**Basic Port Scanner Using Python Sockets**

**Abstract:**  
This project implements a basic network port scanner using Python’s socket library. The scanner checks if common ports (SSH, HTTP, HTTPS etc) are open on a target machine.

**Introduction:**  
Port scanning is used to discover open ports on a system.  
It can be useful for cybersecurity assessments and network troubleshooting.  
Only scan systems that you own or have permission to scan.

**Methodology / Code Explanation:**

A screenshot of a computer program

AI-generated content may be incorrect.

Import socket *(This line imports Python’s built-in socket library.)*  
target\_ip = “192.0.2.1” *(This line creates a variable named target\_ip and stores the IP address of the computer you want to scan. This address can be changed.)*

ports\_to\_check = [22, 80, 443, 8080] *(This line creates a list of common port numbers to be checked for openness)*

for port in ports\_to\_check: *(This line starts a loop. The program will repeat the code indented for all numbers in the ports\_to\_check list)*

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) *(This line creates a new socket objuct named sock. This socket tries to connect to a specific port. AF\_INET specifies that youre using IPv4. Sock\_Stream means youre creating a TCP connection.)*

sock.settimeout(1) (This line sets a timeout of 1 second)

result = sock.connect\_ex((target\_ip, port)) *(This is the core of the scanner. The connect\_ex method tries to make a connection to the target\_ip and the current port number. It doesn't throw an error if the connection fails; instead, it returns a number that tells you what happened. A 0 means the connection was successful.)*

if result == 0:  
 print(f"Port {port} is OPEN!")  
 else:  
 print(f"Port {port} is CLOSED.") *(This is an if else statement that checks the result from the previous line.)*

sock.close() *(This line closes the socket after the check is finished.)*

print("Scan complete.") (*Once the loop has gone through all the ports listed, this line prints the final message to let you know the scan is complete.)*

**Discussion:**  
What was learned about socket programming?

The code demonstrates the fundamental principles of socket programming, which is the basis for all network communication. I learned that a socket is an endpoint for sending or receiving data across a network.   
 The code uses Python’s library to:  
 Establish a TCP connection  
 Handle timeouts  
 Check for open ports

How can this scanner be improved?

Multi-threading – the biggest limitation of the current code is that it scans ports sequentially. When scanning a large amount of ports, this can be really slow. Multi-threading can be used to check multiple ports at the same time.  
Scanning a range – The code could be modified to take a range of ports (e.g., from 1 to 1024) and loop through each one providing a more comprehensive result.

Limitations:   
The scanner is very slow because it checks each port one at a time. This becomes an issue when scanning a large range of ports.  
It only checks for TCP ports. Many services use UDP which is a different type of network communication.  
The scanner is limited to checking one IP address at a time and can not scan an entire network or range of IP addresses.

**Conclusion**  
This project showed how a simple Python script can be used to scan and identify open or closed ports on a target system. Although it is limited compared to professional tools, building this scanner helped me understand socket programming, TCP connections, and the role of ports in network security. The experience highlighted both the usefulness of automation for small network tasks and the importance of securing open ports to reduce vulnerabilities.