Port Scanner

"This project is a Python-based Nmap Port Scanner with a graphical user interface (GUI) " "created using Tkinter. The application allows users to scan ports on a target IP address " "using various Nmap scan types, such as TCP SYN, UDP, Intense Scan, and more. The project " "aims to provide a simple way to perform network security checks and detect open ports on " "a remote system."

Let me explain the entire code step by step:

1. Import Libraries

```
import tkinter as tk

from tkinter import messagebox

import nmap
```

- **tkinter**: Used to create the graphical interface (buttons, text fields, etc.).
- **messagebox**: A part of tkinter used to show error or info messages in pop-up windows.
- **nmap**: A Python wrapper for Nmap, which is a powerful tool for network scanning. We use it to perform different types of scans.

2. scan_ports() Function

This function does the actual scanning when the user clicks "Scan" on the interface.

a) Get User Input:

```
target_ip = target_entry.get()  # Get the IP address entered by the user

ports = port_entry.get()        # Get the port range entered by the user

scan_type = scan_type_var.get()  # Get the selected scan type (from the dropdown menu)
```

- We retrieve the values from the input fields on the GUI for the target IP address, the port range, and the type of scan to perform (TCP, UDP, etc.).

b) Initialize Nmap Scanner:

```
scanner = nmap.PortScanner()
```

- We create an Nmap PortScanner object that will run the actual scan.

c) Define Scan Options:

```
arguments = {
    'TCP SYN': '-sS',
    'TCP Connect': '-sT',
    'TCP ACK': '-sA',
    'TCP Window': '-sW',
    'TCP Masquerade': '-sM',
    'TCP Null': '-sN',
    'TCP FIN': '-sF',
    'TCP Xmas': '-sX',
    'UDP': '-sU',
    'Intense': '-A',
    'Intense Scan Plus': '-A -sV -sC',
    'OS Detection': '-O',
    'Version Detection': '-sV',
    'Script Scan': '-sC',
    'Traceroute': '--traceroute',
}
```

- This is a dictionary where the **key** is the scan type (e.g., 'TCP SYN') and the **value** is the corresponding Nmap option (-sS, -sU, etc.).
- For example, 'UDP': '-sU' means that selecting "UDP" in the dropdown will run Nmap with the -sU option for a UDP scan.

d) Perform the Scan:

```
scanner.scan(target_ip, ports, arguments=arguments[scan_type])
```

- Here we run the scan using:
  - target_ip: The IP address to scan (entered by the user).
  - ports: The port range to scan (entered by the user).
  - arguments[scan_type]: The Nmap option associated with the selected scan type (e.g., -sS for TCP SYN scan).

3. Displaying the Results:

a) Clear Previous Results:

```
results_text.delete(1.0, tk.END)  # Clear the results box before displaying new results
```

- This clears the text box where the previous scan results were displayed, so that the new results can be shown.

b) Display TCP Scan Results:

```
if 'tcp' in scanner[target_ip]:
    results_text.insert(tk.END, "TCP Ports:\n")
    for port in scanner[target_ip]['tcp']:
        state = scanner[target_ip]['tcp'][port]['state']
        results_text.insert(tk.END, f"Port {port}: {state}\n")
```

- If TCP ports are found, the code loops through each port and prints whether it is open or closed.
- The state of the port (open, closed, filtered, etc.) is shown in the text box.

c) Display UDP Scan Results:

```
if 'udp' in scanner[target_ip]:
    results_text.insert(tk.END, "UDP Ports:\n")
    for port in scanner[target_ip]['udp']:
        state = scanner[target_ip]['udp'][port]['state']
        results_text.insert(tk.END, f"Port {port}: {state}\n")
```

- If UDP ports are found, the same thing happens for UDP ports. It lists the state of each scanned UDP port.

4. Handle Errors:

These blocks handle errors that might occur during scanning, and they show the error message in a pop-up window.

a) Nmap Error:

```
except nmap.PortScannerError as e:
   messagebox.showerror("Nmap Error", str(e))
```

- If Nmap itself encounters an error (e.g., if Nmap is not installed properly), this block will show the error message.

b) Key Error:

```
except KeyError:
   messagebox.showerror("Error", f"Scan could not be completed for {target_ip}. The target may be down or unreachable.")
```

- If the target IP is unreachable or invalid, this block will show a message to the user.

c) General Error:

```
except Exception as e:
   messagebox.showerror("Unexpected Error", str(e))
```

- This catches any other unexpected errors and displays them in a message box.

5. Creating the GUI Elements:

a) Main Window:

```
root = tk.Tk()  # Create the main window
root.title("Nmap Port Scanner")  # Set the window title
```

- This creates the main application window and gives it a title ("Nmap Port Scanner").

b) Input Fields and Labels:

```
tk.Label(root, text="Target IP Address:").grid(row=0, column=0, padx=5, pady=5)
target_entry = tk.Entry(root, width=30)
```

```
target_entry.grid(row=0, column=1, padx=5, pady=5)


tk.Label(root, text="Port Range (e.g., 22-80):").grid(row=1, column=0, padx=5, pady=5)

port_entry = tk.Entry(root, width=30)

port_entry.grid(row=1, column=1, padx=5, pady=5)
```

- We create two labels (for "Target IP Address" and "Port Range") and two corresponding entry fields where the user can type their inputs.
- grid() is used to position these elements in the window in a grid-like layout.

c) Dropdown Menu for Scan Type:

```
scan_type_var = tk.StringVar(value='TCP SYN')

scan_type_options = ['TCP SYN', 'TCP Connect', 'TCP ACK', 'TCP Window', 'TCP Masquerade',

              'TCP Null', 'TCP FIN', 'TCP Xmas', 'UDP', 'Intense',

              'Intense Scan Plus', 'OS Detection', 'Version Detection',

              'Script Scan', 'Traceroute']

scan_type_menu = tk.OptionMenu(root, scan_type_var, *scan_type_options)

scan_type_menu.grid(row=2, column=1, padx=5, pady=5)
```

- This creates a dropdown menu (also known as an OptionMenu) with various scan types (like "TCP SYN", "UDP", "Intense").
- The user can choose which scan to perform, and the default selection is set to "TCP SYN."

d) Scan Button:

```
scan_button = tk.Button(root, text="Scan", command=scan_ports)

scan_button.grid(row=3, columnspan=2, padx=5, pady=5)
```

- A button labeled "Scan" is created. When the user clicks it, the scan_ports() function is called to start the scan.

e) Text Box for Results:

```
results_text = tk.Text(root, width=50, height=15)

results_text.grid(row=4, columnspan=2, padx=5, pady=5)
```

- This creates a large text box where the scan results will be displayed.
- The user can see the results (open/closed ports) here after clicking "Scan."

6. Start the Application:

root.mainloop()

- root.mainloop() starts the GUI event loop. This keeps the window open and responsive to user inputs until the user closes it.

Summary of Workflow

1. **User Inputs**: The user enters the target IP, port range, and scan type (e.g., TCP or UDP) in the input fields.
2. **Click Scan**: When the user clicks "Scan", the scan_ports() function is executed.
3. **Run Nmap**: The function performs the network scan based on the user input and selected scan type using Nmap.
4. **Display Results**: The results (open/closed ports) are displayed in the text box.
5. **Handle Errors**: If something goes wrong, appropriate error messages are shown in a pop-up window.

This setup allows the user to perform different types of port scans easily through a graphical interface!