรวจสอบการเชื่อมคำสั่ง SQL ด้วยวิธีที่ไม่ปลอดภัย

    if "' in sql_code or '" in sql_code:

        return "พบการประกอบคำสั่ง SQL ที่ไม่ปลอดภัย: การเชื่อมต่อด้วยเครื่องหมาย ' หรือ \""

    return "ไม่พบการประกอบคำสั่ง SQL ที่ไม่ปลอดภัย"


# ฟังก์ชันสำหรับการใช้ OpenAI เพื่อวิเคราะห์ SQL Injection

def use_openai_for_sql_detection(sql_code):

    try:

        # ใช้ OpenAI วิเคราะห์คำสั่ง SQL
```

```
        openai.api_key = "YOUR_API_KEY"  # ใส่ API Key ของคุณที่นี่
        response = openai.Completion.create(
            engine="text-davinci-003",
            prompt=f"Analyze this SQL query for potential injection vulnerabilities: {sql_code}",
            max_tokens=100
        )
        return response.choices[0].text.strip()
    except Exception as e:
        return f"เกิดข้อผิดพลาดในการเชื่อมต่อกับ OpenAI API: {e}"

# 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)

    # ตรวจสอบ SQL Injection แบบพื้นฐาน
    basic_scan_result = detect_sql_injection(code_to_scan)
```

```
    result_text.insert(tk.END, basic_scan_result + "\n")

    # ตรวจสอบการประกอบคำสั่ง SQL ที่ไม่ปลอดภัย
    unsafe_concat_result = check_for_unsafe_concatenation(code_to_scan)
    result_text.insert(tk.END, unsafe_concat_result + "\n")

    # ใช้ OpenAI วิเคราะห์ SQL Injection
    openai_result = use_openai_for_sql_detection(code_to_scan)
    result_text.insert(tk.END, openai_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()
```

การใช้งาน:

1. การตรวจสอบ SQL Injection:
   - ในช่องกรอกโค้ด SQL คุณสามารถใส่คำสั่ง SQL ที่ต้องการตรวจสอบ
   - เมื่อกดปุ่ม "สแกน SQL Injection" ระบบจะทำการ:
     - ตรวจสอบ SQL Injection ด้วยวิธีพื้นฐาน (Regular Expressions)
     - ตรวจสอบการใช้ Parameter Binding ว่ามีการเชื่อมข้อความใน SQL ที่ไม่ปลอดภัยหรือไม่
     - ใช้ OpenAI API เพื่อตรวจสอบคำสั่ง SQL และแจ้งผลการวิเคราะห์ว่ามีช่องโหว่ SQL Injection หรือไม่
2. การเชื่อมต่อกับ OpenAI API:
   - คุณต้องใส่ API Key ของคุณในบรรทัด openai.api_key = "YOUR_API_KEY" ก่อนการใช้งาน
   - เมื่อใช้ OpenAI API ผลการวิเคราะห์คำสั่ง SQL จะถูกแสดงในช่องผลลัพธ์

สรุป:

โค้ดนี้เพิ่มการตรวจสอบทั้งการประกอบคำสั่ง SQL ที่ไม่ปลอดภัย และการใช้ OpenAI API เพื่อทำการวิเคราะห์ SQL Injection ในเชิงลึก ซึ่งจะช่วยให้เครื่องมือมีความสามารถในการตรวจจับช่องโหว่ SQL Injection ที่มีประสิทธิภาพมากยิ่งขึ้น