Python Nmap Port Scanner

Contents

Time required: 60 minutes

NOTE: Complete this tutorial in your Kali Linux VM with a bridged adapter or your local Windows computer.

WARNING: Only scan your network or the D1 classroom.

What is Nmap?

Nmap (Network Mapper) is a security scanner, originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich), and used to discover hosts and services on a computer network, thereby building a map of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host(s) and then analyzes their responses.

Some of the useful Nmap features include:

- **Host Discovery**: This enables to identify hosts on any network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular **port** open.
- **Port Scanning**: Enumerating (counting and listing one by one) all the open ports on the target hosts.
- **Version Detection**: Interrogating network services on remote devices to determine application name and version number.

- **OS Detection**: Determining the operating system and hardware characteristics of the network devices.
- **Scriptable Interaction with the target**: Using Nmap Scripting Engine (NSE) and Lua programming language, we can easily write sripts to perform operations on the network devices.

Tutorial 1: Install Nmap

The nmap install now includes the GUI interface for Nmap known as **zenmap**. Nmap is already installed in Kali Linux.

For Windows and Mac OS X, download and install Nmap→ https://nmap.org/download.html

python-nmap

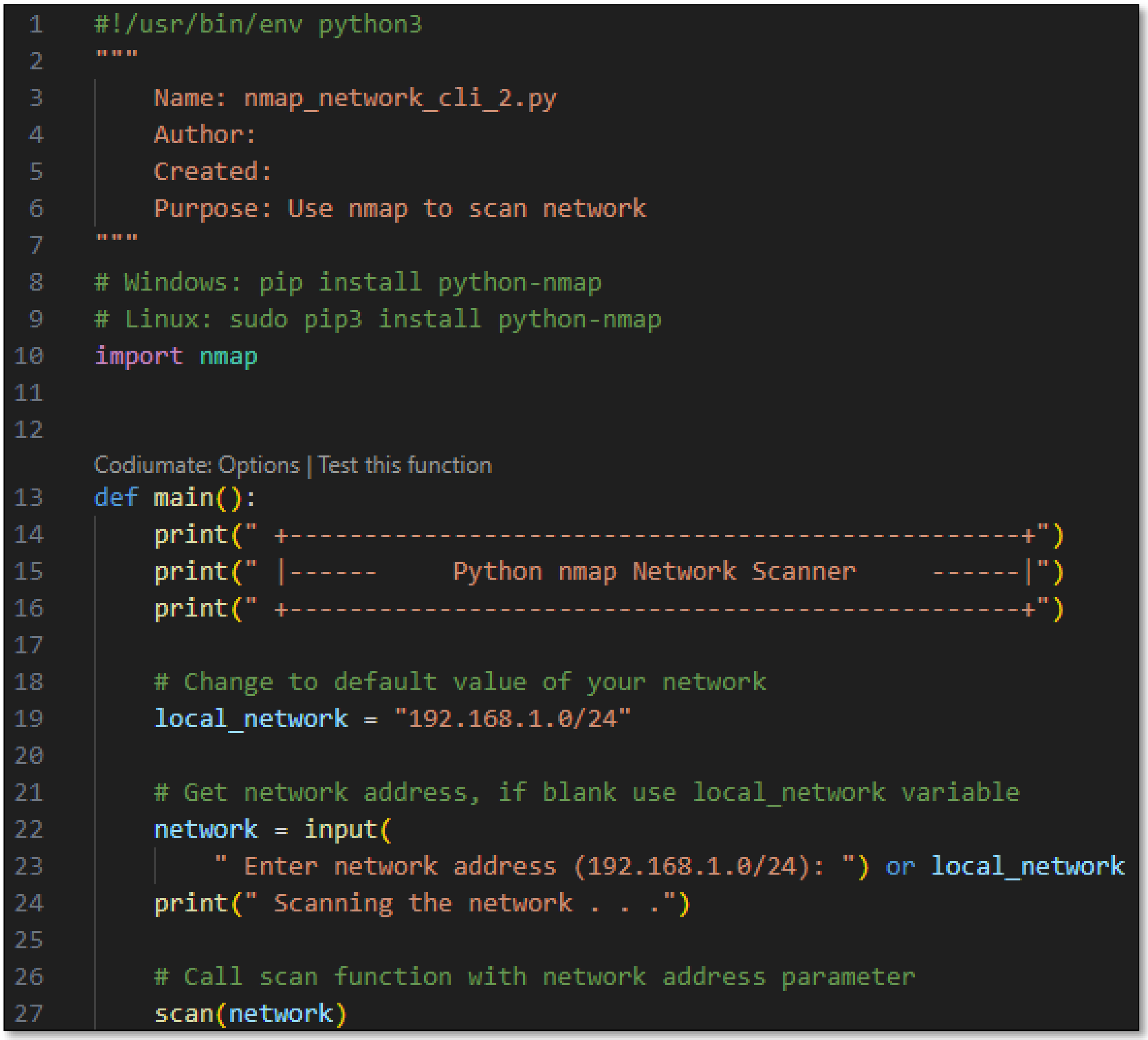
The **python-nmap** library is a wrapper around the **nmap** program. You must have **nmap** installed first. This library allows you to create any type of CLI or GUI program using **nmap**.

Before we start using Nmap, let's install the python-nmap module:

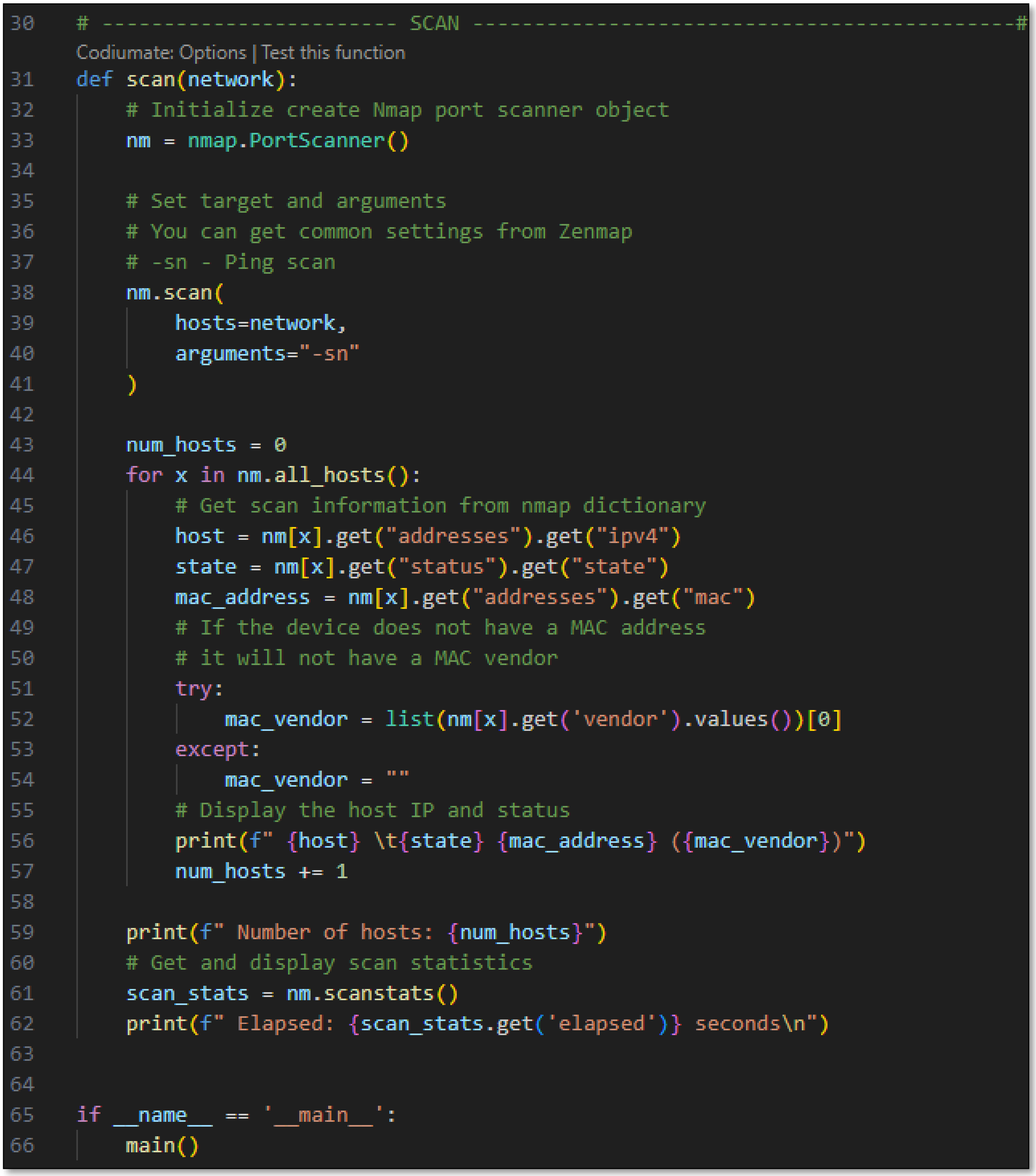
```
# Windows: pip install python-nmap
# Linux: sudo pip3 install python-nmap
```

Tutorial 2: Nmap with Python

Create a Python file named **nmap_ping_scan_cli.py**



```
#!/usr/bin/env python3
"""
    Name: nmap_network_cli_2.py
    Author:
    Created:
    Purpose: Use nmap to scan network
"""
# Windows: pip install python-nmap
# Linux: sudo pip3 install python-nmap
import nmap


Codiumate: Options | Test this function
def main():
    print(" +-------------------------------------------------+")
    print(" |------     Python nmap Network Scanner     ------|")
    print(" +-------------------------------------------------+")

    # Change to default value of your network
    local_network = "192.168.1.0/24"

    # Get network address, if blank use local_network variable
    network = input(
        " Enter network address (192.168.1.0/24): ") or local_network
    print(" Scanning the network . . .")

    # Call scan function with network address parameter
    scan(network)
```



```python
# ------------------------ SCAN --------------------------------------------#
def scan(network):
    # Initialize create Nmap port scanner object
    nm = nmap.PortScanner()

    # Set target and arguments
    # You can get common settings from Zenmap
    # -sn - Ping scan
    nm.scan(
        hosts=network,
        arguments="-sn"
    )

    num_hosts = 0
    for x in nm.all_hosts():
        # Get scan information from nmap dictionary
        host = nm[x].get("addresses").get("ipv4")
        state = nm[x].get("status").get("state")
        mac_address = nm[x].get("addresses").get("mac")
        # If the device does not have a MAC address
        # it will not have a MAC vendor
        try:
            mac_vendor = list(nm[x].get('vendor').values())[0]
        except:
            mac_vendor = ""
        # Display the host IP and status
        print(f" {host} \t{state} {mac_address} ({mac_vendor})")
        num_hosts += 1

    print(f" Number of hosts: {num_hosts}")
    # Get and display scan statistics
    scan_stats = nm.scanstats()
    print(f" Elapsed: {scan_stats.get('elapsed')} seconds\n")


if __name__ == '__main__':
    main()
```

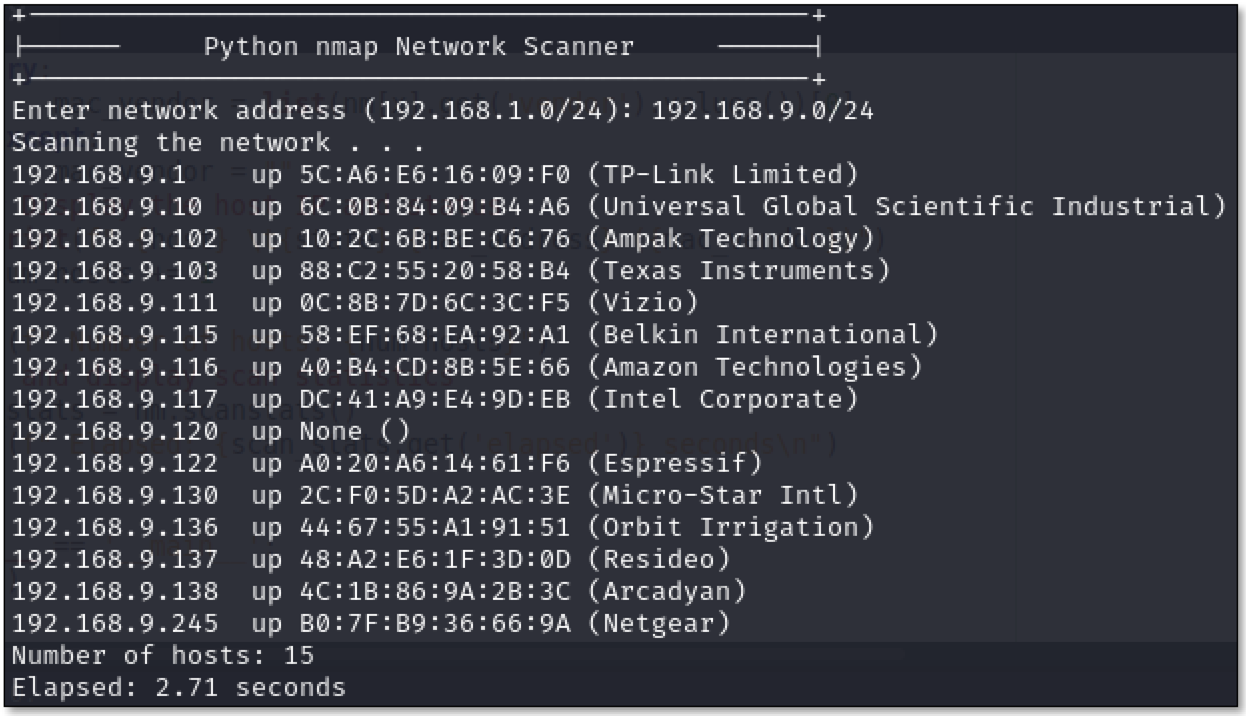
Run the Scan

Use a bridged adapter for network scan from a VM

- In Linux → Terminal → **sudo python3 nmap_ping_scan_cli.py**

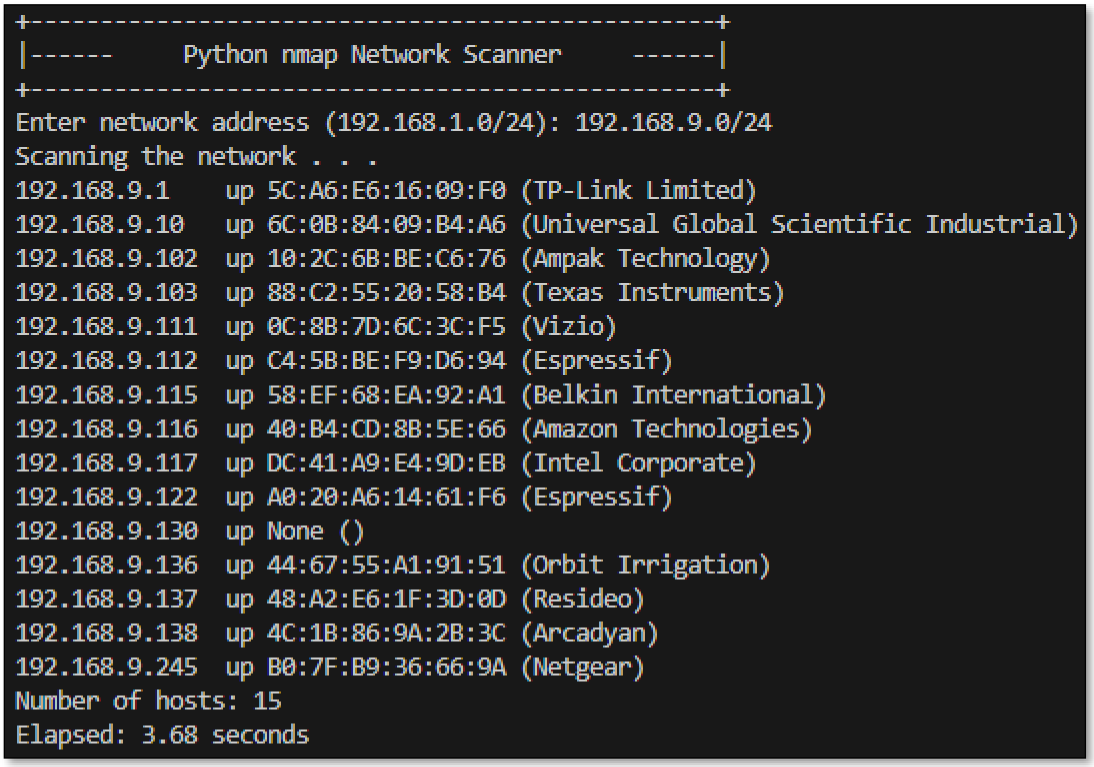
- In Windows -> F5 in VsCode

Example run using bridged adapter on Kali Linux:



```
+──────────────────────────────────────────────────────+
├──────     Python nmap Network Scanner     ──────┤
+──────────────────────────────────────────────────────+
Enter network address (192.168.1.0/24): 192.168.9.0/24
Scanning the network . . .
192.168.9.1    up 5C:A6:E6:16:09:F0 (TP-Link Limited)
192.168.9.10   up 6C:0B:84:09:B4:A6 (Universal Global Scientific Industrial)
192.168.9.102  up 10:2C:6B:BE:C6:76 (Ampak Technology)
192.168.9.103  up 88:C2:55:20:58:B4 (Texas Instruments)
192.168.9.111  up 0C:8B:7D:6C:3C:F5 (Vizio)
192.168.9.115  up 58:EF:68:EA:92:A1 (Belkin International)
192.168.9.116  up 40:B4:CD:8B:5E:66 (Amazon Technologies)
192.168.9.117  up DC:41:A9:E4:9D:EB (Intel Corporate)
192.168.9.120  up None ()
192.168.9.122  up A0:20:A6:14:61:F6 (Espressif)
192.168.9.130  up 2C:F0:5D:A2:AC:3E (Micro-Star Intl)
192.168.9.136  up 44:67:55:A1:91:51 (Orbit Irrigation)
192.168.9.137  up 48:A2:E6:1F:3D:0D (Resideo)
192.168.9.138  up 4C:1B:86:9A:2B:3C (Arcadyan)
192.168.9.245  up B0:7F:B9:36:66:9A (Netgear)
Number of hosts: 15
Elapsed: 2.71 seconds
```

Example on a local Windows computer:



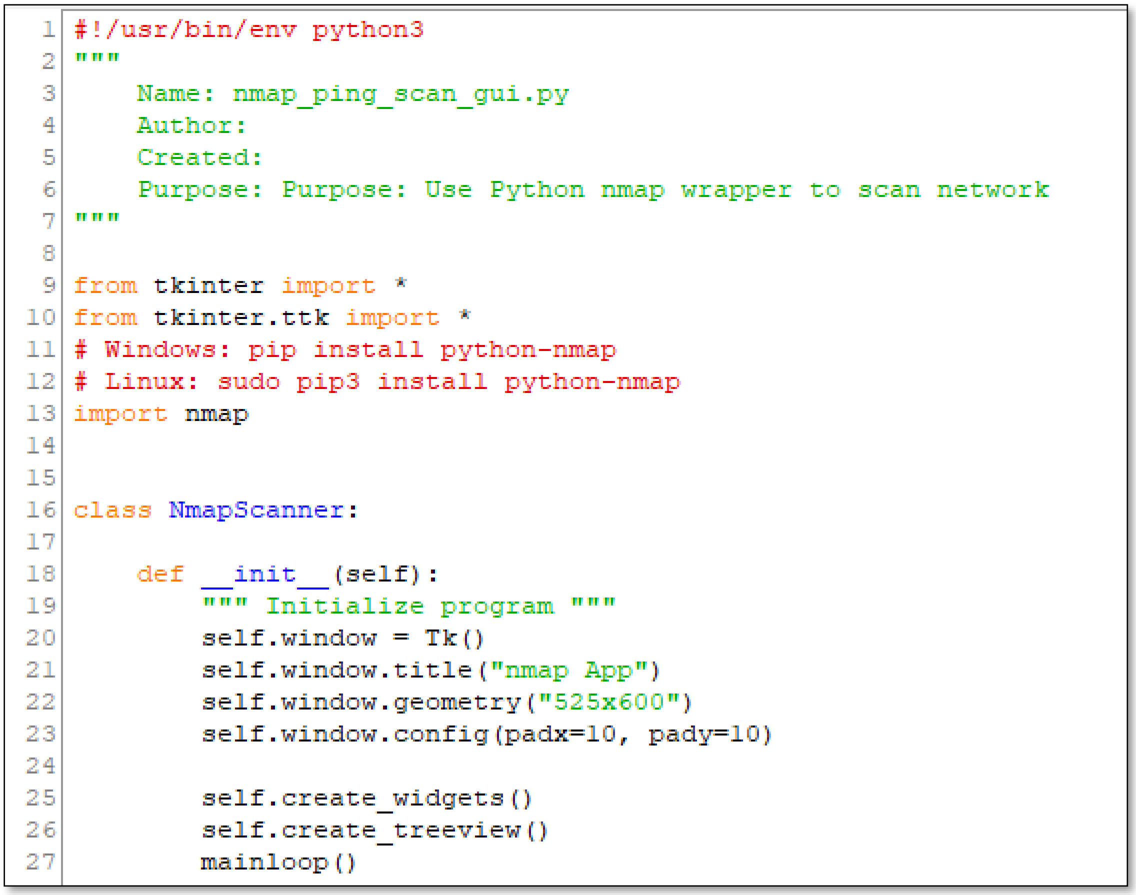
```
+------------------------------------------------+
|------     Python nmap Network Scanner     ------|
+------------------------------------------------+
Enter network address (192.168.1.0/24): 192.168.9.0/24
Scanning the network . . .
192.168.9.1    up 5C:A6:E6:16:09:F0 (TP-Link Limited)
192.168.9.10   up 6C:0B:84:09:B4:A6 (Universal Global Scientific Industrial)
192.168.9.102  up 10:2C:6B:BE:C6:76 (Ampak Technology)
192.168.9.103  up 88:C2:55:20:58:B4 (Texas Instruments)
192.168.9.111  up 0C:8B:7D:6C:3C:F5 (Vizio)
192.168.9.112  up C4:5B:BE:F9:D6:94 (Espressif)
192.168.9.115  up 58:EF:68:EA:92:A1 (Belkin International)
192.168.9.116  up 40:B4:CD:8B:5E:66 (Amazon Technologies)
192.168.9.117  up DC:41:A9:E4:9D:EB (Intel Corporate)
192.168.9.122  up A0:20:A6:14:61:F6 (Espressif)
192.168.9.130  up None ()
192.168.9.136  up 44:67:55:A1:91:51 (Orbit Irrigation)
192.168.9.137  up 48:A2:E6:1F:3D:0D (Resideo)
192.168.9.138  up 4C:1B:86:9A:2B:3C (Arcadyan)
192.168.9.245  up B0:7F:B9:36:66:9A (Netgear)
Number of hosts: 15
Elapsed: 3.68 seconds
```

Extra Credit Tutorial 3: Nmap in Python with GUI

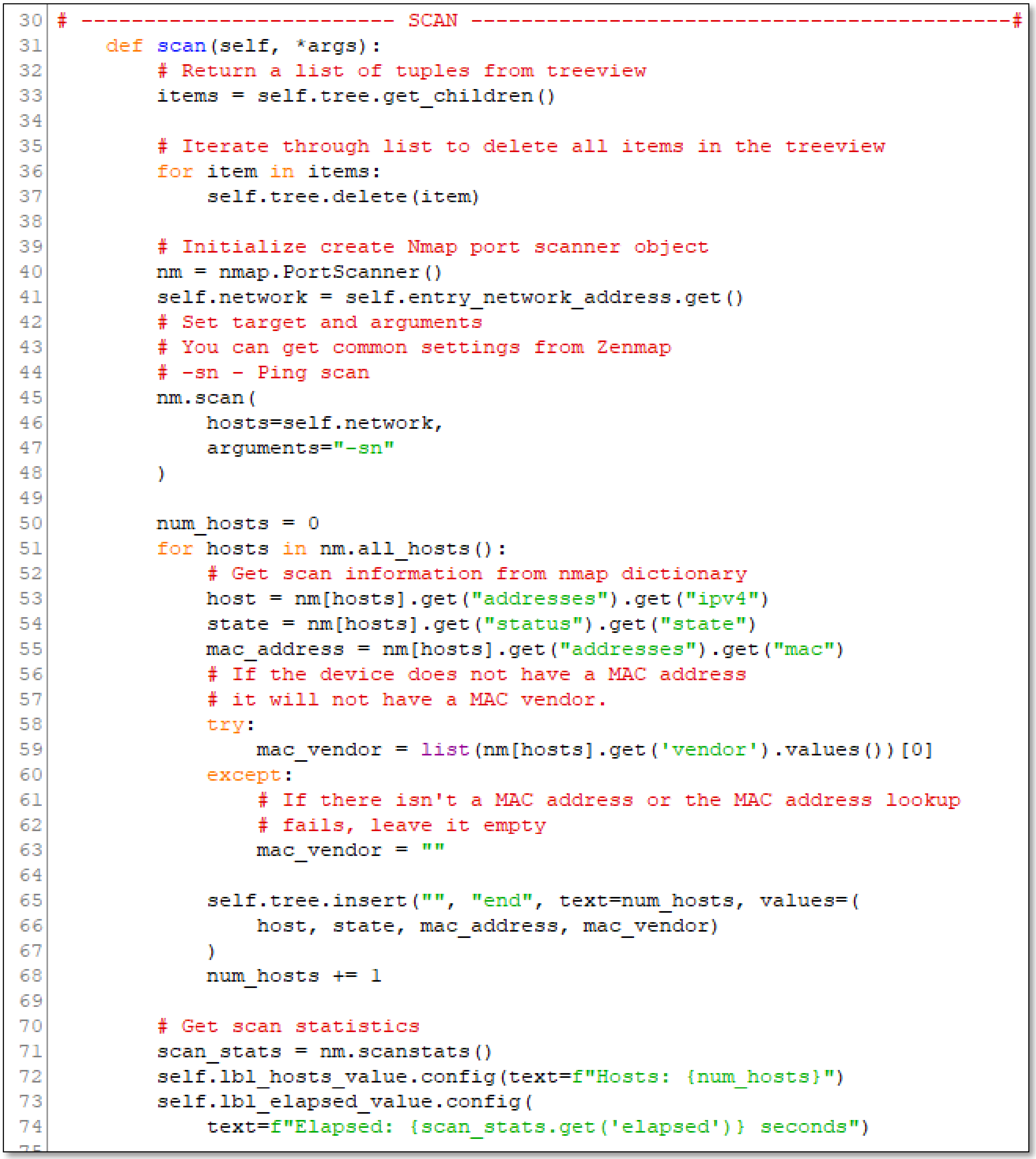
Same program with a Tkinter GUI.

In Linux → Terminal → **sudo python3 nmap_ping_scan_gui.py**

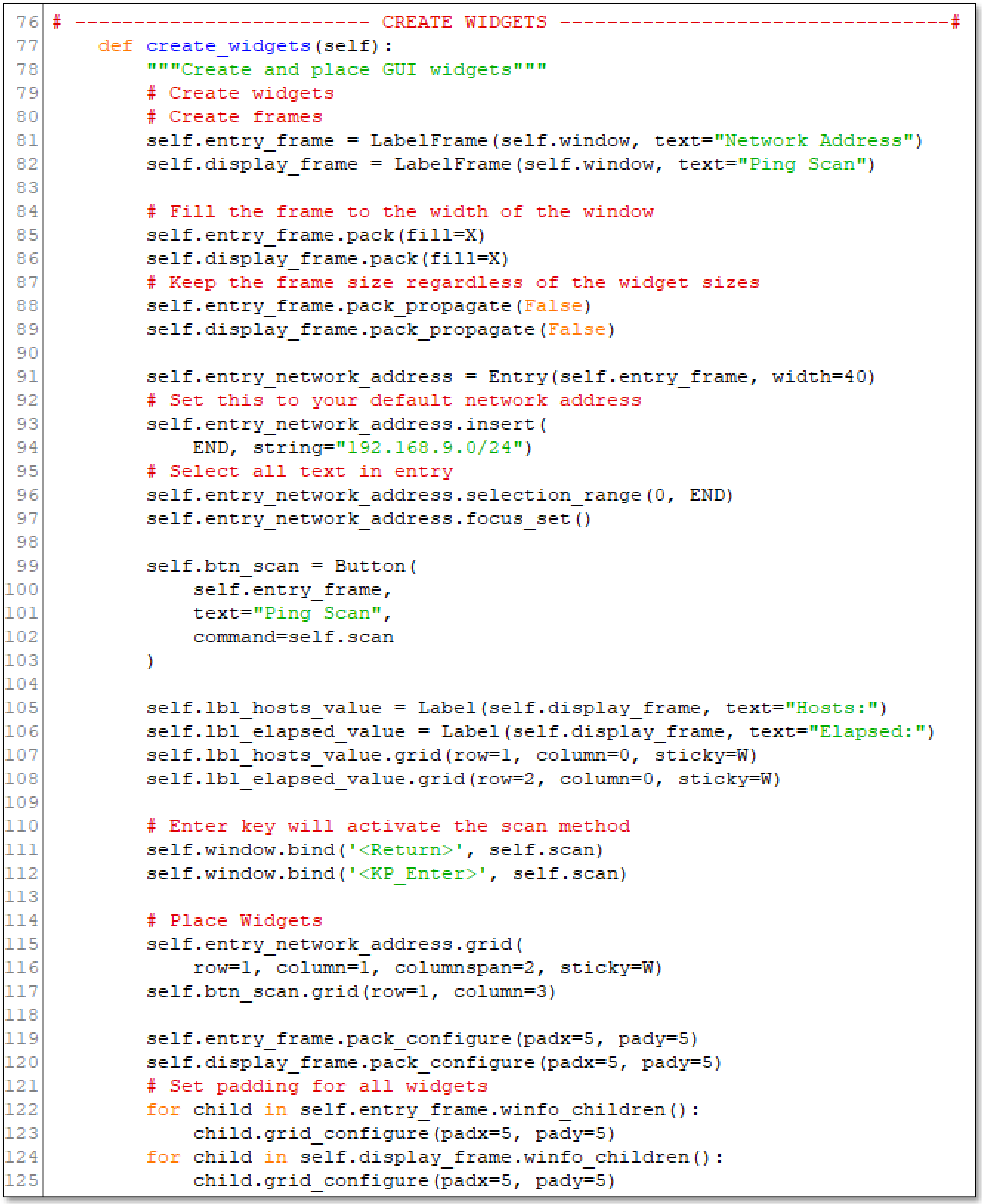
In Windows: F5 in VSCode



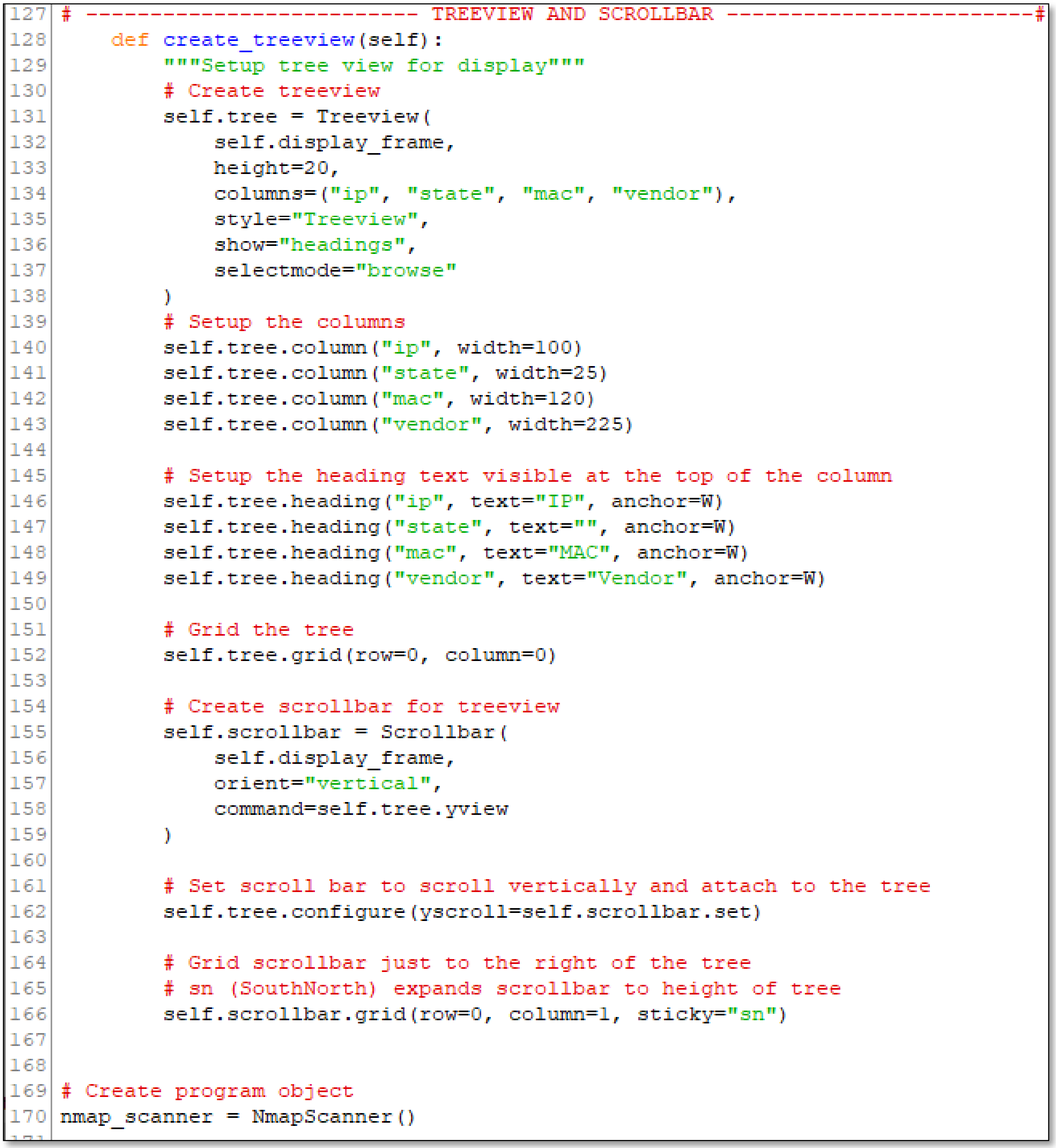
```python
#!/usr/bin/env python3
"""
    Name: nmap_ping_scan_gui.py
    Author:
    Created:
    Purpose: Purpose: Use Python nmap wrapper to scan network
"""

from tkinter import *
from tkinter.ttk import *
# Windows: pip install python-nmap
# Linux: sudo pip3 install python-nmap
import nmap


class NmapScanner:

    def __init__(self):
        """ Initialize program """
        self.window = Tk()
        self.window.title("nmap App")
        self.window.geometry("525x600")
        self.window.config(padx=10, pady=10)

        self.create_widgets()
        self.create_treeview()
        mainloop()
```



```python
# -------------------------- SCAN ------------------------------------------#
    def scan(self, *args):
        # Return a list of tuples from treeview
        items = self.tree.get_children()

        # Iterate through list to delete all items in the treeview
        for item in items:
            self.tree.delete(item)

        # Initialize create Nmap port scanner object
        nm = nmap.PortScanner()
        self.network = self.entry_network_address.get()
        # Set target and arguments
        # You can get common settings from Zenmap
        # -sn - Ping scan
        nm.scan(
            hosts=self.network,
            arguments="-sn"
        )

        num_hosts = 0
        for hosts in nm.all_hosts():
            # Get scan information from nmap dictionary
            host = nm[hosts].get("addresses").get("ipv4")
            state = nm[hosts].get("status").get("state")
            mac_address = nm[hosts].get("addresses").get("mac")
            # If the device does not have a MAC address
            # it will not have a MAC vendor.
            try:
                mac_vendor = list(nm[hosts].get('vendor').values())[0]
            except:
                # If there isn't a MAC address or the MAC address lookup
                # fails, leave it empty
                mac_vendor = ""

            self.tree.insert("", "end", text=num_hosts, values=(
                host, state, mac_address, mac_vendor)
            )
            num_hosts += 1

        # Get scan statistics
        scan_stats = nm.scanstats()
        self.lbl_hosts_value.config(text=f"Hosts: {num_hosts}")
        self.lbl_elapsed_value.config(
            text=f"Elapsed: {scan_stats.get('elapsed')} seconds")
```

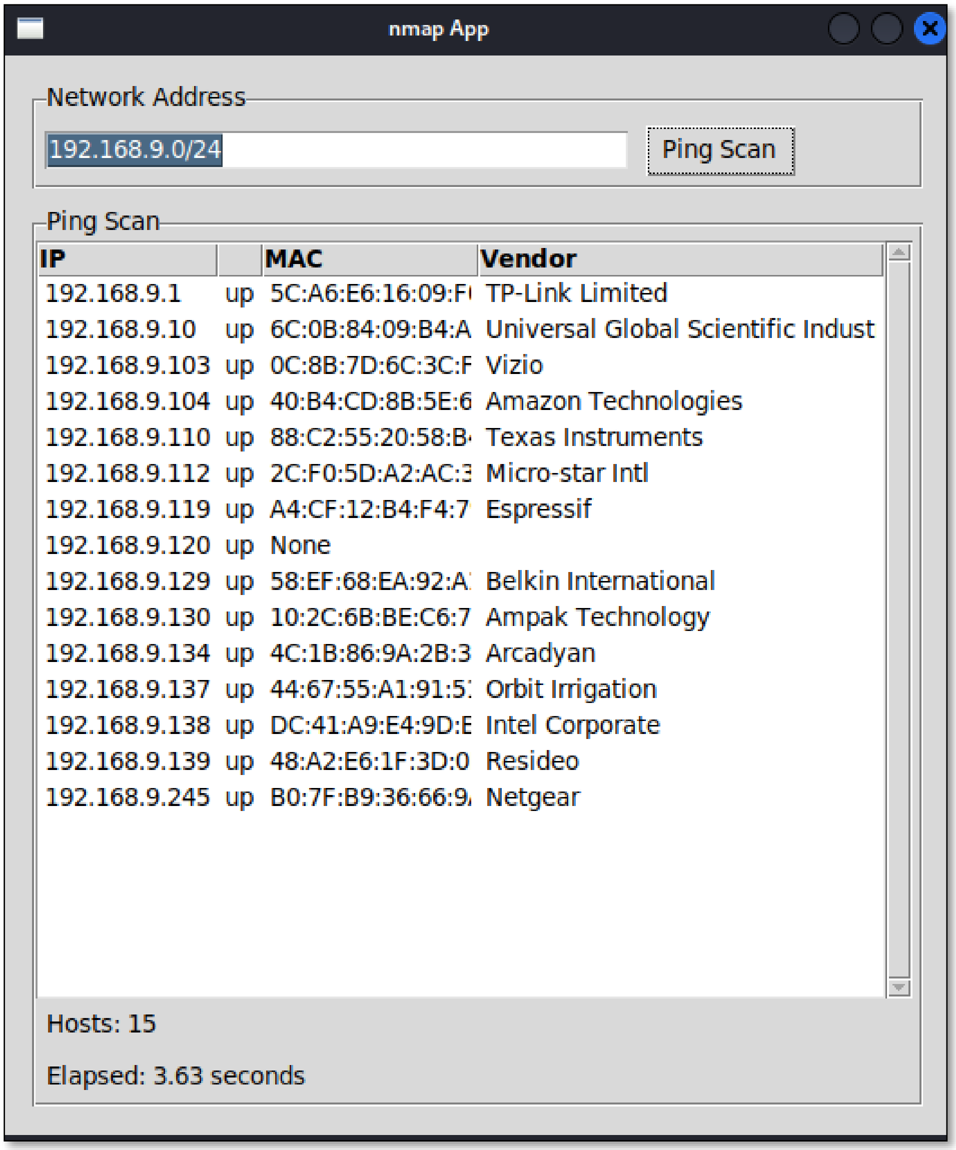


```
# ------------------------- CREATE WIDGETS ---------------------------------#
    def create_widgets(self):
        """Create and place GUI widgets"""
        # Create widgets
        # Create frames
        self.entry_frame = LabelFrame(self.window, text="Network Address")
        self.display_frame = LabelFrame(self.window, text="Ping Scan")

        # Fill the frame to the width of the window
        self.entry_frame.pack(fill=X)
        self.display_frame.pack(fill=X)
        # Keep the frame size regardless of the widget sizes
        self.entry_frame.pack_propagate(False)
        self.display_frame.pack_propagate(False)

        self.entry_network_address = Entry(self.entry_frame, width=40)
        # Set this to your default network address
        self.entry_network_address.insert(
            END, string="192.168.9.0/24")
        # Select all text in entry
        self.entry_network_address.selection_range(0, END)
        self.entry_network_address.focus_set()

        self.btn_scan = Button(
            self.entry_frame,
            text="Ping Scan",
            command=self.scan
        )

        self.lbl_hosts_value = Label(self.display_frame, text="Hosts:")
        self.lbl_elapsed_value = Label(self.display_frame, text="Elapsed:")
        self.lbl_hosts_value.grid(row=1, column=0, sticky=W)
        self.lbl_elapsed_value.grid(row=2, column=0, sticky=W)

        # Enter key will activate the scan method
        self.window.bind('<Return>', self.scan)
        self.window.bind('<KP_Enter>', self.scan)

        # Place Widgets
        self.entry_network_address.grid(
            row=1, column=1, columnspan=2, sticky=W)
        self.btn_scan.grid(row=1, column=3)

        self.entry_frame.pack_configure(padx=5, pady=5)
        self.display_frame.pack_configure(padx=5, pady=5)
        # Set padding for all widgets
        for child in self.entry_frame.winfo_children():
            child.grid_configure(padx=5, pady=5)
        for child in self.display_frame.winfo_children():
            child.grid_configure(padx=5, pady=5)
```



```
# -------------------------- TREEVIEW AND SCROLLBAR ------------------------#
    def create_treeview(self):
        """Setup tree view for display"""
        # Create treeview
        self.tree = Treeview(
            self.display_frame,
            height=20,
            columns=("ip", "state", "mac", "vendor"),
            style="Treeview",
            show="headings",
            selectmode="browse"
        )
        # Setup the columns
        self.tree.column("ip", width=100)
        self.tree.column("state", width=25)
        self.tree.column("mac", width=120)
        self.tree.column("vendor", width=225)

        # Setup the heading text visible at the top of the column
        self.tree.heading("ip", text="IP", anchor=W)
        self.tree.heading("state", text="", anchor=W)
        self.tree.heading("mac", text="MAC", anchor=W)
        self.tree.heading("vendor", text="Vendor", anchor=W)

        # Grid the tree
        self.tree.grid(row=0, column=0)

        # Create scrollbar for treeview
        self.scrollbar = Scrollbar(
            self.display_frame,
            orient="vertical",
            command=self.tree.yview
        )

        # Set scroll bar to scroll vertically and attach to the tree
        self.tree.configure(yscroll=self.scrollbar.set)

        # Grid scrollbar just to the right of the tree
        # sn (SouthNorth) expands scrollbar to height of tree
        self.scrollbar.grid(row=0, column=1, sticky="sn")


# Create program object
nmap_scanner = NmapScanner()
```

Example run using bridged adapter in Kali Linux:

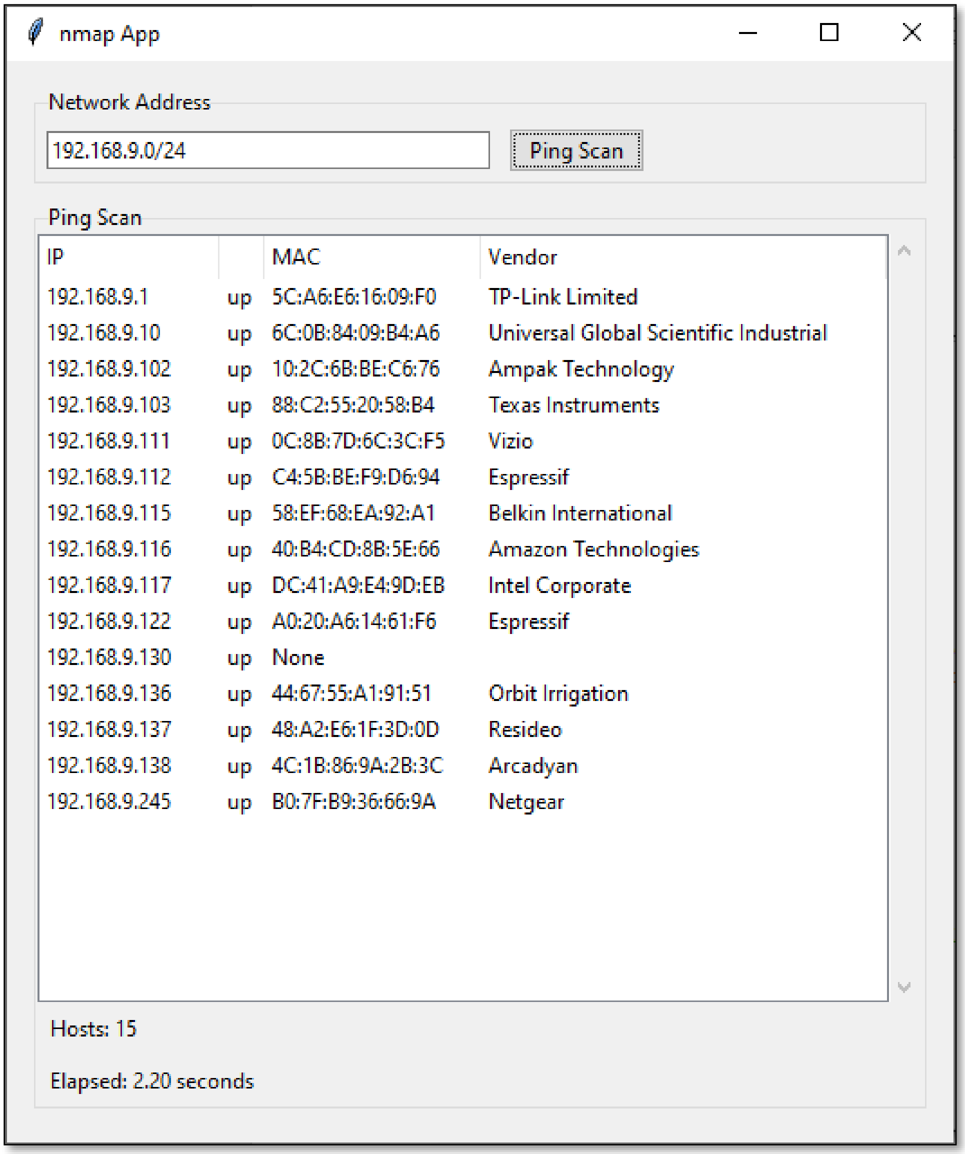


IP		MAC	Vendor
192.168.9.1	up	5C:A6:E6:16:09:F	TP-Link Limited
192.168.9.10	up	6C:0B:84:09:B4:A	Universal Global Scientific Indust
192.168.9.103	up	0C:8B:7D:6C:3C:F	Vizio
192.168.9.104	up	40:B4:CD:8B:5E:6	Amazon Technologies
192.168.9.110	up	88:C2:55:20:58:B	Texas Instruments
192.168.9.112	up	2C:F0:5D:A2:AC:3	Micro-star Intl
192.168.9.119	up	A4:CF:12:B4:F4:7	Espressif
192.168.9.120	up	None	
192.168.9.129	up	58:EF:68:EA:92:A	Belkin International
192.168.9.130	up	10:2C:6B:BE:C6:7	Ampak Technology
192.168.9.134	up	4C:1B:86:9A:2B:3	Arcadyan
192.168.9.137	up	44:67:55:A1:91:5	Orbit Irrigation
192.168.9.138	up	DC:41:A9:E4:9D:E	Intel Corporate
192.168.9.139	up	48:A2:E6:1F:3D:0	Resideo
192.168.9.245	up	B0:7F:B9:36:66:9	Netgear

Hosts: 15

Elapsed: 3.63 seconds

Example run using a local Windows computer:



Network Address

192.168.9.0/24

Ping Scan

IP		MAC	Vendor
192.168.9.1	up	5C:A6:E6:16:09:F0	TP-Link Limited
192.168.9.10	up	6C:0B:84:09:B4:A6	Universal Global Scientific Industrial
192.168.9.102	up	10:2C:6B:BE:C6:76	Ampak Technology
192.168.9.103	up	88:C2:55:20:58:B4	Texas Instruments
192.168.9.111	up	0C:8B:7D:6C:3C:F5	Vizio
192.168.9.112	up	C4:5B:BE:F9:D6:94	Espressif
192.168.9.115	up	58:EF:68:EA:92:A1	Belkin International
192.168.9.116	up	40:B4:CD:8B:5E:66	Amazon Technologies
192.168.9.117	up	DC:41:A9:E4:9D:EB	Intel Corporate
192.168.9.122	up	A0:20:A6:14:61:F6	Espressif
192.168.9.130	up	None	
192.168.9.136	up	44:67:55:A1:91:51	Orbit Irrigation
192.168.9.137	up	48:A2:E6:1F:3D:0D	Resideo
192.168.9.138	up	4C:1B:86:9A:2B:3C	Arcadyan
192.168.9.245	up	B0:7F:B9:36:66:9A	Netgear

Hosts: 15

Elapsed: 2.20 seconds

Assignment Submission

1. Attach the code.
2. Attach a screenshot showing a successful run of the program.
3. Submit the assignment in Blackboard.