# Secure SHell: Port Forwarding

## Network Infrastructures labs

Marco Spaziani Brunella


SAPIENZA
UNIVERSITÀ DI ROMA

# Lecture details

- **Readings:**
  - SSH: The Definitive Guide; D.J. Barret et al.; O'Reilly
- **Lecture outline:**
  - SSH local port forwarding
  - SSH remote port forwarding

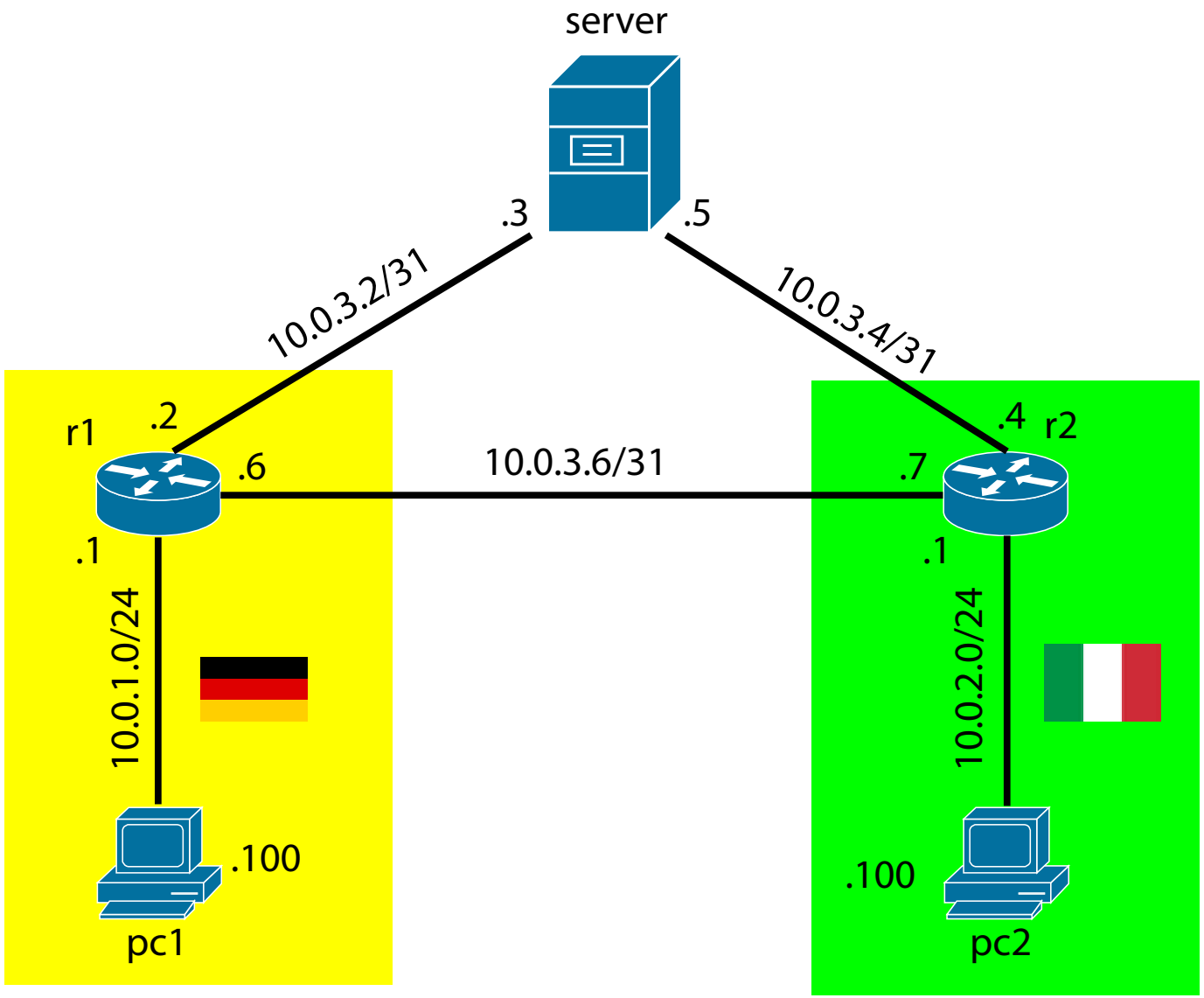
# Local Port Forwarding scenario

Suppose you are abroad (e.g. Germany) and the server you wish to connect has some regional restrictions (e.g. Netflix) that does not allow you to visit a very important page that you could actually access in Italy.

How to solve that? → SSH Local Port forwarding

The only thing we need is that our home PC has an SSH server running and that we can access with a user (not necessarily with asymmetric auth).

# Local Port Forward Topology



# SSH Local Port Forwarding

If we try to access server using *links*, we get the German page of the site!
Lets first create an "user" with password "user" on pc2 and start the ssh daemon.
We now want to redirect all the traffic directed to the 80 port of server trough an SSH tunnel over 9000 from pc1 to pc2:

```
pc1:~$ ssh -NL 9000:10.0.3.5:80 user@10.0.2.100
```

In general, we have:

```
pc1:~$ ssh -NL local_p:host_ip_addr:remote_p user@tunnel_ip
```

Where:

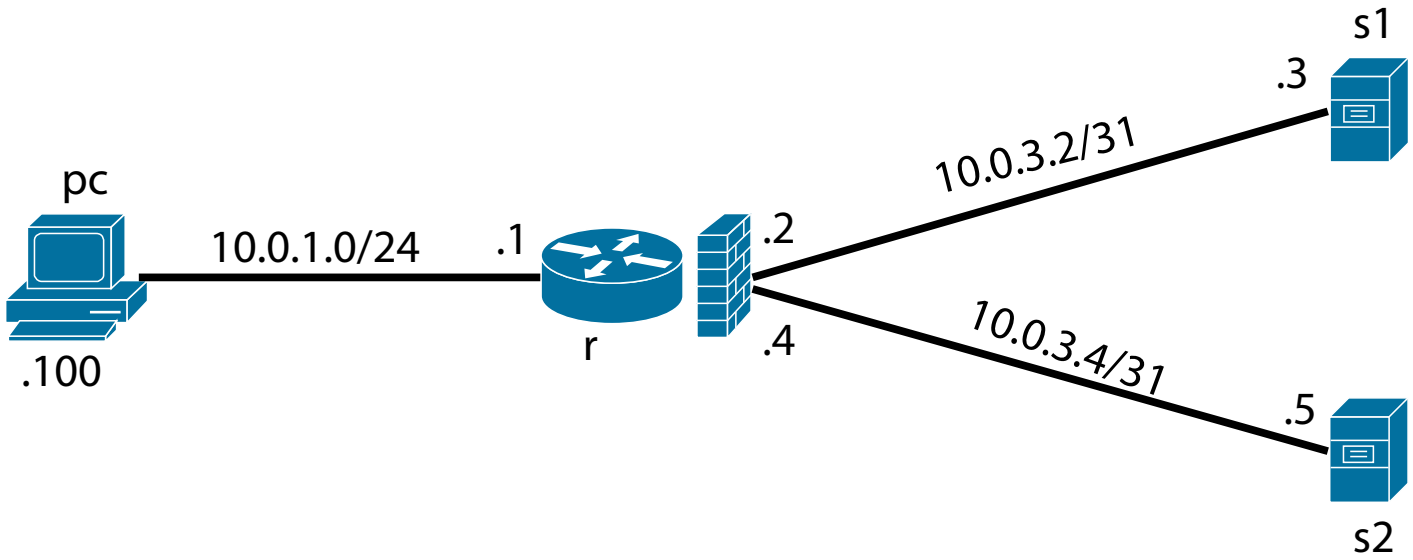
- -N no shell
- -L local port forwarding

# Remote Port Forwarding scenario

Suppose you are on a LAN and your default gateway is blocking all the traffic on port 80 (using iptables for example) but you have a nice web app on your local machine you wish to show to some other. Suppose now you have also a server with a public IP (e.g Amazon AWS).

How can you bypass firewall? → Remote port forwarding

# Remote Port Forward Topology



# SSH Remote Port Forwarding

If we try to access pc using *links* from s1 and s2, we do not get anything!
Lets first create an "user" with password "user" on s2 and start the ssh daemon.
We now want to redirect all the traffic directed to the 9000 port of s2 to the 80 port of pc, trough an SSH tunnel from pc to s2:

```
pc1:~$ ssh -NR 9000:10.0.3.5:80 user@10.0.2.100
```

In general, we have:

```
pc1:~$ ssh -NR remote_p:remote_ip_addr:local_p user@tunnel_ip
```

Where:

- -N no shell
- -R remote port forwarding

If we now want to access pc from s1, we need an SSH local port forward from s1 to s2!!!

```
pc1:~$ ssh -NL 80:127.0.0.1:9000 user@10.0.3.5
```