

Forum:[Hack Forums](#)

Thread: [Python Port Scanner - How to make it](#)

??????????? - August 29, 2025, 5:13 am

Tutorial: Writing a Python Port Scanner (Step by Step for Beginners)

Introduction

A port scanner is one of the most important tools in a hacker's or penetration tester's toolbox. Before you exploit a system, you first need to know what services it's running. Services = ports, and ports = attack surface.

While tools like Nmap are industry standards, writing your own port scanner in Python is one of the best beginner projects in cybersecurity. It teaches you: How networking works under the hood.

How sockets and TCP connections are established.

How to use Python for offensive security scripting.

In this tutorial, we'll write a working port scanner in Python step by step, test it, and then discuss improvements you can add to make it more powerful.

What is a Port Scanner?

Every service on a computer (web, FTP, SSH, DNS, etc.) communicates through a port. Some common ports include: Port 21 -> FTP

Port 22 -> SSH

Port 25 -> SMTP

Port 80 -> HTTP

Port 443 -> HTTPS

Port 3306 -> MySQL

A port scanner attempts connections to a range of ports and reports which ones are open.

Why this matters: Open ports = potential entry points.

It reveals what technologies the target is running.

It's the foundation of every penetration test.

Step 1: Import the Required Library

Python includes the Code socket library for handling network connections. We'll use it to connect to different ports.

```
Code import socket
```

This is all we need for a basic TCP scanner.

Step 2: Define the Target

We need to specify the IP address or hostname of the machine we want to scan. For safety, always test on your own machine or lab.

```
Code target = "127.0.0.1" # localhost for testing
```

Tip: Replace this with the IP of a VM or test environment you control. Scanning unauthorized systems is illegal.

Step 3: Create a Function to Scan a Single Port

We'll write a function that tries to connect to one port and reports if it's open.

```
Code def scan_port(port):
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1) # Prevents hanging
        s.connect((target, port))
        print(f"[+] Port {port} is OPEN")
        s.close()
    except (socket.timeout, ConnectionRefusedError):
        pass
    except Exception as e:
        print(f"Error on port {port}: {e}")
```

How it works: Code socket.AF_INET -> IPv4.

Code socket.SOCK_STREAM -> TCP connection.

Code settimeout(1) -> limits wait time to 1 second.

If it connects successfully -> port is open.

If it fails -> port is closed (we ignore it).

Step 4: Loop Through Multiple Ports

Instead of checking just one port, let's scan a whole range.

```
Code for port in range(1, 1025): # Scan ports 1–1024
    scan_port(port)
```

Ports 1–1024 are the “well-known ports” used by common services.

Step 5: Full Script (dummy/testing)

Here's a complete, working Python port scanner.

Code import socket

```
target = "127.0.0.1" # Replace with a legal target you control
```

```
def scan_port(port):
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1)
        s.connect((target, port))
        print(f"[+] Port {port} is OPEN")
        s.close()
    except (socket.timeout, ConnectionRefusedError):
        pass
    except Exception as e:
        print(f"Error on port {port}: {e}")
```

```
print(f"[*] Scanning target: {target}")
```

```
for port in range(1, 1025):
    scan_port(port)
```

Run it with:

Code python3 scanner.py

You'll see open ports printed in the terminal.

Step 6: Improvements

Our scanner works, but we can make it better:

Threading: Use Code threading or Code concurrent.futures to scan ports faster.

Banner Grabbing: Try reading service banners to identify software versions.

Custom Input: Ask the user for a target and port range instead of hardcoding.

Logging: Save results to a file.

UDP Scanning: Extend functionality to scan UDP ports.

Example: Banner Grabbing

Code def scan_port(port):

```
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)
    s.connect((target, port))
    try:
        banner = s.recv(1024).decode().strip()
        print(f"[+] Port {port} OPEN | Banner: {banner}")
    except:
        print(f"[+] Port {port} OPEN (no banner)")
    s.close()
except:
    pass
```

This can reveal information like “Apache/2.4.49” or “OpenSSH 8.2” which can be checked for vulnerabilities.

Real-World Uses Pentesters use port scans during reconnaissance.

Admins use them to find misconfigured or unnecessary services.

Learners use them to understand network behavior.

Conclusion

By building your own Python port scanner, you gain more than just a tool, you gain an understanding of how network services respond, how connections are made, and why scanning is so fundamental to hacking.

Tools like Nmap will always be superior in features, but writing your own scanner builds the foundation for every other security project you'll do in the future.

ttbr0 - August 30, 2025, 1:55 am

i truly believe that to be successful as a hacker you should first be learning to build something. when you have built something, you gain understanding of how something works. when you gain understanding of a system, you might realize vulnerabilities in it. when you realize the vulnerabilities, you know how to attack it. when you know how to attack it, you know can figure out how to defend it. this is the crucial first step that people always skip and it is detrimental to their progress. its why i always recommend learning programming before hacking. if you cant even program, i doubt you have any meaningful knowledge of how a computer works at the fundamental level. if you dont even have this simple knowledge, i wouldnt trust you could exploit a system even if a blatant vulnerability was staring at you right in your face.

??????????? - August 30, 2025, 2:22 am

(Aug 30, 2025 01:55 AM)ttbr0 Wrote: [CITE](#) i truly believe that to be successful as a hacker you should first be learning to build something. when you have built something, you gain understanding of how something works. when you gain understanding of a system, you might realize vulnerabilities in it. when you realize the vulnerabilities, you know how to attack it. when you know how to attack it, you know can figure out how to defend it. this is the crucial first step that people always skip and it is detrimental to their progress. its why i always recommend learning programming before hacking. if you cant even program, i doubt you have any meaningful knowledge of how a computer works at the fundamental level. if you dont even have this simple knowledge, i wouldnt trust you could exploit a system even if a blatant vulnerability was staring at you right in your face.

Gem of an advice.

Stanley - September 12, 2026, 6:21 am

I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley