Port Scanner

Rationale

The idea of building my own port scanner stemmed from my curiosity about how network security tools like Nmap worked. While using Nmap in Kali Linux, I often wondered what went on behind the scenes. How did it scan ports?, interact with network protocols?, and identify open or closed ports on a system? Additionally, I wanted to deepen my understanding of concepts like TCP/UDP connections, common protocols, and IPv4 addressing, which I had explored in Hack The Box's Introduction to Networking module.

New Things I Learned

While building my port scanner, I learnt a lot about sockets. Sockets enable communication between devices over the internet or a local network. I learned how to use Python’s socket module to create and manage connections effectively. For instance, I discovered that `socket.AF_INET` is used to specify IPv4 addresses, while `socket.SOCK_STREAM` indicates a TCP connection.

I also explored DNS resolution. Using `socket.gethostbyname`, I learned how to resolve domain names (like google.com) into their corresponding IP addresses. Conversely, `socket.gethostbyaddr` provided the ability to perform a reverse lookup. It revealed the hostname associated with a given IP address.

Also, by using the `settimeout` function, I was able to control how long the scanner would wait before moving on, ensuring the program didn’t stall.

Building the Port Scanner

The port scanner uses two key functions. The first function, `con_scan`, focuses on scanning individual ports, while the second, `port_sca`n, manages the overall scanning operation and handles multiple ports.

The `con_scan` function is responsible for determining whether a specific port on a target host is open or closed. Using Python's socket module, it establishes a connection to the target host and port. If the connection is successful, the function outputs that the port is open. Otherwise, it indicates that the port is closed.

The second function, `port_scan`, starts by resolving the target host's domain name to its corresponding IP address. If successful, it attempts to identify the hostname via reverse DNS lookup. For each port provided in the input list, the function calls `con_scan` to determine its status, iterating through the list and reporting the findings.

I set the target as microsoft.com and specified ports 80 (HTTP) and 22 (SSH) for the test. Port 80 was identified as open, while port 22 was closed.