Lab 4:

Linux network with   
IP forwarding

Linux Server Security  
 2024-2025

© Daan Pareit

## Introduction

# Lab concept

We will expand our previous network to the topology below by cloning a VM and putting it in another network:



# Learning goals

* Cloning a VM and changing its identity settings
* Basic network segmentation
  + Linux IP forwarding
  + Custom routes
  + Configuring pfSense

# Practicalities and prerequisites

You’ll need the following:

* We will build upon the network you created previously. First take snapshots of your VMs to revert to these if you would mess things up in the next steps…

## Create another VMnet in VMware Workstation/Fusion

We will connect our Debian machine to yet another network in a minute. First, we’ll create that other virtual network and virtual switch by creating another VMnet in VMware:

1. In VMware Workstation, go the ***Virtual Network Editor*** and again add a new VMnet (e.g. VMnet6 or whatever unused VMnet so far)
2. Set this VMnet to be host-only and give it subnet **192.168.22.32/27.** Note that we use another prefix here ( /27 instead of /24 ). Recap ‘Computer Networks’: an IPv4 IP network address doesn’t always end with ‘.0’ (it’s .32 in this case).

A screenshot of a computer screen

Description automatically generated

1. You can disablethe DHCP server for this subnet. (We’ll only use fixed IP addresses here.)
2. Leave the ‘connect a host virtual adapter to this network’ turned on. It will assign a fake NIC to your laptop using the **first** available host IPv4 address within the 192.168.22.32/27 subnet. Which IP address is this?

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

192.168.22.33 !

## Give your original Debian an extra interface

🡺 In the VM settings, add another network interface to your debian VM and connect it to the VMnet you’ve just created (the 192.168.22.32/27 subnet of e.g. VMnet6).

A computer screen shot of a program

Description automatically generated

🡺 Check with the ip tool in your debian VM for the name of the new interface.

ens37

A screenshot of a computer program

Description automatically generated

🡺 We’ll assign that interface the **last available host IPv4 address** within the 192.168.22.32/27 subnet. What IP address is that (recap ‘Computer Networks’)?

192.168.22.62!

A screen shot of a computer

Description automatically generated

Yeah I am super lazy but I have tools, hence why not use them, right?

🡺 Change the network settings in your debian VM to give your new interface that fixed IP address via ifupdown (the /etc/network/interfaces file) or systemd-networkd (a /etc/systemd/network/<2digits>-<some\_name>.network file and optional .link file), depending on what method you preserved during last lab.

cd /etc/systemd/network

sudo vim 30-static.network

A screen shot of a computer

Description automatically generated

🡺 Now apply these changes and bring the new interface up (cfr previous lab if you forgot the command). Now, your debian VM will have two interfaces, each having its own fixed IP address.

sudo networkctl reload

🡺 Verify both IP addresses and the routing table via the ip tool.

A screenshot of a computer program

Description automatically generated

🡺 Test your settings by pinging from your laptop to debian’s second IP address.

A screenshot of a computer program

Description automatically generated

Without the Gateway - it does not work.

## Create another Debian machine via VM cloning…

🡺 We need a fourth VM, which will put within the new 192.168.22.32/27 network: ‘debian-extra’. Shut down your ‘Debian’ VM, then clone it. Choose to take a ‘full clone’ instead of a ‘linked clone’ (requires more disk space but debian has a relatively small footprint and a full clone could avoid more issues later).

Now make your clone different than the original one:

🡺 The cloned only needs to be connected to the newly created network. In the VM settings, keep the network interface connected to the VMnet with the 192.168.22.32/27 IP subnet (e.g. VMnet6). Remove the other interface from the VM.

A screenshot of a computer

Description automatically generated

🡺 You need to change the hostname of your clone from ‘debian-<firstname>-<lastname>’ to ‘**debian-extra-<firstname>-<lastname>**’. In which file is the hostname specified? You could change this file manually, but even better is to use hostnamectl with its appropriate option. Afterwards, you’ll need to logout and login again to view the updated hostname in your shell prompt.

🡺 Sometimes, your hostname is also being resolved locally. Thus change your hostname also in the /etc/hosts file.

easiest way is to just change /etc/hosts and /etc/hostname

A screen shot of a computer

Description automatically generated

è change the SSH host keys (to have different SSH host keys than the original debian VM). Consult <https://wiki.debian.org/SSH#Regenerating_host_keys> to do so.

sudo rm /etc/ssh/ssh\_host\_\*

sudo dpkg-reconfigure openssh-server

🡺 We’ll assign that interface the **192.168.22.42** IP address which lies within the 192.168.22.32/27 subnet. Why did we pick that one? Well, we had to pick one arbitrarily for your setup and try googling “the answer to life the universe and everything” 😉

LOOOL

A screenshot of a calculator

Description automatically generated

🡺 Change the network settings in your debian-extra VM to change the fixed IP address to that IP address. You can make that change via ifupdown (the /etc/network/interfaces file) or systemd-networkd (a /etc/systemd/network/<2digits>-<some\_name>.network file and optional .link file), depending on what method you preserved during last lab. Set the default gateway to the fixed IP address of your original debian VM within this 192.168.22.32/27 subnet.

A screenshot of a computer program

Description automatically generated

🡺 Test your settings by pinging from your laptop to debian-extra’s IP address.

A screenshot of a computer program

Description automatically generatedAlas, it works!

A screenshot of a computer screen

Description automatically generated

## Give your original Debian routing capabilities

🡺 Start capturing ICMP packets on RHEL with “tcpdump -i ens160 icmp“. Now on debian-extra start a ping to RHEL. Are the ping echo requests arriving at your RHEL?

Of course not, because the machine can only see the other machine, which is Debian (.11.10), and hence the packages do not get forwarded.

A screen shot of a computer

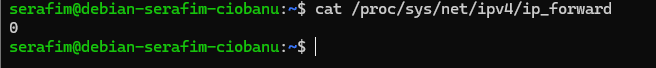
Description automatically generated

This is because your linux is configured by default as an end host. To allow it to function as a *router* and forward IP packets from one network to another network (or VMnet), you’ll need to enable this functionality.

🡺 Check the current status of IP forwarding of your debian VM with:  
*sysctl net.ipv4.ip\_forward*  
or  
*cat /proc/sys/net/ipv4/ip\_forward*

A screen shot of a computer program

Description automatically generated



The first method is the preferred choice, but the second method reveals you this setting as a ‘file’. This file is however no ordinary file, but addresses via the /proc file system the corresponding kernel driver directly, to get the value of the global kernel variable net.ipv4.ip\_forward .

🡺 Now change this value to 1 to allow IP forwarding. This could be done with:  
*echo 1 > /proc/sys/net/ipv4/ip\_forward*  
However, better is to use sysctl. Look in the manpage of sysctl how to change this setting via *sysctl* .

A screen shot of a computer

Description automatically generated

sudo sysctl -w net.ipv4.ip\_forward=1

🡺 The IP forwarding setting is discarded when rebooted. How to make this change persistent? Hint: note the referral to a config file in the ‘see also’ section at the bottom of the manpage of sysctl

it is going to tell about sysctl.conf

man systctl.conf

vim /etc/sysctl.conf

uncomment the net.ipv4.ip\_forward=1



🡺 Capture ICMP packets on RHEL again using tcpdump and ping again from debian-extra to RHEL. Are the ping echo requests arriving at your RHEL?

No,

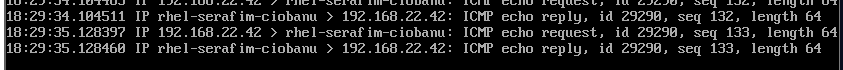
A screenshot of a computer

Description automatically generated

## Make the RHEL VM aware of the new network

Now, in the last test, did RHEL create ping echo replies? Are these received by debian-extra?

RHEL is sending the echo replies, however, they do not get to the machine.



🡺 Where is RHEL the replies wrongly sending to? Check the routing table via the ip tool.

The machine has only one connection, and that is to the Debian and pfsense network. Hence, it can not route the packets to the other network.

🡺 Your RHEL VM needs to know that it can reach the new VMnet network (and the debian-extra VM therein) via the debian VM which you’ve set to act as router (by enabling IP forwarding). Therefore, on RHEL, you’ll need to add the route to the 192.168.22.32/27 subnet which will have to pass through the debian VM (which is why we enabled IP forwarding on it). You can do this via the nmtui tool. Don’t forget to reload this new config with nmcli to apply your changes (see previous lab if you forgot the command).

sudo nmcli connection reload

sudo nmcli connection down ens160 && sudo nmcli connection up ens160

A computer screen shot of a computer

Description automatically generated

So the whole idea was to set the destination (as in the actual machine) to the Vmnet3, and the “Next Hop” to actually take the next hop to the Debian machine (since it is the forwarding machine)

A screenshot of a computer

Description automatically generated

🡺 Verify the routing table via the ip tool afterwards, to see the route to 192.168.22.32/27.

A screen shot of a computer

Description automatically generated

Access the other Vmnet via the Debian machine… it started to make so much sense.

🡺 Test your settings by pinging from debian-extra to RHEL and vice versa.

A screenshot of a computer screen

Description automatically generated

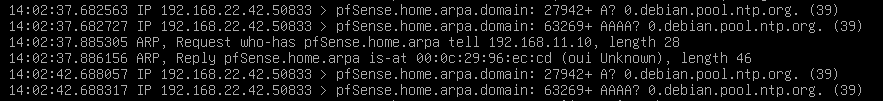
Now I can ping both ways!! Hooray

## Make the pfSense VM aware of the new network

🡺 Make sure all the previous steps were correct. Now on, your debian-extra, ping following addresses: IP address of debian within 192.168.22.32/27, IP address of debian within 192.168.11.0/24 and IP address of pfSense within 192.168.11.0/24. Which ones are replying? While pinging, use the tcpdump tool on your debian to find out if your pings are being sent on the interface in debian’s 192.168.11.0/24 subnet.

ping 192.168.22.62 - works (sent on .22.32 interface)

ens33



ens37



ping 192.168.11.10 - works (shows only on the .22.32, probably because it comes from that network)

ens33

A black screen with white text

Description automatically generated

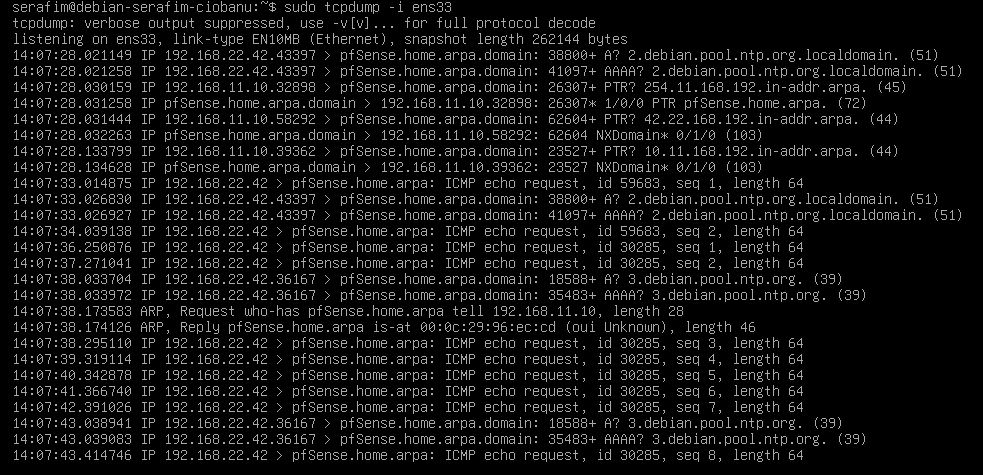
ens37

A screen shot of a computer

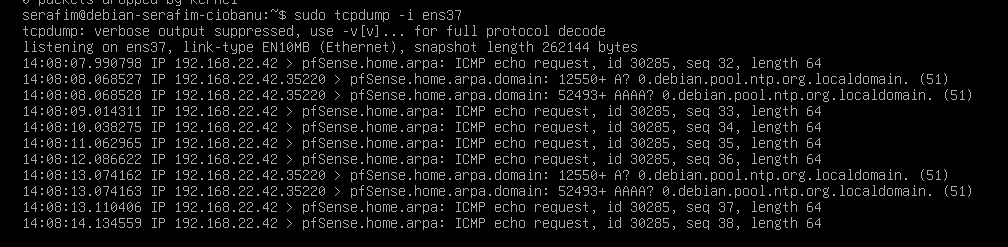
Description automatically generated

ping 192.168.11.254 - does not work (request showing in both interfaces, because it is received from one network, and needs to be sent to the other one.

ens33



ens37

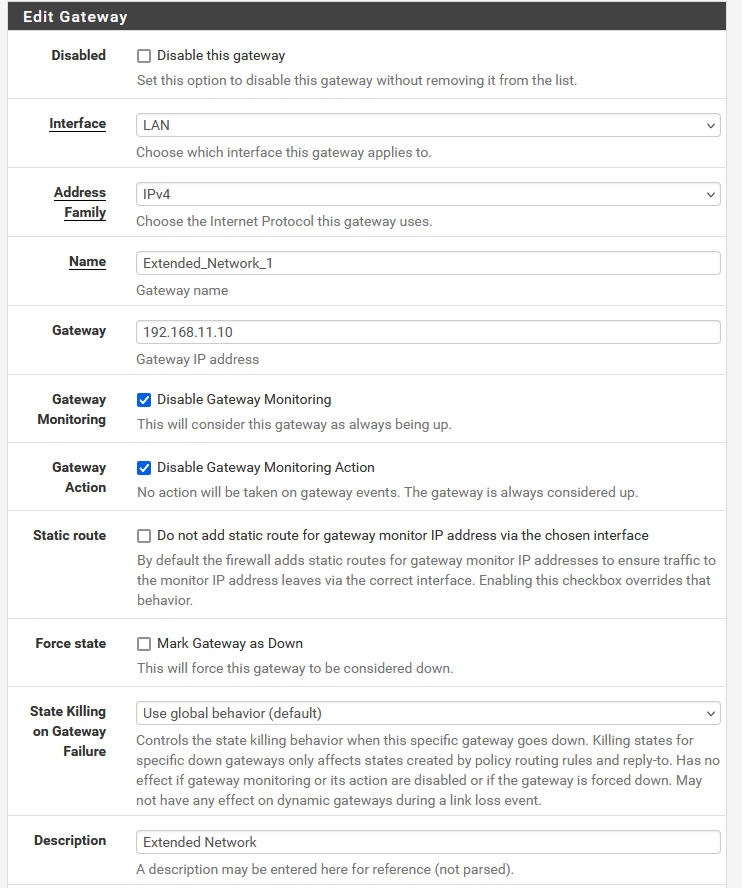


The requests are sent though, and looks like they are achieving it.

The reason for the above failing ping is that your debian-extra knows how to reach the pfSense (thus echo ping requests from debian-extra reach the pfSense), but pfSense doesn’t know how to answer back to debian-extra, because it doesn’t know where the 192.168.22.32/27 network is located. Thus, a similar problem as we just had with our RHEL.  
We thus need to ensure that the pfSense VM also needs to know how it can reach the VMnet6 network (and the debian-extra VM therein). This is also via the debian VM which acts as a router. Therefore, you’ll need to add the route to the 192.168.22.32/27 subnet in pfSense. You can do this via the web interface of pfSense at the “System / Routing” tabs via the following steps:

🡺 In the “System / Routing” section of the web interface, add a new “Gateway” on the LAN interface to tell pfSense that there are other subnets ‘behind’ the 192.168.11.10 address of your Debian VM. You can disable the ‘Gateway Monitoring’ and the ‘Gateway Action’. (Note: make sure you did this at the “System / Routing” section of the web interface, not at the “Interface / LAN”. Otherwise, you would have set the gateway for ALL traffic to go via Debian.)

System > Routing > Gateways



🡺 Add a static route to 192.168.22.32/27 over this 192.168.11.10 LAN gateway.

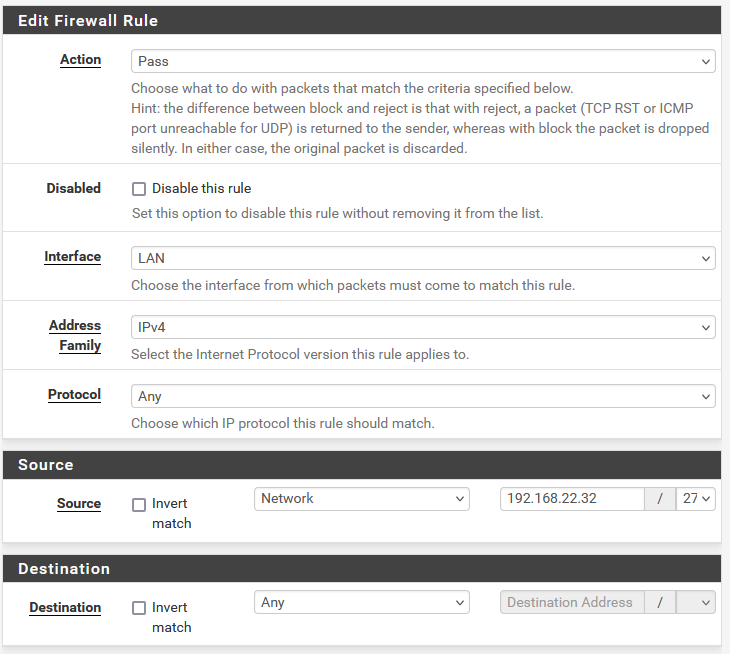
System > Routing > Static Routes

A screenshot of a computer

Description automatically generatedAll we say, is that the Debian machine acts as a Gateway to the “outer world”, and then we can access a specific network in that “outer world”

🡺 Create a firewall rule to allow **any** 192.168.22.32/27 source traffic on the LAN interface of pfSense.

Firewall > Rules > LAN

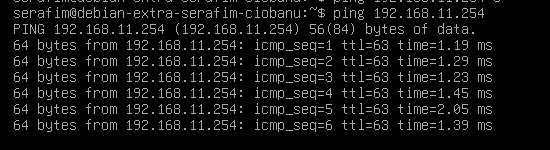


PROTOCOL IS IMPORTANT

🡺 Test your settings by pinging from debian-extra to pfsense. And from pfsense to debian-extra (via de ping tool in the web interface).

A screen shot of a computer

Description automatically generated



IT WORKS!!!!

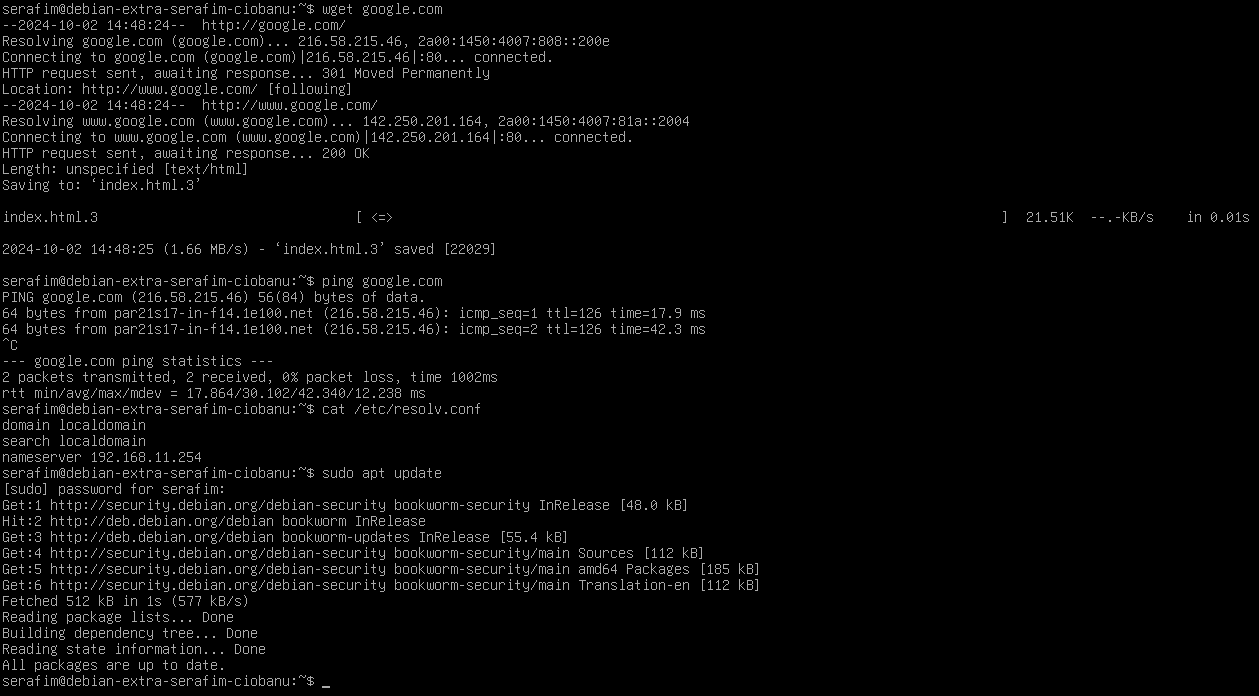
Now IP connectivity to and from your new virtual network should be OK. Let’s move a step up in the OSI model and check if name resolving and HTTP works.

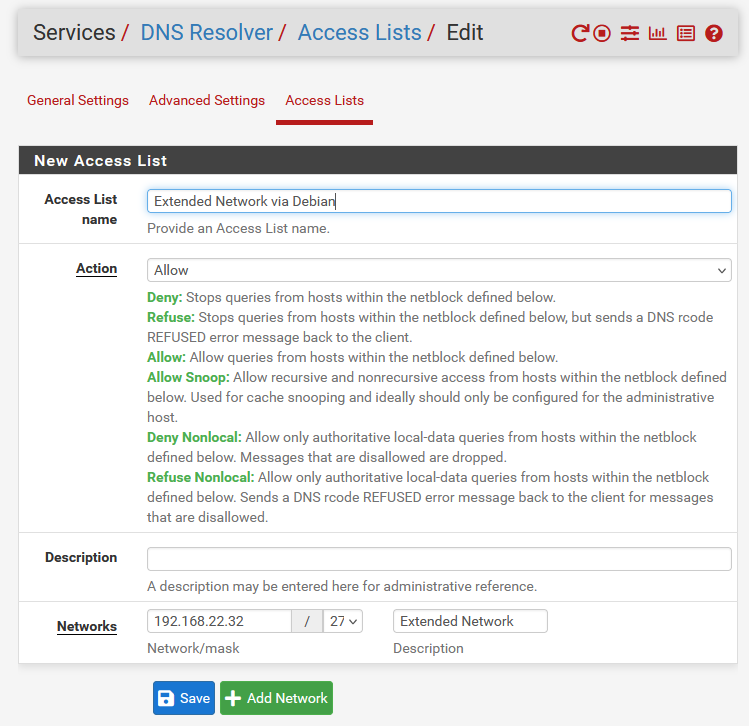
🡺 Test if you can successfully wget google.be on debian-extra. If it doesn’t, use the host tool, to DNS query the IP address of google.be . Compare the output on debian-extra to the output on debian.

🡺 Because the 192.168.22.32/27 network isn’t directly attached to pfSense’s LAN network, the DNS Resolver doesn’t allow DNS queries from that network. You need to explicitly allow that network in the “Access Lists” section in the “DNS Resolver” service in the pfSense web interface.

🡺 Test if you can now successfully wget google.be or run apt update on debian-extra.

Probably due to manual changes in the /etc/resolv.conf file, my DNS server copied from the Debian machine.





I did it anyway.

## Final checks

🡺 Try temporarily disabling the IP forwarding on debian or deleting the route on RHEL to see their importance.

🡺 Reboot all machines and check if they can all ping each other and if they still all have Internet connectivity to ensure if you correctly set all changes to be permanent.

Seems like everything works amazingly!

🡺 If all works fine, taking snapshots might be a good idea 😊 (best to take when they are powered off)

Best practice from now on: avoid working in the VMware window. This is the equivalent of being physically present in a datacenter with a screen/keyboard attached to your server. Chances are unlikely this will happen a lot. In real life, you’ll have to login remotely to your servers. Therefore, use the ‘ssh’ command within a terminal on your laptop (e.g. Powershell), e.g.: ssh <user>@192.168.11.10

A black screen with white text

Description automatically generated