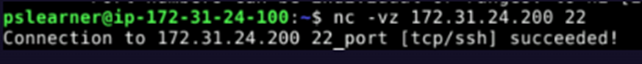
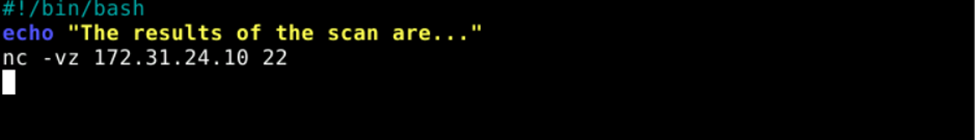
# Build a Simple Port Scanner with Bash

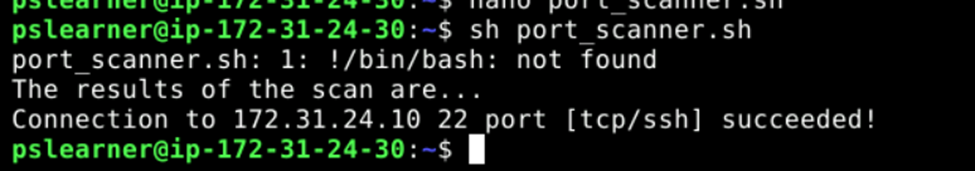
Port scanning is a critical aspect of network security, as it allows administrators and security professionals to identify open ports on a system, detect vulnerabilities, and assess potential attack vectors. This project demonstrates the process of creating a Bash script to automate port scanning using the Netcat utility. By first ensuring connectivity between the Ubuntu desktop and a target machine with the IP address 172.31.24.10 on port 22, the foundational functionality of Netcat is verified. The importance of this step lies in validating that the tools and network configuration are correctly set up before automating the process. Automating port scanning with a Bash script simplifies repetitive tasks, ensures consistent execution, and provides flexibility for scanning different targets and ports. This project not only highlights the practical application of Bash scripting for security tasks but also emphasizes the role of automation in enhancing the efficiency and reliability of security assessments. By systematically building and refining the script, we explore how to make the tool dynamic, user-friendly, and robust enough to handle diverse scenarios.



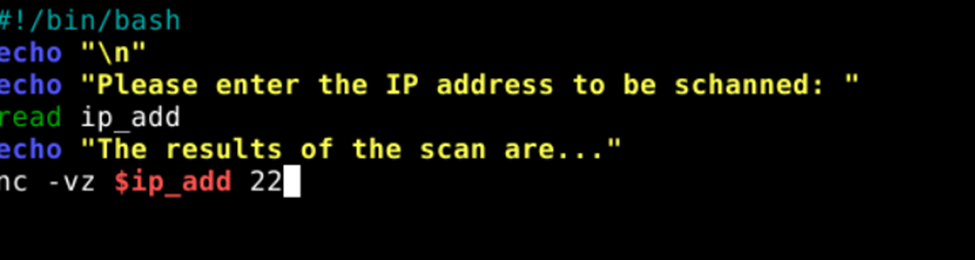
The goal of this challenge is to automate the Netcat command using a Bash script, creating a streamlined and repeatable process for port scanning. To begin, we create a new Bash script file named port\_scanner.sh in the current directory. The script starts with the shebang #!/bin/bash, indicating that the Bash shell will interpret the commands within the script. This foundational line ensures compatibility with the Bash scripting environment. The next step introduces user feedback by adding a line to output "The results of the scan are..." when the script is executed, providing clarity about the script's purpose and functionality. Finally, the Netcat command nc -vz 172.31.24.10 22 is included, replicating the manual test performed earlier. This initial script serves as a starting point, demonstrating the transition from manual command execution to an automated, reusable solution for basic port scanning tasks.



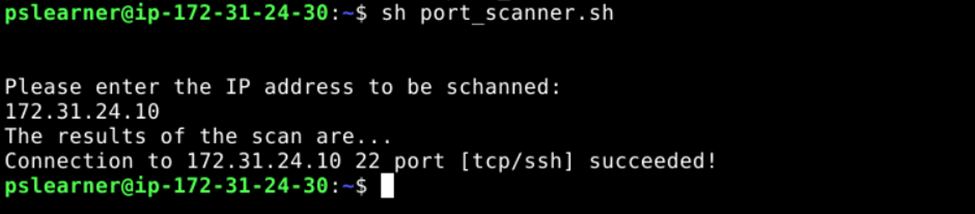
To execute the port scanner, run the script by typing sh port\_scanner.sh in the terminal. This will display the scan results along with the custom text output defined in the script, providing immediate feedback on its functionality.To execute the port scanner, run the script by typing sh port\_scanner.sh in the terminal. This will display the scan results along with the custom text output defined in the script, providing immediate feedback on its functionality.



To enhance the flexibility of the script, the next step involves modifying the existing port\_scanner.sh to create a more dynamic and user-friendly interface. Unlike the initial version, which relied on hard coded values, this iteration allows the user to input the IP address to be scanned, making the script adaptable for different targets. Begin by editing the script to remove the hard coded Netcat command and its associated output lines, leaving only the #!/bin/bash declaration. The script is then updated to include user prompts and input handling. A blank line is added to improve output readability, followed by a prompt asking the user to enter the IP address. The input is captured and stored in the variable ip\_add, which is then used in the Netcat command. By replacing the static IP address with the ip\_add variable, the script becomes versatile, enabling it to scan any specified target without manual modifications. This marks a significant improvement in functionality and usability compared to the initial version.

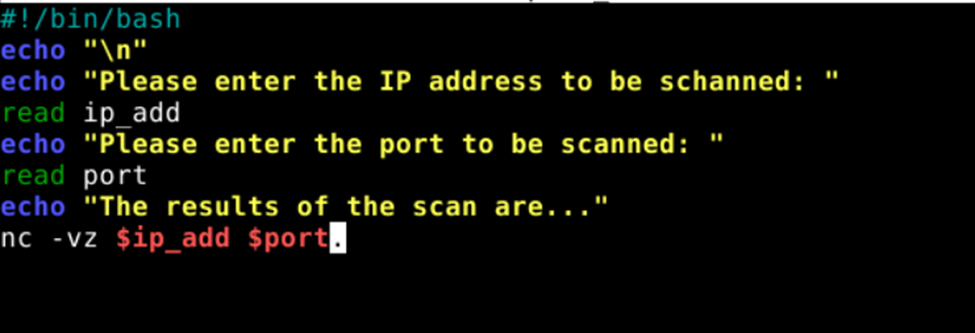


To run the updated port scanner, execute the script by typing sh port\_scanner.sh in the terminal. Unlike the initial version, this iteration prompts you to manually input the IP address to be scanned, adding flexibility and eliminating the need to modify the script for each new target. Once the script runs, it will ask for an IP address, allowing you to enter 172.31.24.10 (or any other target). Pressing Enter will execute the scan for the specified IP, demonstrating the enhanced adaptability and functionality of the script. This improvement makes the tool more practical for diverse scanning needs.



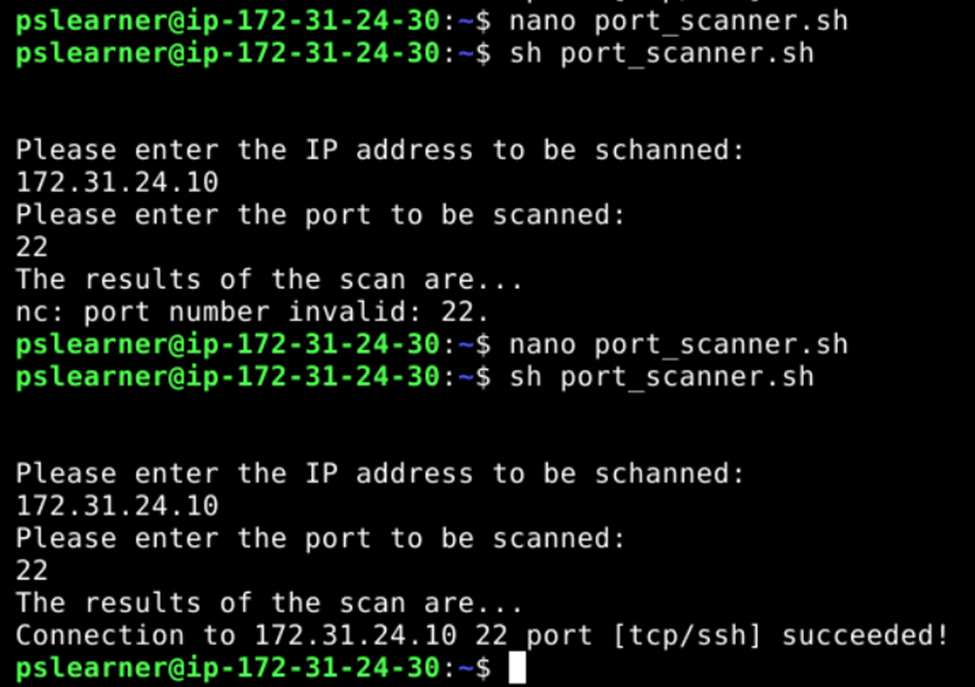
As part of enhancing the functionality of the Bash-based port scanner, the script was further modified to allow dynamic input for both the IP address and the port to be scanned. This improvement involved adding a user prompt for the port number, making the tool more flexible and adaptable for different scanning scenarios. Specifically, a prompt (echo "Please enter the port to be scanned: ") was added to guide the user, followed by a command (read port) to capture the input and store it in a variable named port.

The Netcat command at the end of the script was updated to incorporate this port variable, changing from a static configuration to a dynamic one (nc -vz $ip\_add $port). This enhancement ensures that the script can now scan any port on any target IP address provided by the user, without requiring modifications to the script itself for each new scan. By making the script more interactive, this step significantly increases its usability and applicability, aligning with the overall goal of creating a robust and efficient network scanning tool.



To execute the enhanced port scanner, run the script by typing sh port\_scanner.sh in the terminal. This version of the script now prompts you to input both the IP address and the port to be scanned, making it fully dynamic.

When prompted, type 172.31.24.10 as the IP address and press Enter. Next, type 22 as the port to be scanned and press Enter again. The script will then execute the Netcat command using the provided inputs, scanning the specified port on the target IP address. This demonstrates the improved flexibility of the tool, as it allows different combinations of IP addresses and ports to be scanned without requiring modifications to the script itself.



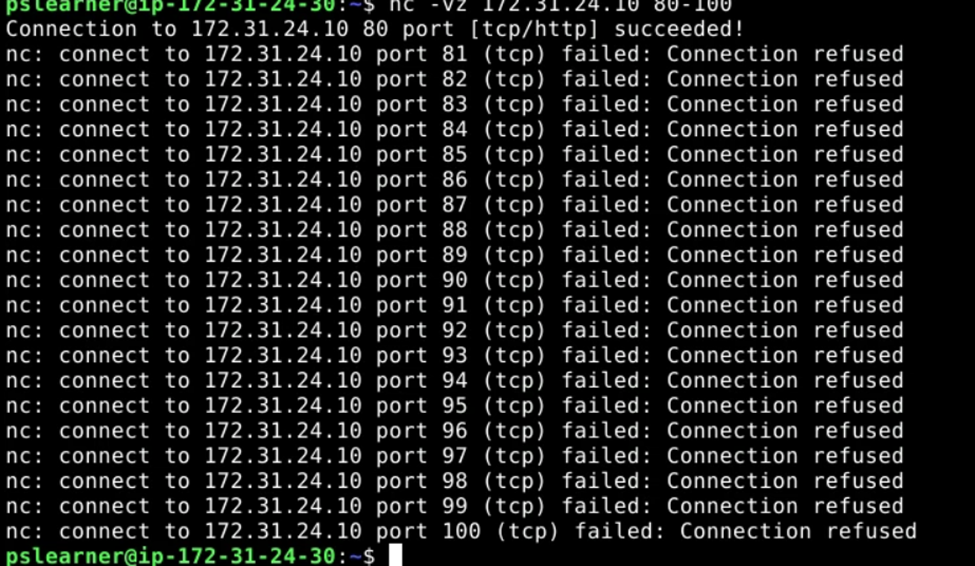
**Build a Robust Port Scanner with Bash**

In this challenge, the focus shifts to creating a more robust and versatile port scanner, building on the foundational skills learned previously. The new scanner will be capable of handling not only single ports but also ranges of ports and multiple non-consecutive ports, showcasing the flexibility of Netcat in handling diverse scanning scenarios. Additionally, the scanner will output the results to both a text file and the console, improving usability and providing a record of the scan.

To begin, open the Terminal application on the Ubuntu Desktop. Test the functionality of scanning a range of ports using Netcat by typing the following command:

nc -vz 172.31.24.10 80-100

This command initiates a scan of ports 80 through 100 on the target machine with the IP address 172.31.24.10. Each individual port within the specified range will be tested for connectivity, and the results will be displayed in the terminal. This step validates the ability of Netcat to handle port ranges, setting the stage for incorporating this functionality into the new scanner script. By testing this command first, you ensure that the concept works as expected before moving forward with automating it in a script.



To begin building the new and improved port scanner, start by creating a new Bash script file named new\_scanner.sh. This script will incorporate the capability to scan a range of ports, enhancing its functionality compared to the previous version.

1. **Create the Script File**: Open a new file in your text editor by typing:

nano new\_scanner.sh

1. **Set Up the Script Header**: On the first line, add the shebang:

#!/bin/bash

This tells the system to interpret the script using Bash.

1. **Prompt for IP Address**: On the second line, add a prompt to request the target IP address:

echo "Please enter the IP address to be scanned: "

1. **Read User Input**: On the third line, include a command to capture the input for the IP address:

read ip\_add

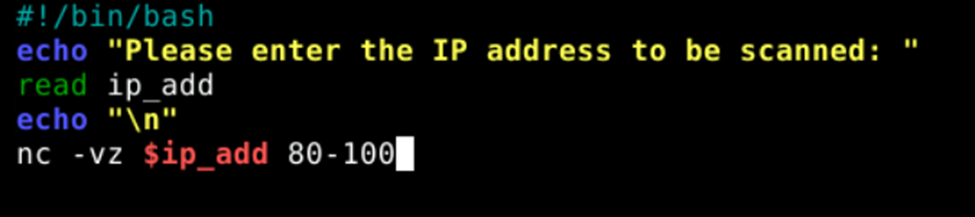
1. **Enhance Readability**: Add a blank line in the output for better clarity by typing:

echo "\n"

1. **Invoke Netcat for Port Range**: Finally, on the next line, add the Netcat command to scan a range of ports (80–100) on the provided IP address:

nc -vz $ip\_add 80-100

This script builds on the functionality developed in earlier challenges by incorporating port range scanning, making it a more comprehensive tool for network assessments. Once the script is complete, save and exit by pressing CTRL + O, then Enter, and CTRL + X. This enhanced scanner allows users to target a range of ports, adding versatility and practicality to the script.

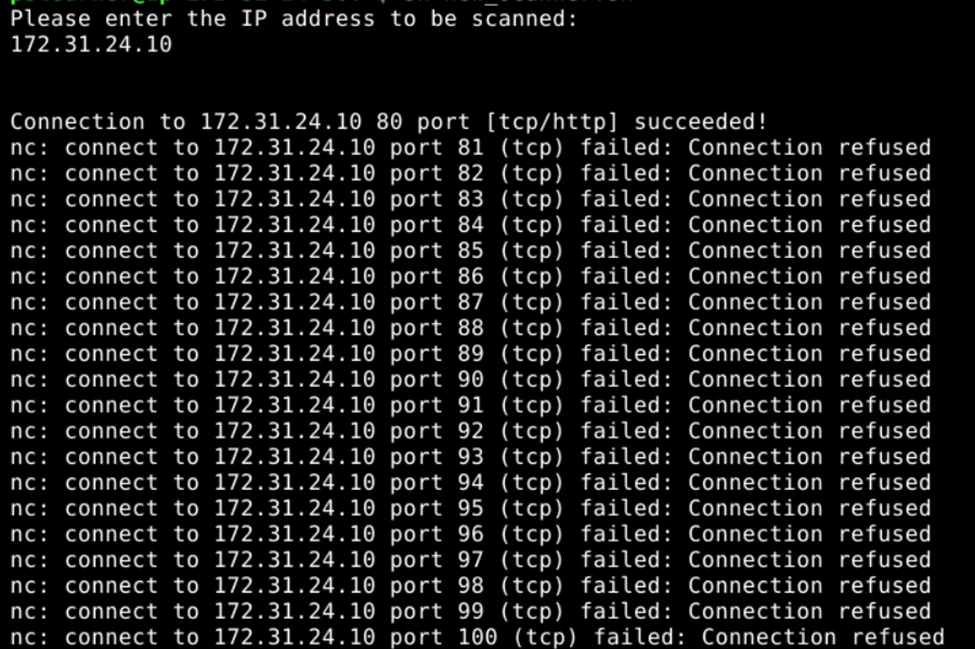


To validate the new script, execute it by typing the following command in the terminal:

sh new\_scanner.sh

When prompted, enter the IP address 172.31.24.10 and press Enter. The script will then execute the Netcat command, scanning the range of ports (80-100) on the specified IP address. The results of the scan will be displayed in the terminal, showing the connection attempts to each port within the range.

This step ensures that the modifications to the script are functioning as expected and that it can successfully handle port range scanning. If the script runs correctly, it confirms the ability to dynamically take input and perform a comprehensive scan of multiple ports on the target machine.



To make the port scanner more dynamic and capable of scanning a range or set of non-consecutive ports, the script can be updated to accept input from a text file containing the desired ports. Follow these steps to modify the script:

1. **Open the Script for Editing**:

nano new\_scanner.sh

1. **Add the Port Range Variable**: Between the lines where the IP address is read (read ip\_add) and the blank line for output formatting (echo "\n"), insert the following line:

port\_range=$(cat ./ports.txt)

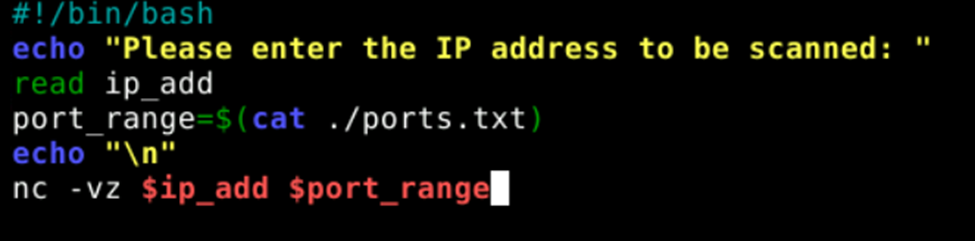
This line reads the contents of a file named ports.txt and assigns it to the port\_range variable. The file will contain a list of ports or port ranges to be scanned.

1. **Update the Netcat Command**: Modify the Netcat command to utilize the port\_range variable. Replace the static port range (80-100) with the following:

nc -vz $ip\_add $port\_range

1. **Explanation of Functionality**: The ports.txt file will act as an external configuration for specifying ports. Users can edit this file to define a range (e.g., 80-100) or a list of non-consecutive ports (e.g., 22 80 443). The script dynamically reads these ports and includes them in the Netcat scan, allowing maximum flexibility without requiring changes to the script.
2. **Save and Exit**: Save the updated script by pressing CTRL + O, then Enter, and exit with CTRL + X.

This enhancement allows the scanner to read and process port information from a file, simplifying the process of updating the ports to be scanned and making the tool more adaptable to various use cases. This is a significant improvement, as users can now define custom scan configurations without modifying the script itself.



To set up the text file for specifying ports, follow these steps:

1. **Create the Ports File**: In the terminal, create a new file named ports.txt:

nano ports.txt

1. **Enter the Common Ports**: On the first line of the file, enter the following ports, separated by spaces:

21 22 23 25 53 80 110 111 135 139 143 443 445 993 995 1723 3306 3389 5900 8080

1. These ports represent commonly used services and protocols, such as FTP (21), SSH (22), HTTP (80), HTTPS (443), and others.
2. **Save and Exit**: Save the file by pressing CTRL + O, then Enter, and exit with CTRL + X.

This design simplifies the process of scanning multiple ports by keeping the port configuration external to the script, ensuring ease of use and adaptability for future needs.

To validate the newly enhanced port scanner, follow these steps:

1. **Run the Script**: Execute the script by typing:

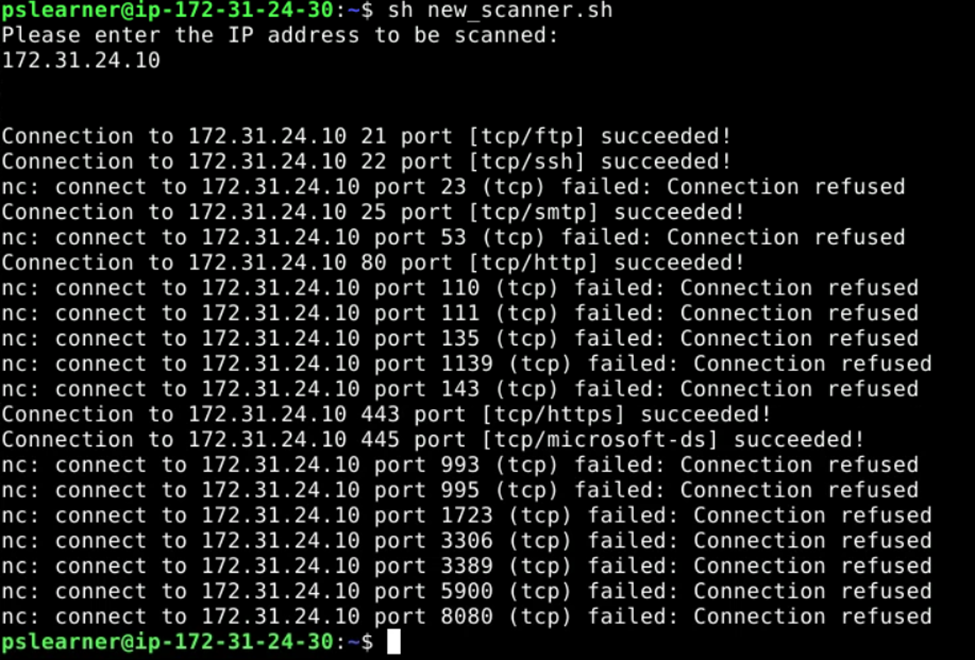
sh new\_scanner.sh

1. **Provide the IP Address**: When prompted, enter the target IP address:

172.31.24.10

1. Press **Enter** to proceed.
2. **Observe the Output**: The script will read the list of ports from the ports.txt file and use Netcat to scan each specified port on the target IP address. The results of the scan will be displayed in the terminal, showing the connection attempts and any successful responses.
3. **Verify Functionality**: Check that the script correctly scans all the ports listed in ports.txt and outputs the results as expected. Each connection attempt should be displayed, and any open ports will be noted as "succeeded."

This test confirms that the script successfully integrates the dynamic port configuration, allowing it to scan multiple ports specified in an external file, making the tool more flexible and user-friendly.



o modify the script for listing only the open ports and saving the results to a text file, follow these steps:

1. **Edit the Script**: Open the scanner script for editing:

nano new\_scanner.sh

1. **Add a Summary Line**: After the Netcat command, add the following line to indicate the scan has completed and display the target's open ports:

echo "Scanning completed, the following ports are open on the target $ip\_add: "

1. **Add a Blank Line for Readability**: Add a blank line in the output to improve clarity:

echo "\n"

1. **Filter and Display Open Ports**: Use grep to filter the scan results and display only the open ports. Modify the script to display the relevant lines from the text file:

cat scan\_$ip\_add.txt | grep "succeeded"

1. **Add Another Blank Line**: Finally, add one more blank line for neatness:

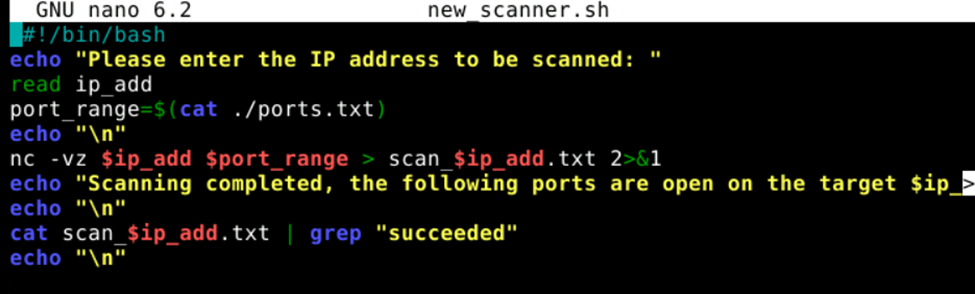
echo "\n"

1. **Save and Exit**: Save the updated script by pressing CTRL + O, then Enter, and exit with CTRL + X.

### **How This Works:**

* **Text File Output**: The Netcat results are saved in a file named scan\_$ip\_add.txt.
* **Filtered Output**: The grep "succeeded" command extracts only lines indicating successful connections (open ports).
* **User Feedback**: The script outputs a user-friendly summary of open ports in the terminal while maintaining a full record in the text file for reference.

This enhancement ensures the scanner provides concise and relevant output, making it easier for the user to identify open ports while preserving detailed logs for further analysis.



This enhancement allows the script to provide a clear and concise output of open ports directly in the terminal, making it easier to quickly identify which ports are accessible on the target system. At the same time, storing all scan results in a text file ensures that a comprehensive log is available for detailed analysis or future reference. This dual functionality improves usability and ensures efficient reporting of scan results for network assessments.

