IT Technology

Assignment 53

Raspberry Pi as a virtual machine and wepserver.

Author
Emil C. Simonsen
ecsi38572@edu.ucl.dk

Thursday 10. Marts 2022

Table of Contents

Introduction

The goal is to make a raspberry pi virtual machine host a wepserver, using Nginx, construct a small python script to run it and access it form another machine on the network

Learning goals

- How to create a RaspberryPi VM.
- What a RaspberryPi VM is and what the purpose of it is.
- What a Web Server is and what the HTTP application layer protocol is.
- Very fundamentals of what HTML is.
- How to make a basic dynamic web page by means of Python.
- How to set up a Raspberry Pi as a VM Virtual Machine and as a webserver.

Tasks

1. Create a network diagram with explanation. Network diagram and Configuration(s) must also be linked to in a GitLab repository.
2. Set up the network.
    - Configure the router.
    - Allow PC1 to access PC2 but not vice versa.
      Hint: Put ge-0/0/1 and ge-0/0/2 in two different zones and then set up policies to control traffic between these zones.
      .
3. Install the PC2 RaspberryPi VM.
    - Get the RaspberryPi .iso file.
    - Install the RaspberryPi on VMware Workstation.
    - Configure the network settings on the Raspberry for the given network.
    - Install necessary software. E.g. Wireshark.
      .
    - Install a Web Server on the PC2 RaspberryPi. E.g. the Nginx webserver.
    - Test the webserver.
      .
    - Run the Python server: `$ python3 -m http.server`
    - Create an .index test Web Page in HTML. Something very basic.
    - Test that the Web Server is reachable from PC1. Show the Python Webserver .index page and show the resulting page in a browser.
      .
    - Use Wireshark to monitor the HTTP traffic to and from the Web server with explanation and highlighting of the following layer information:

    a. Layer 2: Highlight source and destination MAC addresses.
    b. Layer 3: Highlight source and destination IP addresses.
    c. Layer 4: Highlight source and destination Ports.
    d. Layer 5: Highlight application layer data.
    .
    Write two lines of commenting for each layer item.
4. Explain in two lines what the application layer HTTP protocol is. Make a drawing to support the explanation.

Audience
Anybody looking to learn how to set up a wepserver using Nginx, python and Wmww

Inventory
- PC
- WMWW
- Raspberry Pi OS
- Python3
- NginX

-

1. Networking diagram

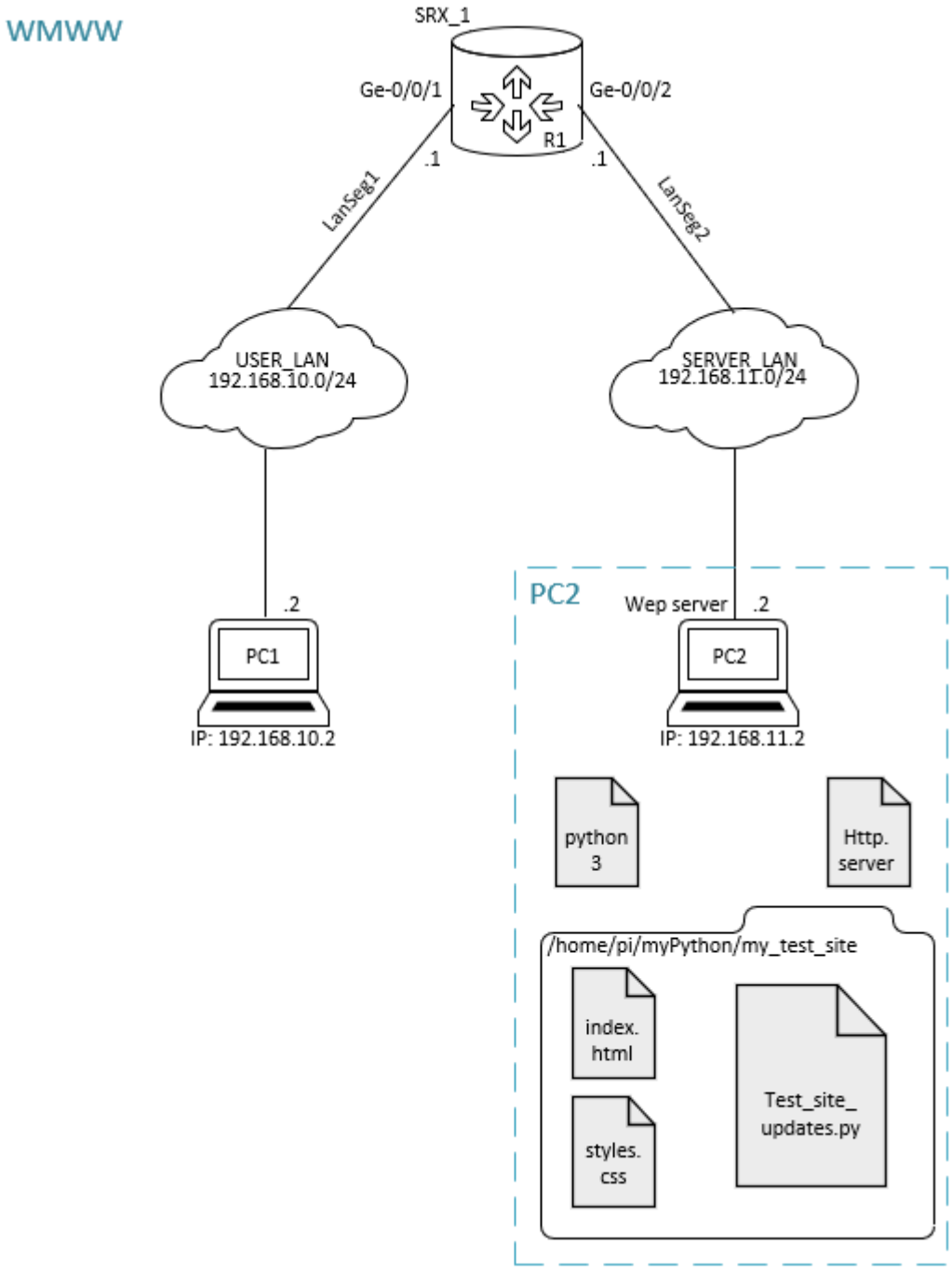

Figure 1: Networking diagram: Two subnets with a PC on each, one PC is hosting a webserver

2. Setting up the network.

To see how to set up a network with two subnets and one router, have a look at assignment 10[1]

Once we have our basic network, we can edit the configuration to only allow traffic going from PC1 to PC2, we do this by putting each machine in its own security zone. I called my zone for USER_LAN and SERVER_LAN.
After that we need to define our policy, we want to permit traffic from the USER_LAN zone to the SERVER_LAN zone, but not from the SERVER_LAN zone to the USER_LAN zone.
we do this by defining a deny policy

this is how the policies are going to look when defined:

```
security {
    policies {
        from-zone USER_LAN to-zone SERVER_LAN {
            policy default-permit {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
        from-zone SERVER_LAN to-zone USER_LAN {
            policy default-deny {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    deny;

                }
            }
        }
    }
```

Figure 2: security policies for assignment 53.

Full config here: https://gitlab.com/emil.privat/networking_emil_simonsen/-/tree/main/Configs/Ass54

[1] Reference to assignment 10

3. Setting up a Raspberry pi virtual machine with NginX

For a guide on how to setup a Raspberry pi Virtual machine look up assignment 1[2]

Once we have set up our raspberry pi virtual machine, we now need to install NginX. It's the software we are going to be using to host our webserver.

First we need to connect out new virtual machine to the internet. We do this by following this diagram:

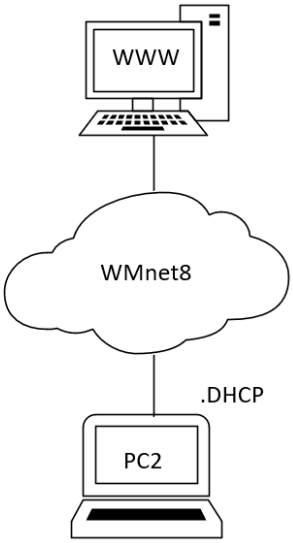

Figure3: Simple internet connection

We are going to set up our PC2 with a simple DHCP network connection via VMnet8 which is going to provide us with internet to install NginX

First go into your network configurations and configure or crate a new network and configure it to DHCP like this:

Figur3 4: Simple DHCP connection.

[2] Reffrance to assignment 1

Now we want to change PC2's network adaptor to VMnet8
You do its by right-clicking on your machine on the sidebar in WMWW and selecting settings > network adaptor check "Custom: Specific virtual network" in the dropdown menu, select VMnet8

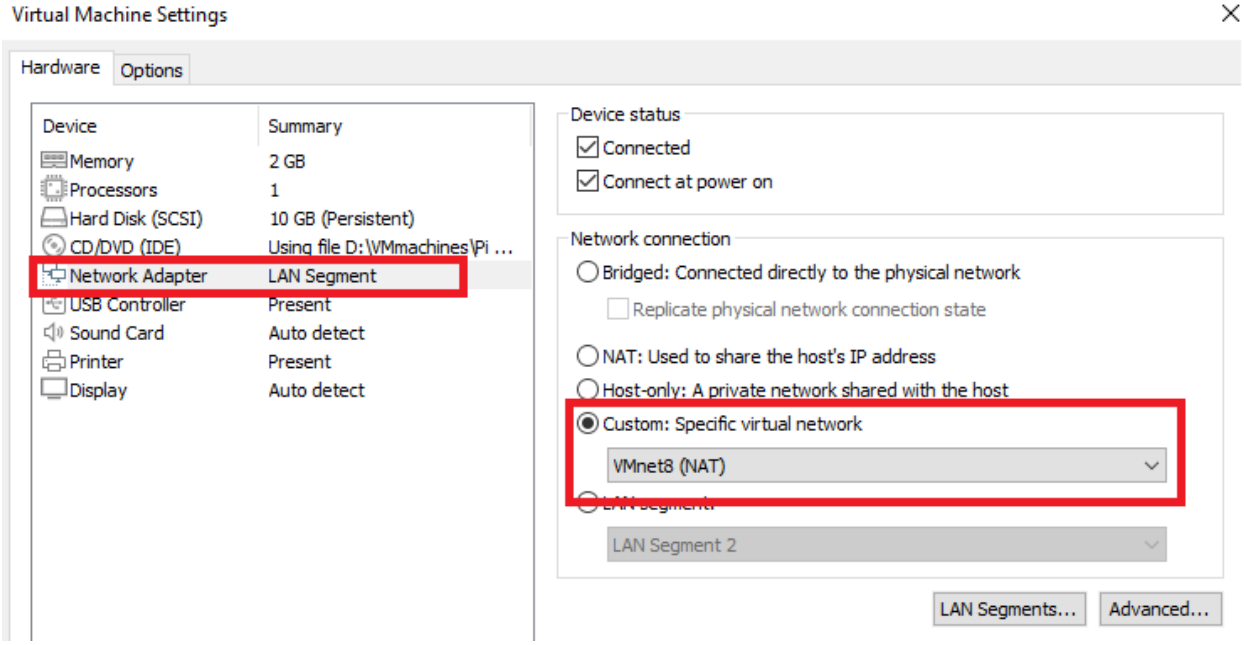

Figure5: selecting VMnet8

We now have internet access

Now we can install NginX and Wireshark with the command: **"sudo apt install nginx" and "sudo apt install wireshark"** in a new terminal.

3.1 Crating the webserver

We can now crate our webserver
Use the following commands in a new terminal:

Mkdir myPython/
Cd myPython/
Mkdir my_test_site
Nano index-html

Now we are in a nano window type the following text from appendix figure 6
Ctrl O to save
Ctrl X to exit.
Now we need to run it via Python3 to do this with the following command:
"python3 -m http.server"
now change back to your static ip network configuration and open your browser on PC2
write 192.168.11.2:8000 to load the wepserver. If it loads and looks like this :

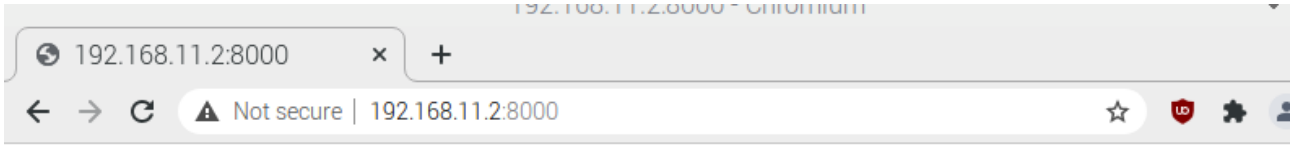

Figure 7: Webserver test.

Then we crated the webserver correctly.

3.2 Making the webserver dynamic

We start by adding some color to the webpage

In our terminal press ctrl C to stop the webserver.
Now type:

Cd myPython/my_test_site/
Nano myStyles.backup (we start by typing backup to have color coding in nano)

Now in nano type in the text from figure 7 in the appendix
Save and type:
Mv myStyles.backup myStyles.css
Ls -all (to show if all the files are correct)

Now we need to link the index and myStyles files with the following commands

Nano index.html

Copy text form appendix figure 8 into nano, this will crate a linktag to myStyles.css
The page is still static but it now has colors 😊

To make the webserver dynamic we need to make a pyhon program that keeps updating it. We are going to make a program that updates the number on the page every second
To do this write the following command in a terminal

Nano my_test_updater.py

Copy the text from appendix figure 9 into nano and save.
The webserver is now dynamic.

To test if it work open the python program and the site in two different terminals

On PC1 open up the site

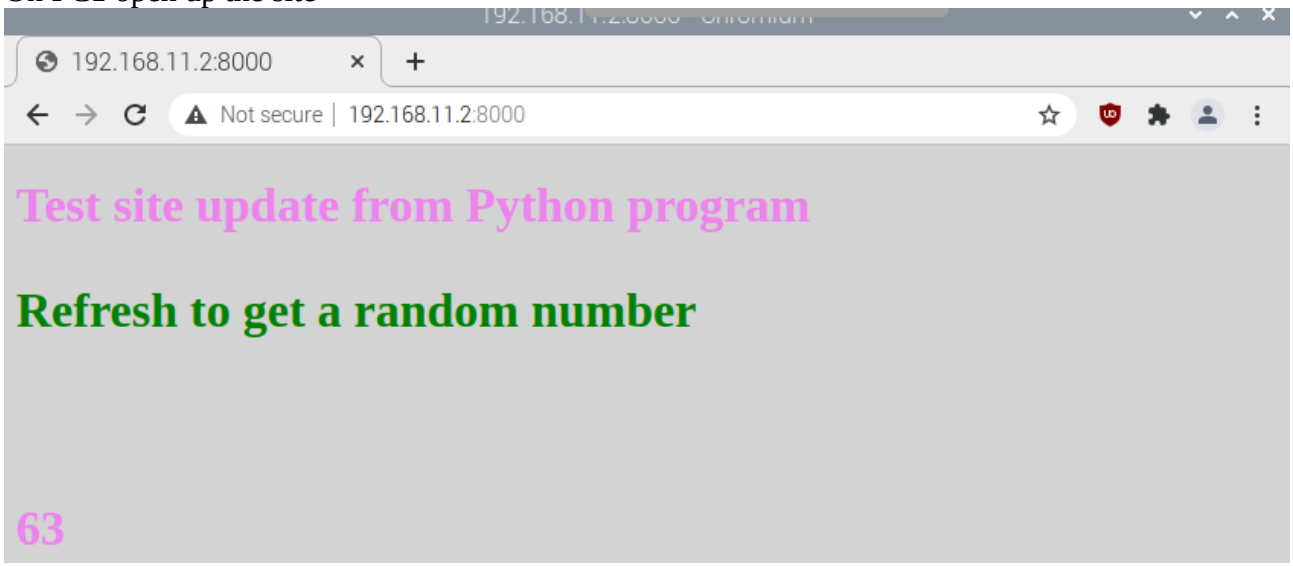

Figure 10: final dynamic webserver.

If the number keeps updating then congratulations. The webserver works.

3.3 Wiresharking

3.3.1 TCP filter

First lets have a look at the TCP filter in Wireshark from pc1 when we refresh the webserver

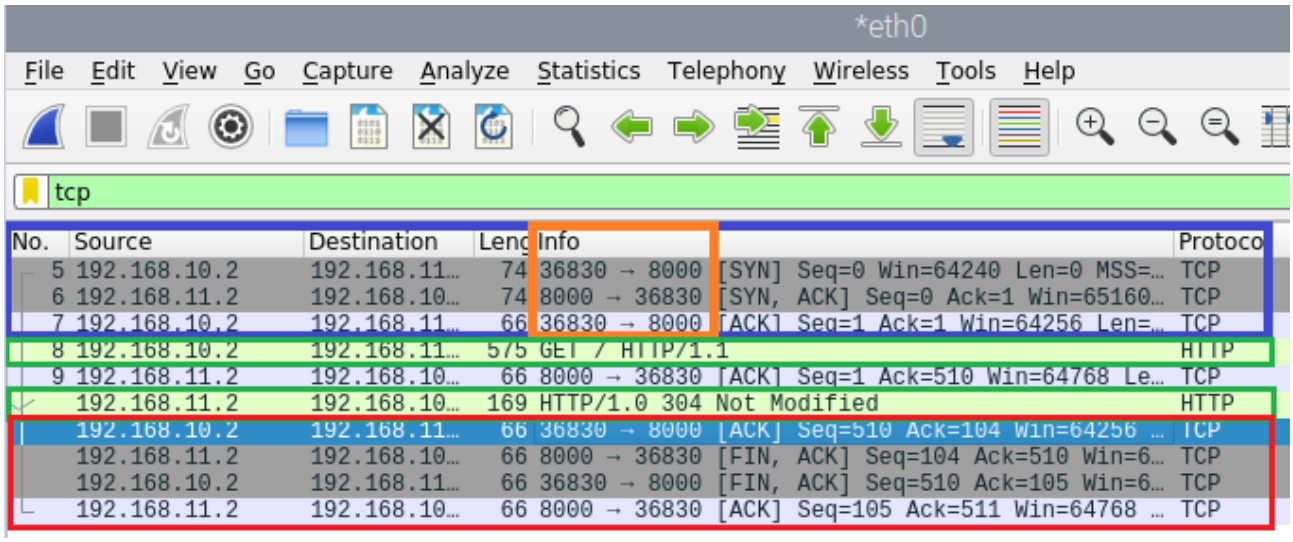

Figure 11: TCP filter highlighted

The first 3 frames we see marked in blue in figure 11 is what is called the 3 way handshake. Its setting up a 2 way connection form PC1 to PC2. Marked in orange we can see the ports in use. Port 36830 on PC1 is connecting to port 8000 on PC2 and port 8000 is looking for incoming requests on 36830. Finaly we see that a connection has been established on the transport layer.

The two green marked frames are http frames, the frames here is the exchange happening on the application layer.

Lastly the 4 frames marked in red is the connection being dropped between the two applications

3.4 HTTP filter

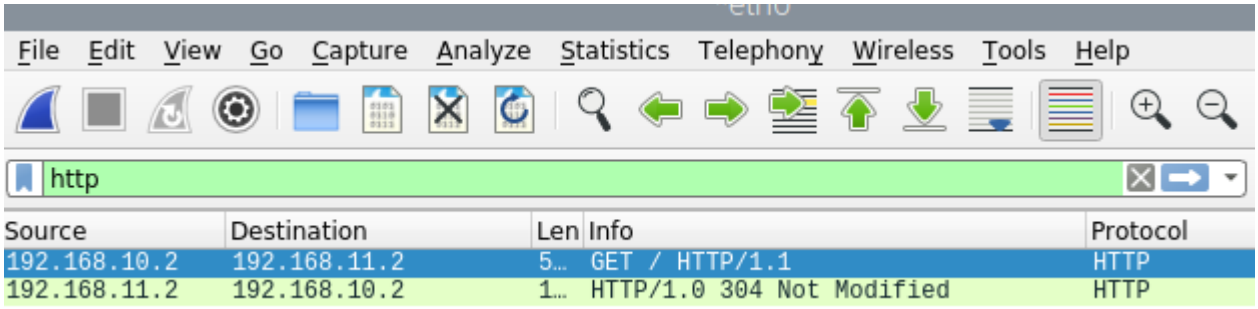

Figure 12: HTTP filter

Here is the same frames as maked in green on figure 11, just with a cleaner filter if we only want to see http frames.

3.5 Source, destination addresses and layers

When we look at the application layer, we are in this assignment talking about HTTP protocol. This is where the get requests are happening. Who is getting the get request and the get response

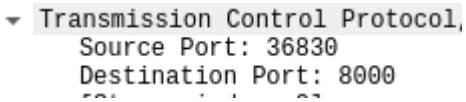

Figure 13: ports

One layer down we have the transport layer, in this assignment its where we are talking about TCP. TCP is determined by the ports in play when sending out a frame.
For instance if two different ports were used during the TCP then a different site would be loaded on the machine if it was assigned the corresponding port.

The next layer, the datalink layer. We have Mac addresses. This is useful when we are trying to identify who is on a network and doing ARP

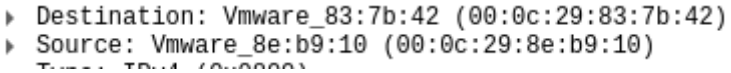

Figure 14: mac addresses

Layer two is the networking layer where IP addresses reside. Again it's a useful tool when we want to identify who is on a network (look at the diagram to see ip's).

Lastly we have the physical layer, this is where we have the physical connection (wires, cables, routers, switches and so on)

4. HTTP explanation

HTTP application protocol is used to fetch files such as HTML documents from one pc to another. Its sent over the transport layer using TCP

App layer	DNS	
Transport layer	UDP	
Ports	S:xxxx	D:yyyy
IPs	S:IP x	D:IP y
MACs	S:MAC x	D: MAC y

App layer	WEB	HTTP
Transport layer	TCP	
ports	S:zzzz	D: QQ
IPs	S:IP x	D:IP y
MACs	S:MAC x	D:MAC y

5. Conclusion

During this assignment the student has learned how to set up a dynamic webserver using NginX on a raspberry pi virtual machine running linux os. The student has also gained a further understanding of the different layers and whats going on at each layer.

As such the assignment has been completed in accordance with the learning goals.

As suc

6. Appendix

6.1 Index source code

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>


<h1>Test site update from Python program<h1>
<p>Refresh to get a random number
<br>
<br>
<br>
<h1>
</h1>
</body>
</html>
```

Figure 6: Index.html source code

6.2 myStyles.css code

```
body {
  background-color: lightgrey;
}
h1 {
  color: violet;
}
p {
  color: green;
}
```

Figure 7: styles for the webpage

6.3 Link-tag Index.html

```
<!DOCTYPE html>
<html>
<head>
</head>
      <link rel="stylesheet" href="myStyles.css">
<body>


<h1>Test site update from Python program<h1>
<p>Refresh to get a random number
<br>
<br>
<br>
<h1>
22
</h1>
</body>
</html>
```

Figure 8: Edited index file with linktag

6.4 Python program my_test_updater.py

```
import random, time

while True:
   random_number = random.randint(1,101)

   text = ('''
      <!DOCTYPE html>
      <html>
      <head>
      </head>
            <link rel="stylesheet" href="myStyles.css">
      <body>
```

```
    <h1>Test site update from Python program<h1>
    <p>Refresh to get a random number
    <br>
    <br>
    <br>
    <h1>
    '''
    + str(random_number) +
    '''
    </h1>
    ''')
myFile = open("index.html", "w")
myFile.write(text)
myFile.close()
print(text)
time.sleep(1)
```

Figure 9: python file my_test_updater.py