Networks

Tópicos de Informática para Automação

Mário Antunes

November 03, 2025

Exercises

Practical Lab: Exploring and Using the Network

Objective: This lab will guide you through the practical fundamentals of networking. You will inspect your local network, explore the wider internet, scan for services, and use professional tools like SSH and `rsync`.

Part 0: Setup & Preparation

Before you begin, you must set up your environment with the necessary tools and a target to work with.

1. Install Networking Tools (Linux) Open your terminal and install the following packages, which contain the tools for our exercises:

```
$ sudo apt update
$ sudo apt install -y nmap traceroute dnsutils curl python3-pip python3-tk
```

2. Create an SSH Keypair We will use key-based authentication, the most secure method for logging into remote servers.

```
$ ssh-keygen -t ed25519 -N ""
```

This creates a **private key** (`~/.ssh/id_ed25519`) and a **public key** (`~/.ssh/id_ed25519.pub`). **NEVER** share your private key.

3. Launch a Secure Target (SSH Server) We need a "remote" server to connect to. We will use Docker to launch a simple, pre-configured SSH server container.

1. Create a folder named `ssh-server` and `cd` into it.
2. Create a file named `custom-openssh-server.Dockerfile`:

   ```
   FROM lscr.io/linuxserver/openssh-server:latest
   RUN apk update && apk add rsync && rm -rf /var/cache/apk/*
   ```

3. Create a file named `compose.yml`:

   ```
   services:
     ssh:
       build:
         context: .
         dockerfile: custom-openssh-server.Dockerfile
       container_name: ssh
       environment:
         - PUID=1000
         - PGID=1000
         - TZ=Europe/Lisbon
         - USER_NAME=student
   ```

```
      - PUBLIC_KEY_FILE=/config/authorized_keys/student.pub
    volumes:
      - ./authorized_keys:/config/authorized_keys
    ports:
      - "2222:2222" # Map Host port 2222 to Container port 2222
    restart: unless-stopped
```

4. Create a folder for your public key: `mkdir authorized_keys`
5. Copy your public key (from step 2) into this folder so the server will trust you:

   ```
   $ cp ~/.ssh/id_ed25519.pub ./authorized_keys/student.pub
   ```

6. Start the server:

   ```
   $ docker compose up -d
   ```

   You now have an SSH server running on `localhost` at port 2222.

Part 1: Local Network Exploration (LAN)

Exercise 1: Find Your IP Address Use the `ip` command to find your computer's logical address.

```
$ ip addr show
```

- Identify your main network interface (e.g., `eth0` or `wlan0`).
- Find your **IPv4 address** (e.g., `192.168.1.50/24`). The `/24` is your **subnet mask**.
- Find your **MAC address** (e.g., `link/ether 0a:1b:2c:3d:4e:5f`).

Exercise 2: Check the Loopback Interface Test that your computer's internal network stack is working. The "loopback" interface (`127.0.0.1` or `localhost`) is a virtual interface that points to your own machine.

```
$ ping 127.0.0.1 -c 4
```

You should get an instant reply. This confirms your system's networking service is active.

Exercise 3: Find Your Default Gateway The "Default Gateway" is your router's IP address—the "door" from your LAN to the internet.

```
$ ip route show default
```

- The output will be something like `default via 192.168.1.1 dev eth0`.
- Now, `ping` that address to confirm you can reach your router.

  ```
  $ ping 192.168.1.1 -c 4
  ```

Exercise 4: View the ARP Table Now that you've pinged your gateway, your computer knows its physical MAC address. Use the **Address Resolution Protocol (ARP)** table to see this mapping.

```
$ arp -n
```

You will see a list of local IPs and their corresponding MAC addresses. This is how your switch knows where to send local packets.

Part 2: Exploring the Wide Area Network (WAN)

Exercise 5: Test External Connectivity (`ping`) Check if you can reach a server on the internet and measure your **latency** (the round-trip time).

```
$ ping google.com -c 4
```

- Note the `time= ...` value (e.g., `time=15.2 ms`). This is your latency to Google's servers.

Exercise 6: Trace the Path (`traceroute`) Find out the exact path your data takes to reach a server. `traceroute` shows every "hop" (router) along the way.

```
$ traceroute 8.8.8.8
```

You will see a list of IP addresses, starting with your own router, then your ISP's routers, and finally Google's.

Exercise 7: Query the "Phonebook" (`dig`) Use **DNS** to translate a domain name into an IP address.

```
$ dig google.com
```

- Look in the "ANSWER SECTION" to find the A record (the IPv4 address).
- **Bonus:** Find the mail servers for `google.com` by querying the `MX` record.

  ```
  $ dig google.com MX
  ```

Part 3: Service Discovery (`nmap`)

Exercise 8: Scan Yourself Use **Nmap** (Network Mapper) to scan your own computer for open ports.

```
$ nmap localhost
```

You will likely see port 2222 open because of the SSH server you started in Part 0.

Exercise 9: Scan a Public Server Let's see what ports are open on a public website. `scanme.nmap.org` is a server *specifically* for practicing Nmap.

```
$ nmap scanme.nmap.org
```

- What services are running? This output shows you why ports `80` (HTTP) and `22` (SSH) are important.

Part 4: Secure Remote Operations (SSH & `rsync`)

Exercise 10: Connect to Your Server (`ssh`) It's time to log in to the SSH server you launched in Part 0. Your public key is already authorized.

```
# Connect to user 'student' on localhost at port 2222
$ ssh student@localhost -p 2222
```

- You are now *inside* the Docker container.
- Run commands like `whoami`, `ls -l`, or `pwd` to prove you are in a different environment.
- Type `exit` to log out.

Exercise 11: Sync Files Securely (`rsync`) `rsync` is the best way to copy files over SSH. It's smart and only copies the differences.

1. On your **host machine**, create a new folder and a file.

   ```
   $ mkdir my-project
   $ echo "This is a test file" > ./my-project/README.md
   ```

2. Use `rsync` to "push" this folder to the server's home directory.

   ```
   # Note the -e flag to specify the SSH port
   $ rsync -avzP -e "ssh -p 2222" ./my-project student@localhost:~/
   ```

3. Log back into the SSH server (`ssh student@localhost -p 2222`) and run `ls`. You will see `my-project` has been copied.

Part 5: Project - Geo-Traceroute

Exercise 12: Build a Visual Traceroute In this project, you will combine `traceroute`, a public API, and a mapping library to visualize the physical path your data takes across the world. Download the code from here.

The code does the following:

1. Runs `traceroute` and parses the IP addresses of each hop.
2. Looks up the geographic location of each IP using the `ip-api.com` API.
3. Caches results in a `cache.json` file to avoid re-querying and hitting rate limits.
4. Plots all the points and the path on an interactive.

Try it, follow the `README.md` instructions to setup the project.

Bonus Exercise: SSH Tunneling

Let's use advanced SSH port forwarding to securely access a "hidden" web server.

1. Log into your SSH container:

   ```
   $ ssh student@localhost -p 2222
   ```

2. **Inside the container,** run a simple web server on port 8000. This port is *not* exposed in your `docker-compose.yml`, so it's inaccessible from your host.

   ```
   # Inside the SSH container
   $ cd ~
   $ echo "Hello from inside the tunnel!" > index.html
   $ python3 -m http.server 8000
   ```

3. Open a **new host terminal** (leave the server running).

4. Create an SSH tunnel. This command says "forward *my* local port 8080 to `localhost:8000` *on the remote server*."

   ```
   # In a NEW host terminal
   $ ssh -N -L 8080:localhost:8000 student@localhost -p 2222
   ```

   (The `-N` flag just opens the tunnel without starting a shell).

5. Open your **host's** web browser and go to `http://localhost:8080`.

6. You should see the "Hello from inside the tunnel!" message. You have successfully accessed a hidden port through a secure SSH tunnel.