

# Python Nmap Port Scanner

## Contents

|   |    |
|---|----|
| Python Nmap Port Scanner .....                        | 1  |
| What is Nmap? .....                                   | 1  |
| Tutorial 1: Install Nmap .....                        | 2  |
| python-nmap.....                                      | 2  |
| Tutorial 2: Nmap with Python .....                    | 2  |
| Run the Scan .....                                    | 5  |
| Extra Credit Tutorial 3: Nmap in Python with GUI..... | 5  |
| Challenge.....  | 11 |
| Assignment Submission.....                            | 11 |

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 scripts 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 in Python, let's install the **python-nmap** module:

```
# Windows: pip install python-nmap
# Linux: sudo apt install python3-nmap
```

## Tutorial 2: Nmap with Python

Create a Python file named **nmap\_ping\_scan\_cli.py**

```

1  #!/usr/bin/env python3
2  """
3      Name: nmap_network_cli.py
4      Author:
5      Created:
6      Purpose: Use python-nmap wrapper library to use nmap to scan network
7  """
8  # Windows: pip install python-nmap
9  # Linux: sudo apt install python3-nmap
10 import nmap
11
12
13 def main():
14     print(" +-----+ ")
15     print(" |----- Python nmap Network Scanner -----| ")
16     print(" +-----+ ")
17
18     # Change to default value of your network
19     local_network = "192.168.9.0/24"
20
21     # Get network address, if blank use local_network variable
22     network = input(
23         " Enter network address (192.168.1.0/24): ") or local_network
24
25     print(" Scanning the network . . . ")
26
27     # Call scan function with network address parameter
28     scan(network)

```

```

30 # ----- SCAN -----#
    Codiumate: Options | Test this function
31 def scan(network):
32     # Initialize create Nmap port scanner object
33     nm = nmap.PortScanner()
34
35     # Set target and arguments
36     # You can get common settings from Zenmap
37     # -sn - Ping scan
38     nm.scan(
39         hosts=network,
40         arguments="-sn"
41     )
42
43     num_hosts = 0
44     for x in nm.all_hosts():
45         # Get scan information from nmap dictionary
46         host = nm[x].get("addresses").get("ipv4")
47         state = nm[x].get("status").get("state")
48         mac_address = nm[x].get("addresses").get("mac")
49         # If the device does not have a MAC address
50         # it will not have a MAC vendor
51         try:
52             mac_vendor = list(nm[x].get('vendor').values())[0]
53         except:
54             mac_vendor = ""
55         # Display the host IP and status
56         print(f" {host} \t{state} {mac_address} ({mac_vendor})")
57         num_hosts += 1
58
59     print(f" Number of hosts: {num_hosts}")
60     # Get and display scan statistics
61     scan_stats = nm.scanstats()
62     print(f" Elapsed: {scan_stats.get('elapsed')} seconds\n")
63
64
65 if __name__ == '__main__':
66     main()

```

Example run:

---

## 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 . . .
192.168.9.101 up 5C:A6:E6:16:09:F0 (TP-Link Limited)
192.168.9.102 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
```

## 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**

```

1  #!/usr/bin/env python3
2  """
3      Name: nmap_ping_scan_gui.py
4      Author:
5      Created:
6      Purpose: Purpose: Use Python nmap wrapper to scan network
7  """
8
9  from tkinter import *
10 from tkinter.ttk import *
11 # Windows: pip install python-nmap
12 # Linux: sudo apt install python3-nmap
13 import nmap
14
15
16 class NmapScanner:
17
18     def __init__(self):
19         """ Initialize program """
20         self.window = Tk()
21         self.window.title("nmap App")
22         self.window.geometry("525x600")
23         self.window.config(padx=10, pady=10)
24
25         self.create_widgets()
26         self.create_treeview()
27         mainloop()

```

```

30 # ----- SCAN ----- #
31 def scan(self, *args):
32     # Return a list of tuples from treeview
33     items = self.tree.get_children()
34
35     # Iterate through list to delete all items in the treeview
36     for item in items:
37         self.tree.delete(item)
38
39     # Initialize create Nmap port scanner object
40     nm = nmap.PortScanner()
41     self.network = self.entry_network_address.get()
42     # Set target and arguments
43     # You can get common settings from Zenmap
44     # -sn - Ping scan
45     nm.scan(
46         hosts=self.network,
47         arguments="-sn"
48     )
49
50     num_hosts = 0
51     for hosts in nm.all_hosts():
52         # Get scan information from nmap dictionary
53         host = nm[hosts].get("addresses").get("ipv4")
54         state = nm[hosts].get("status").get("state")
55         mac_address = nm[hosts].get("addresses").get("mac")
56         # If the device does not have a MAC address
57         # it will not have a MAC vendor.
58         try:
59             mac_vendor = list(nm[hosts].get('vendor').values())[0]
60         except:
61             # If there isn't a MAC address or the MAC address lookup
62             # fails, leave it empty
63             mac_vendor = ""
64
65         self.tree.insert("", "end", text=num_hosts, values=(
66             host, state, mac_address, mac_vendor)
67         )
68         num_hosts += 1
69
70     # Get scan statistics
71     scan_stats = nm.scanstats()
72     self.lbl_hosts_value.config(text=f"Hosts: {num_hosts}")
73     self.lbl_elapsed_value.config(
74         text=f"Elapsed: {scan_stats.get('elapsed')} seconds")
75

```

```

76 # ----- CREATE WIDGETS -----#
77 def create_widgets(self):
78     """Create and place GUI widgets"""
79     # Create widgets
80     # Create frames
81     self.entry_frame = LabelFrame(self.window, text="Network Address")
82     self.display_frame = LabelFrame(self.window, text="Ping Scan")
83
84     # Fill the frame to the width of the window
85     self.entry_frame.pack(fill=X)
86     self.display_frame.pack(fill=X)
87     # Keep the frame size regardless of the widget sizes
88     self.entry_frame.pack_propagate(False)
89     self.display_frame.pack_propagate(False)
90
91     self.entry_network_address = Entry(self.entry_frame, width=40)
92     # Set this to your default network address
93     self.entry_network_address.insert(
94         END, string="192.168.9.0/24")
95     # Select all text in entry
96     self.entry_network_address.selection_range(0, END)
97     self.entry_network_address.focus_set()
98
99     self.btn_scan = Button(
100         self.entry_frame,
101         text="Ping Scan",
102         command=self.scan
103     )
104
105     self.lbl_hosts_value = Label(self.display_frame, text="Hosts:")
106     self.lbl_elapsed_value = Label(self.display_frame, text="Elapsed:")
107     self.lbl_hosts_value.grid(row=1, column=0, sticky=W)
108     self.lbl_elapsed_value.grid(row=2, column=0, sticky=W)
109
110     # Enter key will activate the scan method
111     self.window.bind('<Return>', self.scan)
112     self.window.bind('<KP_Enter>', self.scan)
113
114     # Place Widgets
115     self.entry_network_address.grid(
116         row=1, column=1, columnspan=2, sticky=W)
117     self.btn_scan.grid(row=1, column=3)
118
119     self.entry_frame.pack_configure(padx=5, pady=5)
120     self.display_frame.pack_configure(padx=5, pady=5)
121     # Set padding for all widgets
122     for child in self.entry_frame.winfo_children():
123         child.grid_configure(padx=5, pady=5)
124     for child in self.display_frame.winfo_children():
125         child.grid_configure(padx=5, pady=5)

```

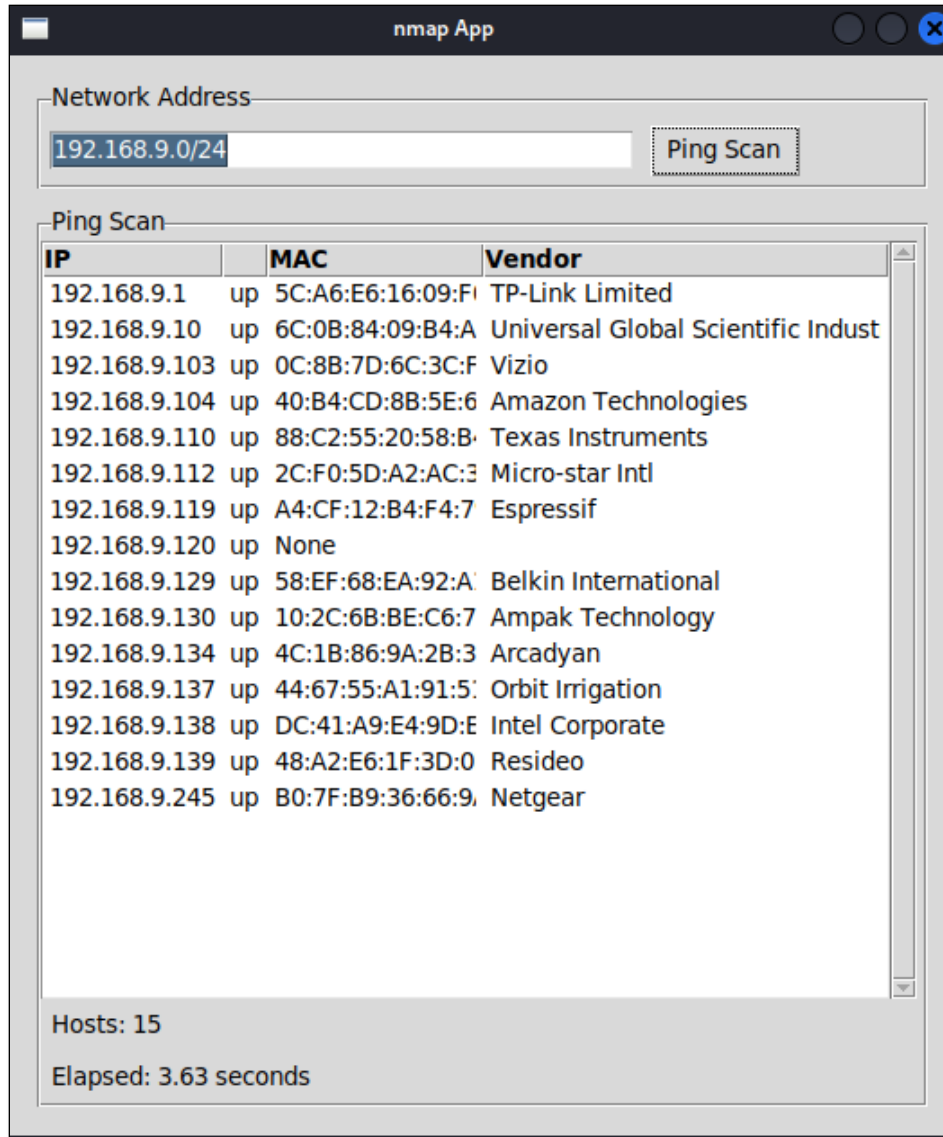


```

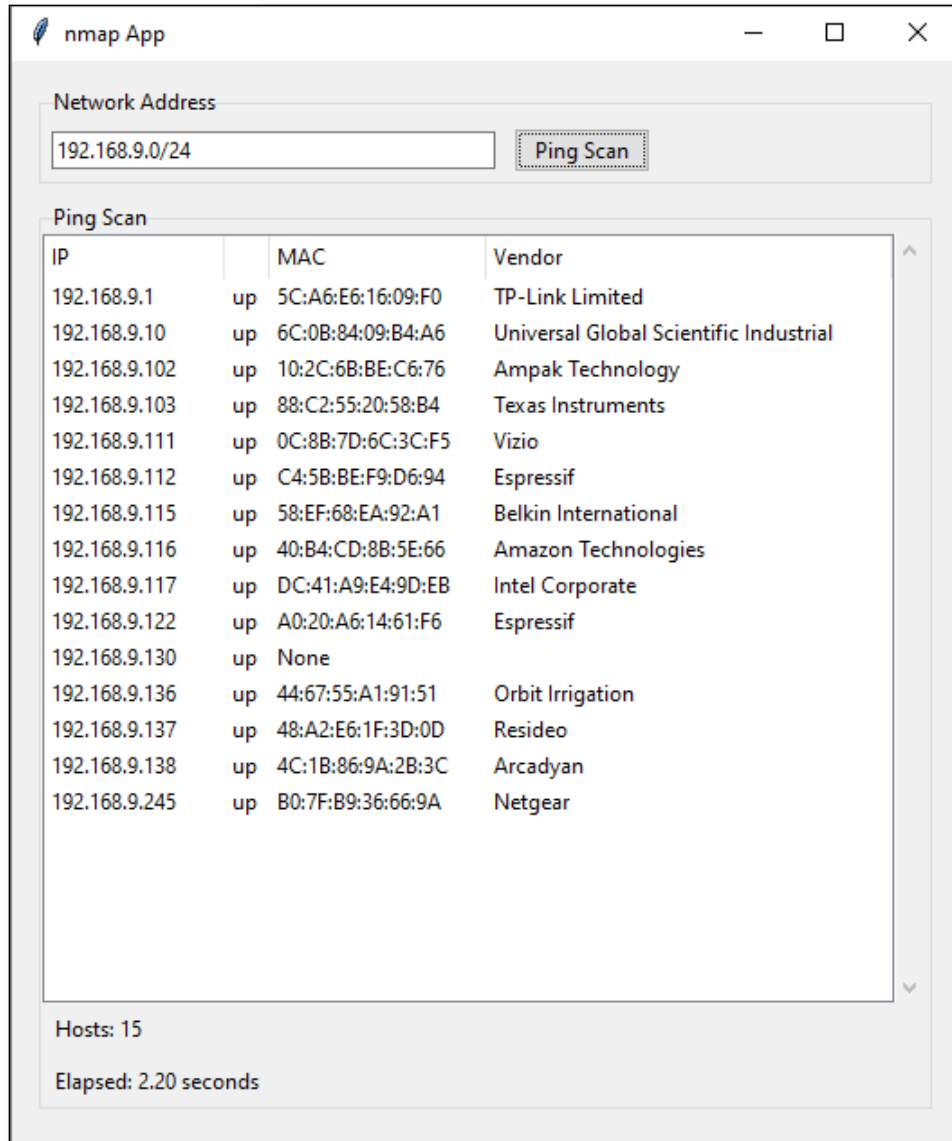
127 # ----- TREEVIEW AND SCROLLBAR ----- #
128 def create_treeview(self):
129     """Setup tree view for display"""
130     # Create treeview
131     self.tree = Treeview(
132         self.display_frame,
133         height=20,
134         columns=("ip", "state", "mac", "vendor"),
135         style="Treeview",
136         show="headings",
137         selectmode="browse"
138     )
139     # Setup the columns
140     self.tree.column("ip", width=100)
141     self.tree.column("state", width=25)
142     self.tree.column("mac", width=120)
143     self.tree.column("vendor", width=225)
144
145     # Setup the heading text visible at the top of the column
146     self.tree.heading("ip", text="IP", anchor=W)
147     self.tree.heading("state", text="", anchor=W)
148     self.tree.heading("mac", text="MAC", anchor=W)
149     self.tree.heading("vendor", text="Vendor", anchor=W)
150
151     # Grid the tree
152     self.tree.grid(row=0, column=0)
153
154     # Create scrollbar for treeview
155     self.scrollbar = Scrollbar(
156         self.display_frame,
157         orient="vertical",
158         command=self.tree.yview
159     )
160
161     # Set scroll bar to scroll vertically and attach to the tree
162     self.tree.configure(yscroll=self.scrollbar.set)
163
164     # Grid scrollbar just to the right of the tree
165     # sn (SouthNorth) expands scrollbar to height of tree
166     self.scrollbar.grid(row=0, column=1, sticky="sn")
167
168
169 # Create program object
170 nmap_scanner = NmapScanner()

```

Example run using bridged adapter in Kali Linux:



Example run using a local Windows computer:



## Challenge

- Add a drop down list for other nmap options than ping scan.

---

## Assignment Submission

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