Python Port Scanner Tutorial

Contents

Time required: 60 minutes

Python Tabs and Spaces Issue

Visual Studio Code automatically changes a tab into four spaces. Other editors, like geany and nano in Linux, do not. You can end up with a combination of spaces and tabs. Python doesn't like a combination, it wants either one or the other. The preferred method is spaces.

Recommendation:

1. Create your Python files in Visual Studio Code in Windows.
2. Copy and paste the code into either nano or geany in Linux.

Run on both Linux and Windows.

Objective: Write a cross platform Python script that scans ports on a specified IP address.

What is Port Scanning?

Port scanning may be defined as a surveillance technique, which is used in to locate the open ports available on a particular host. Network administrator, penetration tester or a hacker can use this technique. We can configure the port scanner according to our requirements to get maximum information from the target system.

Port scanning is just like a thief who wants to enter a house by checking every door and window to see which ones are open. As discussed earlier, TCP/IP protocol suite, use for communication over internet, is made up of two protocols namely TCP and UDP. Both of the protocols have 0 to 65535 ports. As it always advisable to close unnecessary ports of our system hence essentially, there are more than 65000 doors (ports) to lock. These 65535 ports can be divided into the following three ranges −

- **System or well-known ports:** 0 to 1023
- **User or registered ports:** 1024 to 49151
- **Dynamic or private ports:** > 49151

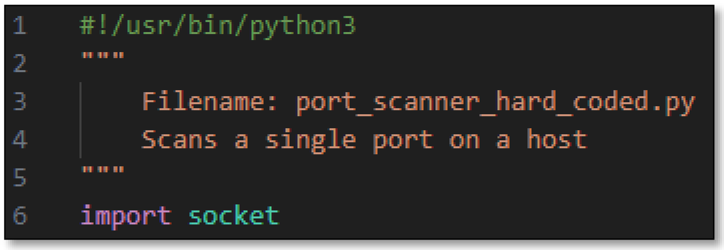
In this tutorial, you will be able to make your own port scanner in Python using socket library. The basic idea behind this simple port scanner is to try to connect to a specific host (website, server, or any device connected to the Internet/network) through a list of ports, if a successful connection has been established, that means the port is open.

When you loaded a web page, you have made a connection to the website on port 443. This script will try to connect to a host on multiple ports. These kinds of tools are useful for hackers and penetration testers.

WARNING: Never use a tool like this on a host that you don't have permission to test!

Tutorial 1: Port Scanner Hard Coded

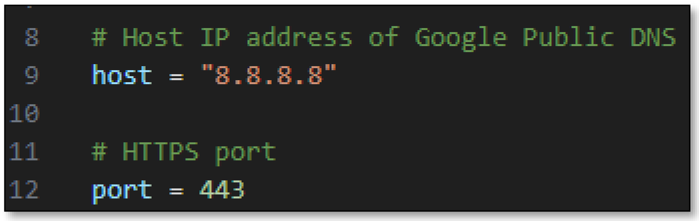
In this section, we will write a hard coded port scanner that scans a single host and a single port.



```python
#!/usr/bin/python3
"""
    Filename: port_scanner_hard_coded.py
    Scans a single port on a host
"""
import socket
```

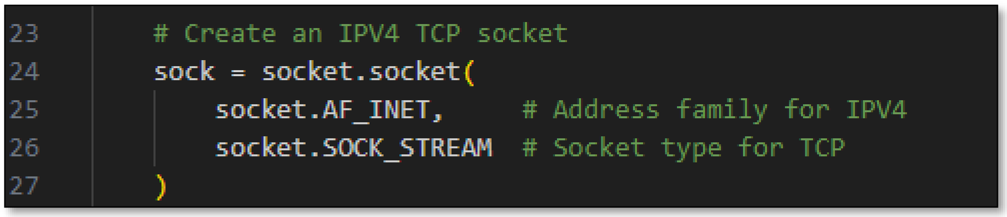
Let’s start by importing socket module, which is are part of the Python standard library.

The **socket** module provides us with socket operations, functions for network-related tasks, etc. They are widely used on the Internet, as they are behind any connection to any network. All network communication goes through a socket.



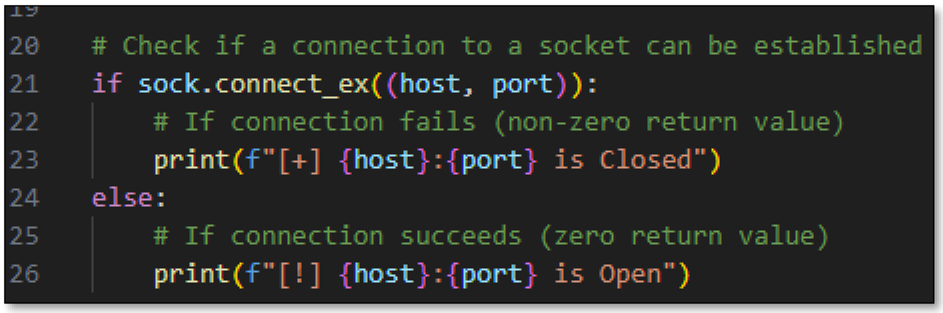
```python
# Host IP address of Google Public DNS
host = "8.8.8.8"

# HTTPS port
port = 443
```

Set the host and port of the server to test.



```python
    # Create an IPV4 TCP socket
    sock = socket.socket(
        socket.AF_INET,     # Address family for IPV4
        socket.SOCK_STREAM  # Socket type for TCP
    )
```

We create a socket object named sock to connect to an IPV4 IP address with TCP.



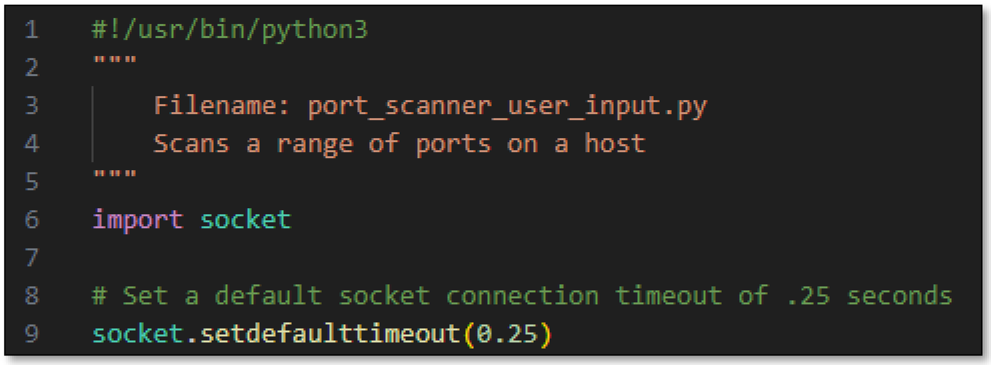
```python
# Check if a connection to a socket can be established
if sock.connect_ex((host, port)):
    # If connection fails (non-zero return value)
    print(f"[+] {host}:{port} is Closed")
else:
    # If connection succeeds (zero return value)
    print(f"[!] {host}:{port} is Open")
```

If the socket object doesn't make a connection, we print closed. The target port is closed.

If the connection is successful, it means the port is open.

Tutorial 2: Python Port Scanner User Input

This Python script **port_scanner_user_input.py** attempts to determine if each port between 0 and 1023 on a specified host is open or closed.

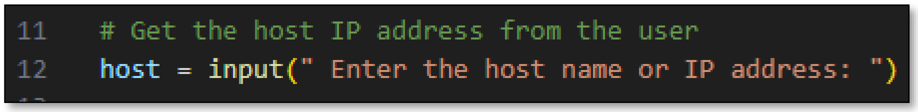


```python
#!/usr/bin/python3
"""
    Filename: port_scanner_user_input.py
    Scans a range of ports on a host
"""
import socket

# Set a default socket connection timeout of .25 seconds
socket.setdefaulttimeout(0.25)
```

Let’s start by importing socket module, which is are part of the Python standard library.

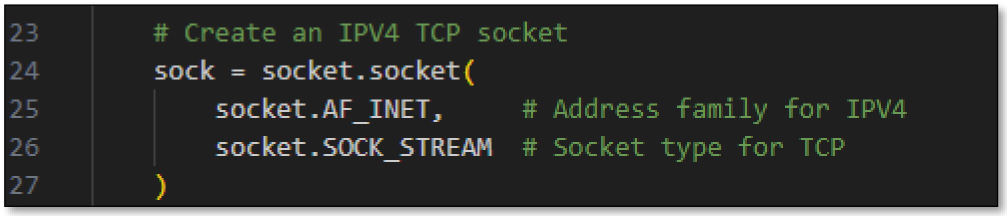
The **socket** module provides us with socket operations, functions for network-related tasks, etc. They are widely used on the Internet, as they are behind any connection to any network. All network communication goes through a socket.

We set a socket connection timeout of 250 milliseconds using **settimeout(0.25)** method. This is the maximum time the script will wait for a connection attempt before considering it unsuccessful.. We are scanning one port at a time, if we don’t set a time out, we could be scanning for a very long time.



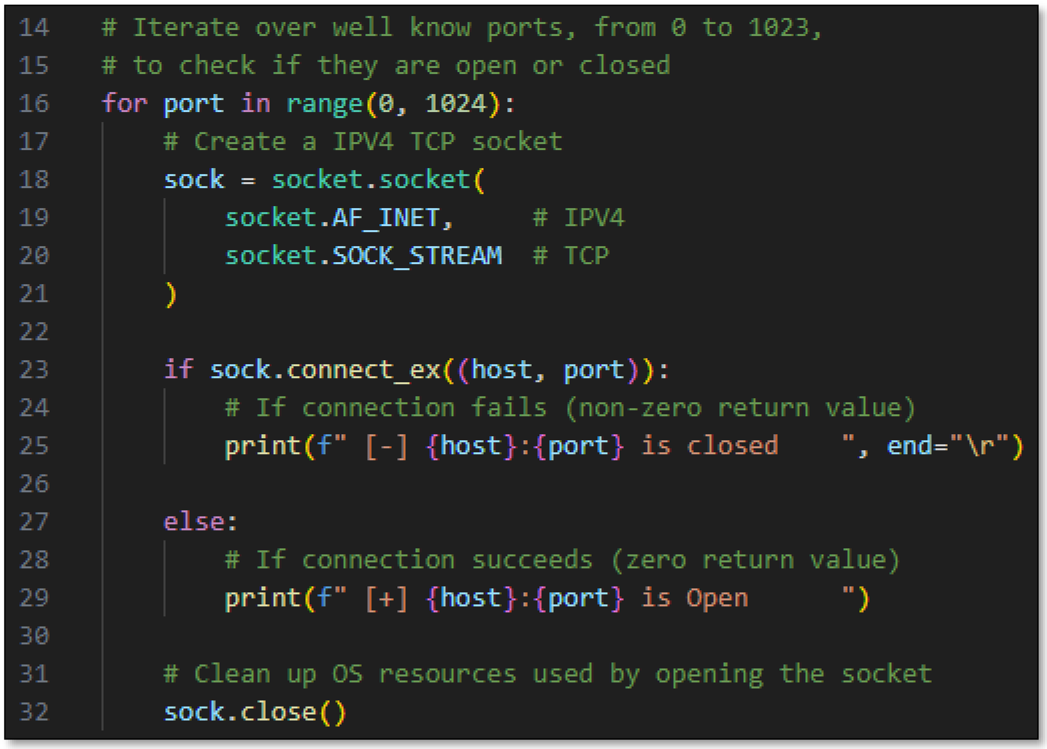
```python
# Get the host IP address from the user
host = input(" Enter the host name or IP address: ")
```

Get the host name or IP address from the user.



```python
    # Create an IPV4 TCP socket
    sock = socket.socket(
        socket.AF_INET,     # Address family for IPV4
        socket.SOCK_STREAM  # Socket type for TCP
    )
```

We create a socket object named sock to connect to an IPV4 IP address with TCP.



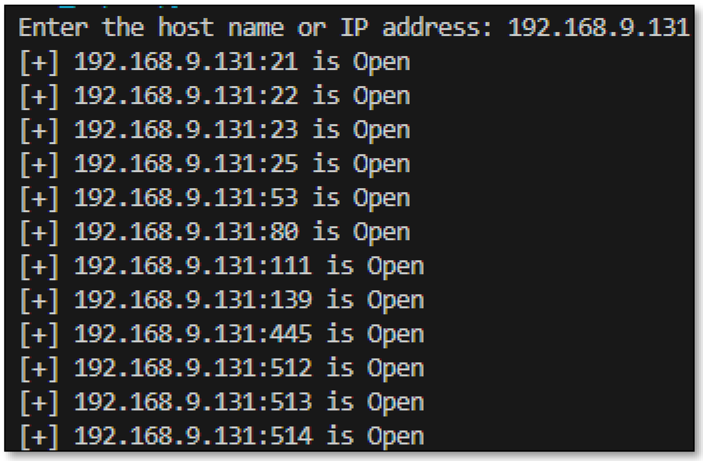
```python
# Iterate over well know ports, from 0 to 1023,
# to check if they are open or closed
for port in range(0, 1024):
    # Create a IPV4 TCP socket
    sock = socket.socket(
        socket.AF_INET,     # IPV4
        socket.SOCK_STREAM  # TCP
    )

    if sock.connect_ex((host, port)):
        # If connection fails (non-zero return value)
        print(f" [-] {host}:{port} is closed     ", end="\r")

    else:
        # If connection succeeds (zero return value)
        print(f" [+] {host}:{port} is Open      ")

    # Clean up OS resources used by opening the socket
    sock.close()
```

If the socket object doesn't make a connection, we return false. The target port is closed. We close the socket object to release the socket operating system resource.

Run the scan.

1. Start and logon to your metasploitable2 VM.
2. Use **ip a** to find out the ip address.
3. Scan the metasploitable ip address from your Kali Linux VM. Make sure each VM is on the same network.

Example run:



```
Enter the host name or IP address: 192.168.9.131
[+] 192.168.9.131:21 is Open
[+] 192.168.9.131:22 is Open
[+] 192.168.9.131:23 is Open
[+] 192.168.9.131:25 is Open
[+] 192.168.9.131:53 is Open
[+] 192.168.9.131:80 is Open
[+] 192.168.9.131:111 is Open
[+] 192.168.9.131:139 is Open
[+] 192.168.9.131:445 is Open
[+] 192.168.9.131:512 is Open
[+] 192.168.9.131:513 is Open
[+] 192.168.9.131:514 is Open
```

You can see from this example there are several ports open.

Conclusion

Port scanning proves to be useful in many cases, an authorized penetration tester can use this tool to see which ports are open and revealing the presence of potential security devices such as firewalls, as well as testing the network security and the strength of a device.

It is also a popular reconnaissance tool for hackers that are seeking weak points to gain access to the target machine.

Most penetration testers use **nmap** to scan ports, as it does not just provide port scanning, but shows services and operating systems that are running, and much more advanced techniques.

Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.