SJSU | DEPARTMENT OF COMPUTER ENGINEERING

CMPE-206 Fall 2018

Report on Enterprise Network

Enterprise Network

Under the guidance of

Dr. Bapi Vinnakota

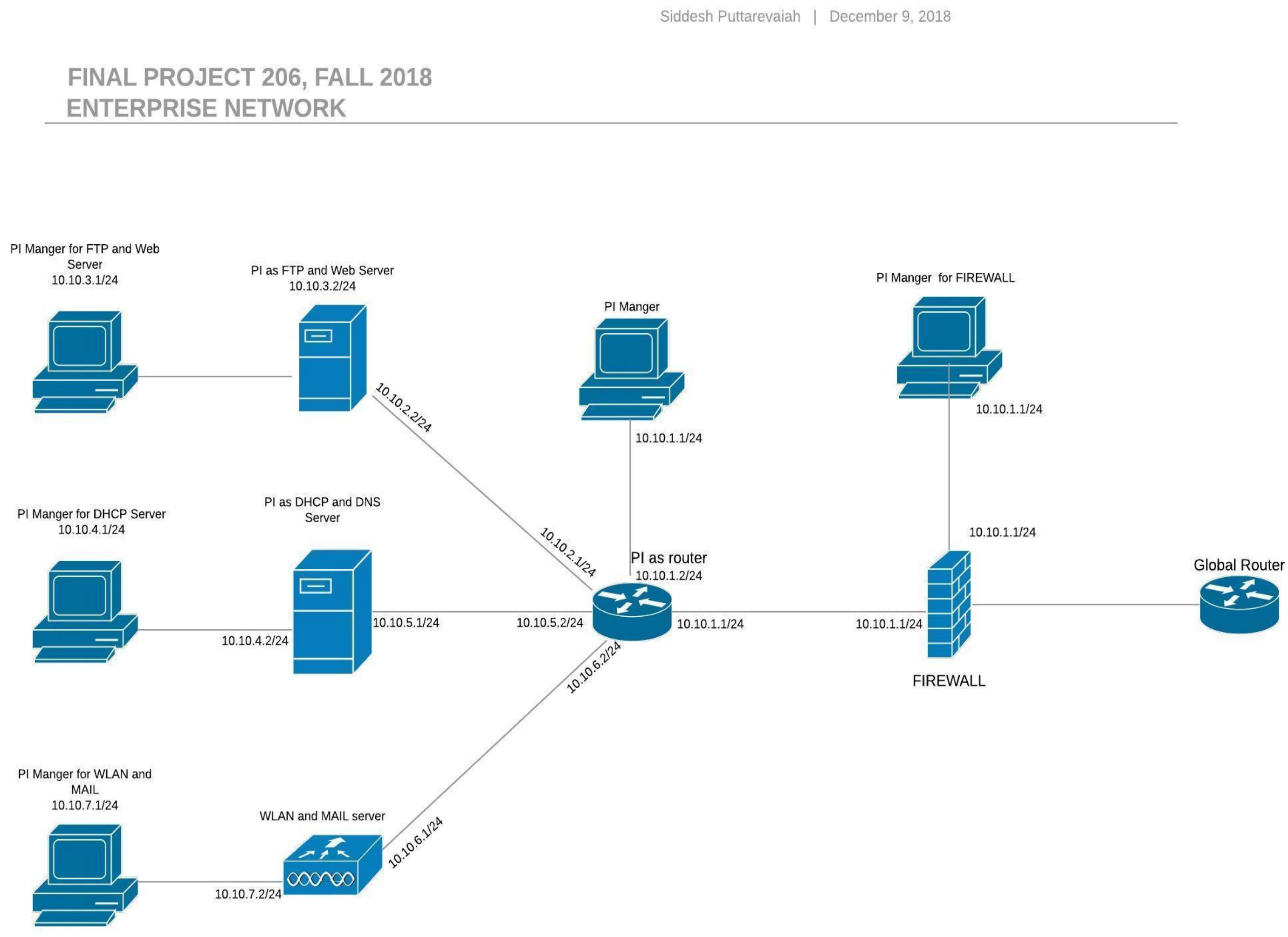
Table of Contents

Configuring the Raspberry PI as a router

Team: Siddesh Puttarevaiah and Nagesh Ramprasad

Overview:
The main purpose of this project is to study and analyse the working of Raspberry PI as Router. We execute many commands related to networking stack TCP/IP in this lab and provide screenshots of output from the commands after execution.

We choose Hybrid Operating Systems to run this Lab Windows and Linux(Ubuntu). Our Raspberry Pi is connected to this Linux machine and we “SSH” to the PI using the Terminal. We execute all the Linux Commands on either one of the SSH terminals hence giving is the Linux command line to our command execution. Other two Windows laptop connect through Putty



Block Diagram of the Configuration Done:

Figure 1: Block Diagram of Enterprise Network

Raspberry PI as Router:

We configure our PI as router using ip address commands and ip route commands.

Step 1:

We do a headless boot into the PI using PI Manger. Below screen shot provides the login performed on the Linux machine.

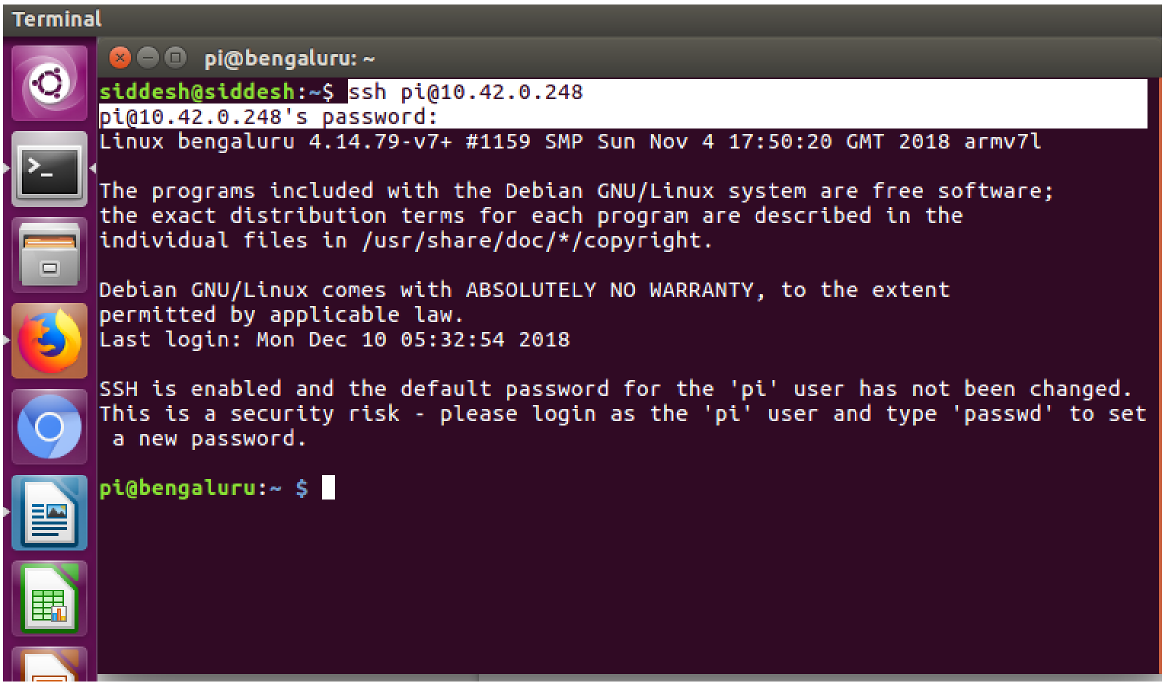


Figure 2: Headless Boot into PI

Step 2:

We get to know the IP address of the interface of the PI. We use command "**ip addr show**", screenshot is mentioned below -

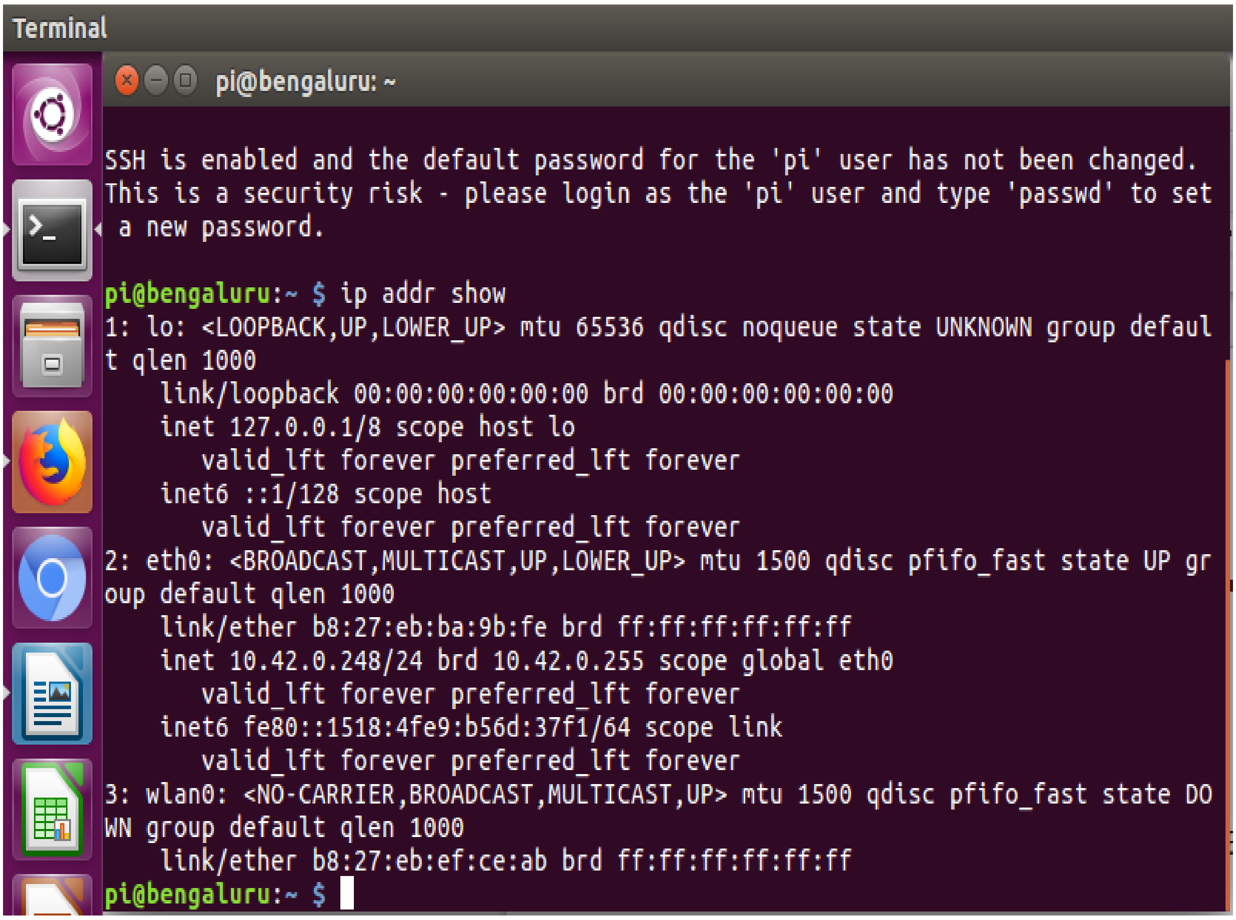


Figure 3: Getting the IP address of the interface in the PI.

Step 3:

Once we get to know the IP address of the PI, we connect the rest of the enterprise network to the PI using USB to Ethernet Dongle. After connecting other 4 network devices to PI, I configure each PI with IP address as mentioned in the Figure 1. In the project we add the IP address to each interface using the command "**ip addr add dev $interface $ip_address/CIDR**". Once we add the IP address to interfaces of the PI we do the same at the other PI interface. Once we are done with the IP address assignment we ping each interface PI address and check the connectivity. Below screenshots shows us the an example of the above procedure for PI as DHCP server -

Step 4:

Once we add the IP addresses to each interface we make router as default interface in each PI so that all the packets unknown toe each PI will be forwarded to router. Router routes the packets accordingly, since most the network is directly connected to router.

Step 5:

We configure our router for dhcp-relay to forward the DHCP request packets. We use isc-dhcp-relay on the router -

install “sudo **apt-get install isc-dhcp-relay**”

Once installing the relay on the PI, we must configure the DHCP server IP address on the router, which is 10.10.5.1/24 in our case. Once configured we must run the dhcp relay service using the commands “**service isc-dhcp-relay start**”, “**service isc-dhcp-relay restart**” and “service isc-dhcp-relay status”. Start command used to start the service and restart to restart the service after changes made to the “conf” file and status to check the status of the dhcp running. Below diagram shows us the status and restart fixing a issues after connecting a new interface -

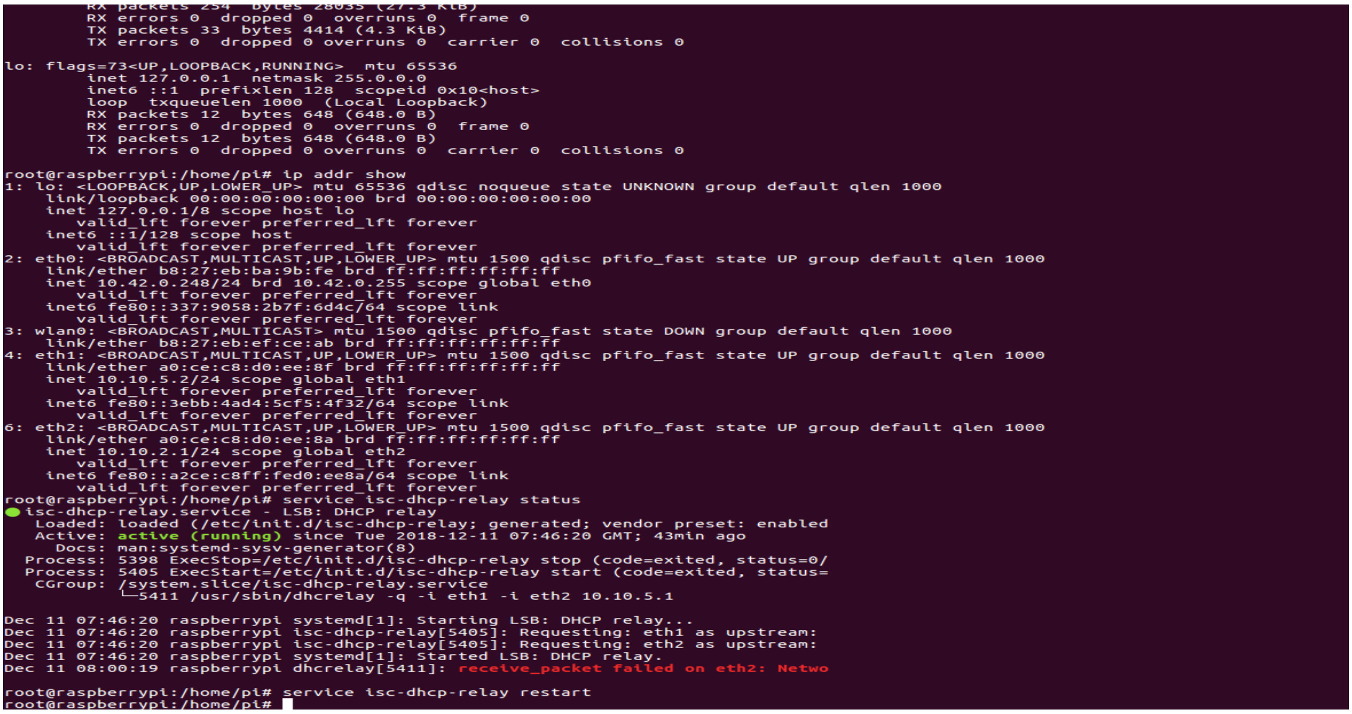


Figure 4: Starting ISC DHCP Relay service

Configuring the Raspberry PI as a DHCP/DNS and integration to router

DHCP/DNS contribution part

A DHCP ethernet server was set up to perform its function as a dynamic host configuration protocol to assign devices connected to the router an IP address. The DNS server was set up as a static DNS ethernet server to perform its function as domain name server, converting entered domain names of devices to corresponding IP addresses, performing a forward lookup; reverse lookup nor dynamic DNS was performed. A significant part of the testing was spent on integrating DHCP correctly with the router and other services connected to the router.

DHCP was set up as a wired DHCP server according to the project's requirement. The server was tested alone, by connecting different devices to it's USB ports using the Ethernet dongles. Once, we confirmed it correctly leased IP addresses according to a devices subnet and specified range for that subnet, there was only one connection from the DHCP/DNS server to the router. We tested with the router, to verify that the router and supported software and settings, correctly routed DHCP requests connected to it to the DHCP/DNS server and these requests were acknowledged and the devices received an IP address in the specified range for its subnet. Lastly, we confirmed that the DNS server functionality was reachable using dig tool and by one of the services connected to the router, namely, the web server.

TESTING DHCP AND INTEGRATION

One Raspberry PI was set up as a DHCP/DNS ethernet-based server. Please see [1] for overall steps followed. The following steps were taken to accomplish this:

1. The PI was updated and the Debian Linux ISCP DHCP server software was installed, using the following command:

   apt-get install isc-dhcp-server

2. The file **/etc/default/isc-dhcp-server** was edited to contain the interfaces, eth1, eth2, and eth3 on which DHCP requests where to be handled for the router and other services, respectively:

   Eth1: router's subnet
   Eth2: 10.10.6.0 # mail server
   Eth3: 10.10.2.0 # web server

```
  GNU nano 2.7.4                            File: /etc/default/isc-dhcp-server

# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="eth1 eth2 eth3"
#INTERFACESv6=""
```

Figure 1: Main isc-dhcp-server file

3. The file **/etc/network/interfaces** was configured to contain the following, which states eth1 as the DHCP server main interface with a **static IP address** and gateway to the router's IP address.

```
  GNU nano 2.7.4                                   File: /etc/network/interfaces

 interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

auto eth1
iface eth1 inet static
        address 10.10.5.1
        netmask 255.255.255.0
        broadcast 10.10.5.255
        gateway 10.10.5.2
```

Figure 2: Interfaces file

4. For testing the DHCP part, we left our default DNS, in the /etc/resolv.conf file:

```
  GNU nano 2.7.4                  File: /etc/resolv.conf

# Generated by resolvconf
nameserver 192.168.2.1
```

Figure 3: Resolv configuration file

5. We edited the main DHCP configuration file **/etc/dhcp/dhcpd.conf** to contain specific details and the subnets for each interface, each with it's own subnet and range, and set the DHCP server to be **authoritative**, meaning this should be the only DHCP server on the network:

```
GNU nano 2.7.4                                          File: /etc/dhcp/dhcpd.conf

# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# option definitions common to all supported networks...
option domain-name "taghreedsamiradhcpdnsserver";
option domain-name-servers 192.168.2.1;

default-lease-time 600;
max-lease-time 7200;

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style none;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;
```

Figure 4: Main dhcp configuration gile

```
GNU nano 2.7.4                                          File: /etc/dhcp/dhcpd.conf

#    range 10.0.29.10 10.0.29.230;
#  }
#}


#eth1
subnet 10.10.5.0 netmask 255.255.255.0 {

}

#eth2
subnet 10.10.6.0 netmask 255.255.255.0 {
        range 10.10.6.10 10.10.6.20;
        option subnet-mask 255.255.255.0;
        option broadcast-address 10.10.6.255;
        option routers 10.10.6.2;
        option domain-name-servers 192.168.2.1;
}

#eth3
subnet 10.10.2.0 netmask 255.255.255.0 {
        range 10.10.2.10 10.10.2.20;
        option subnet-mask 255.255.255.0;
        option broadcast-address 10.10.2.255;
        option routers 10.10.2.2;
        option domain-name-servers 192.168.2.1;
}


authoritative;
```

Figure 5: Continuation of main dhcp configuration file, listing subnets

6. On the server, for interface eth1, we assigned IP address: 10.10.5.1

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 10.10.5.1  netmask 255.255.255.0  broadcast 0.0.0.0
```

7. We added routes for the other subnets to which router connects to:

```
ip route add 10.10.6.0/24 dev eth1
ip route add 10.10.2.0/24 dev eth1
```

8. Once this is complete, we restart the service to ensure all changes take effect:

```
/etc/init.d/isc-dhcp-server restart
```

9. We checked the **/var/log/syslog** file to verify devices had obtained an IP address and we also confirmed on the devices that the IP address showed up on its respective interface:

   In the image below, in the blue rectangle, we can see the sequence:

   DHCPDISCOVER
   DHCPOFFER
   DHCPREQUEST
   DHCPACK

   Device with hostname 'kali' takes IP address 10.10.6.30 in the pool of specified range.

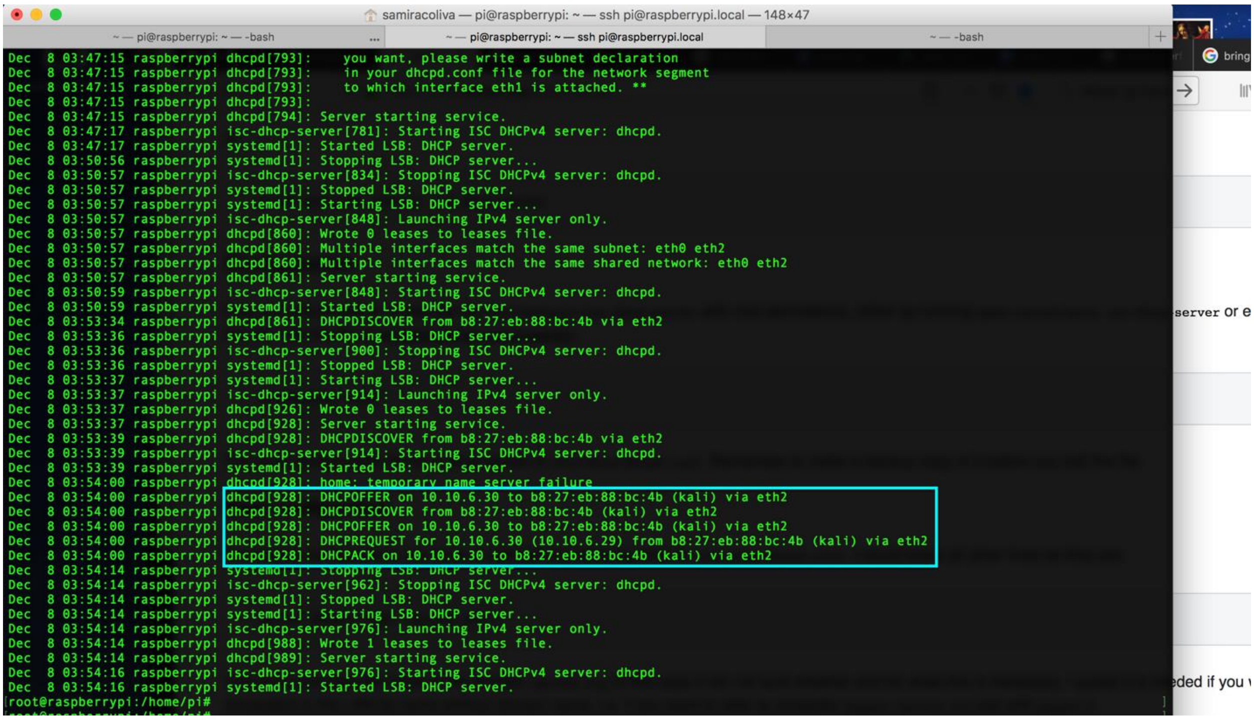


Figure 6: DHCP leases

INTEGRATION: we connect only eth1 to router and other devices on other subnets are connected to the router. We checked syslog again to verify that DHCP was leasing and confirmed the lease on the device's corresponding interface.

Figure 7: DHCP ACK on kali Raspberry PI

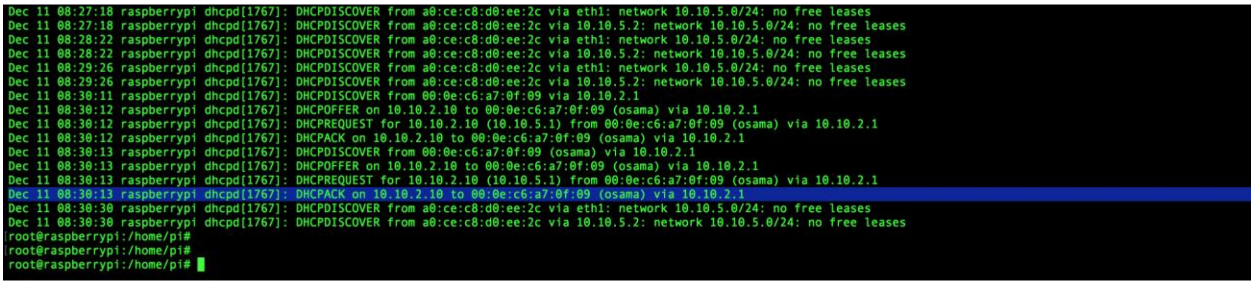


Figure 8: DHCP ACK on the FTP and Web servers

TESTING DNS and INTEGRATION

We used the bind9 Debian Linux DNS software package and referred to [2] and [3] for configurations.

1. We edited **/etc/resolv.conf** to contain the local host IP address and commented out all others, since the server would now become also a DNS server.

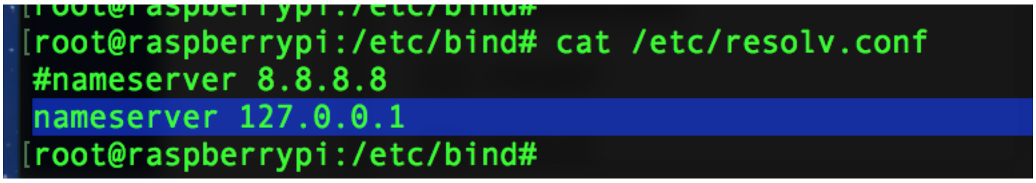


Figure 9: Update of resolv file to local host

2. We installed **bind9**, Debian linux DNS server software and **dnsutils**:

apt-get install bind9 dnsutils

3. We edited two files inside **/etc/bind**/. In the first, ***named.conf.local*** we specify the forwarding zone and file name for said. Here "Gesu.local" is the name we wish to give our DNS forwarding zone. A zone is just an area where you make definitions for performing forward lookups. Similarly, if we were to do, reverse lookups, we would specify that here in another zone.

nano named.conf.local

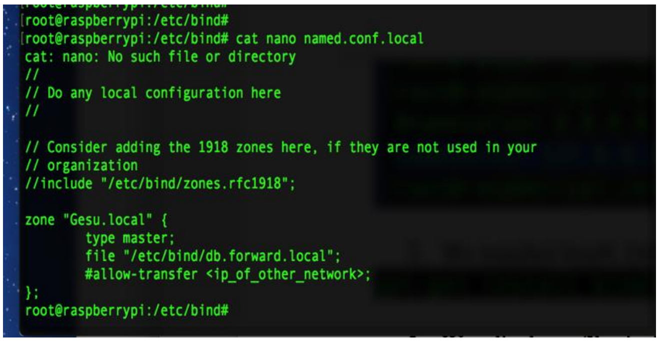
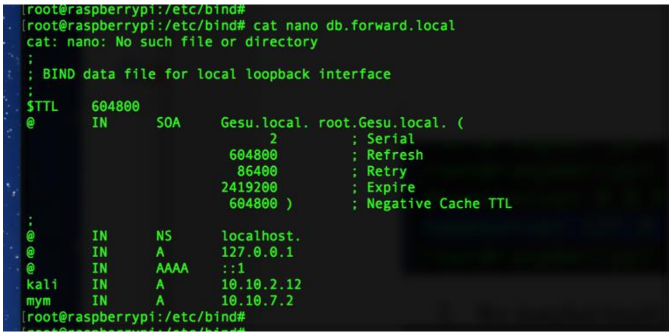


Figure 10: DNS zone declaration

In the forwarding zone file, we specify the static entries for host. Note: we must first assign such addresses statically and then enter them here, since this is not dynamic DNS. The file was created copying the existing file in the bind9 package and renaming to ***db.forward.local***. The only mofications on our end are the last two entries. Kali, a PI running kali linux, the firewall, and the web server (osama's webserver):

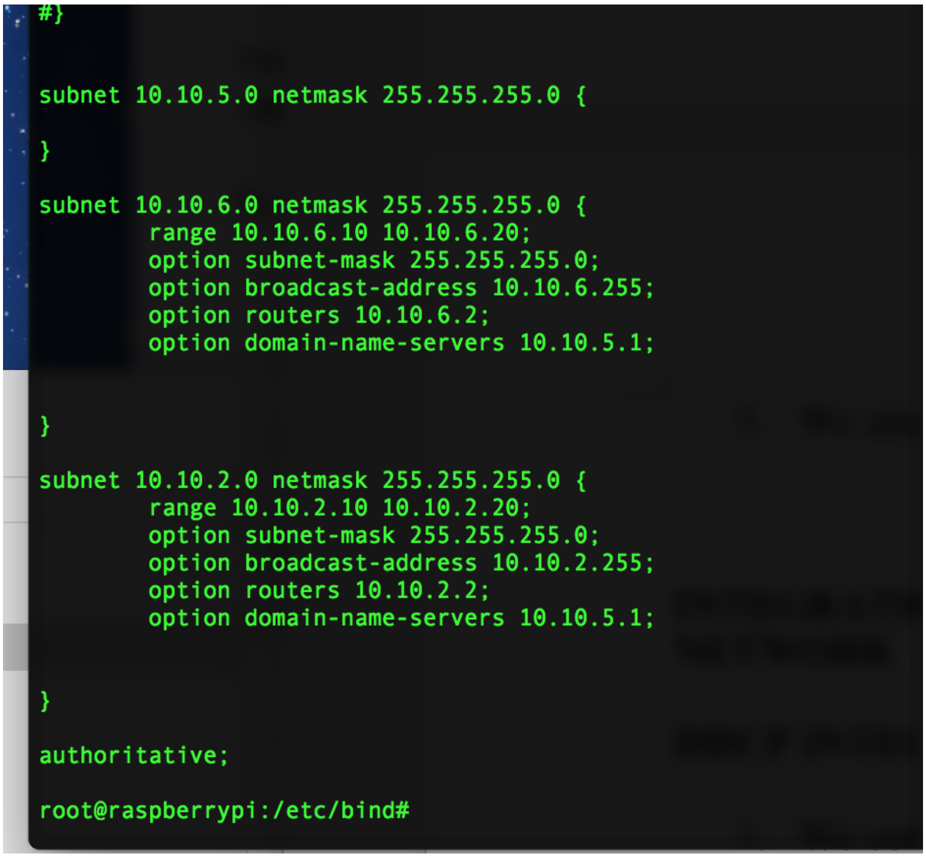
nano db.forward.local



```
root@raspberrypi:/etc/bind#
root@raspberrypi:/etc/bind# cat nano db.forward.local
cat: nano: No such file or directory
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     Gesu.local. root.Gesu.local. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
@       IN      A       127.0.0.1
@       IN      AAAA    ::1
kali    IN      A       10.10.2.12
mym     IN      A       10.10.7.2
root@raspberrypi:/etc/bind#
```

Figure 11: DNS hosts file

4. We also had to make changes inside the ***dhcpd.conf*** file and update the DNS entry in each subnet to the PIs (DNS /DHCP) server IP address.



```
#}

subnet 10.10.5.0 netmask 255.255.255.0 {

}

subnet 10.10.6.0 netmask 255.255.255.0 {
        range 10.10.6.10 10.10.6.20;
        option subnet-mask 255.255.255.0;
        option broadcast-address 10.10.6.255;
        option routers 10.10.6.2;
        option domain-name-servers 10.10.5.1;


}

subnet 10.10.2.0 netmask 255.255.255.0 {
        range 10.10.2.10 10.10.2.20;
        option subnet-mask 255.255.255.0;
        option broadcast-address 10.10.2.255;
        option routers 10.10.2.2;
        option domain-name-servers 10.10.5.1;


}

authoritative;

root@raspberrypi:/etc/bind#
```

Figure 12: Updating the DNS field of each subnet

5. We made the router the default gateway on the DHCP/DNS server and added routes to the other subnets.

6. Because we are using DHCP and static DNS, we tested DNS by adding a node's static IP in the forward zone file, and test that the resolution works correctly. Here we can see that for the firewall using **dig (domain information groper)**, a DNS lookup utility tool, which does a DNS lookup and displays results and verifying with **ping**.

dig mym.Gesu.local (firewall)
ping mym.Gesu.local

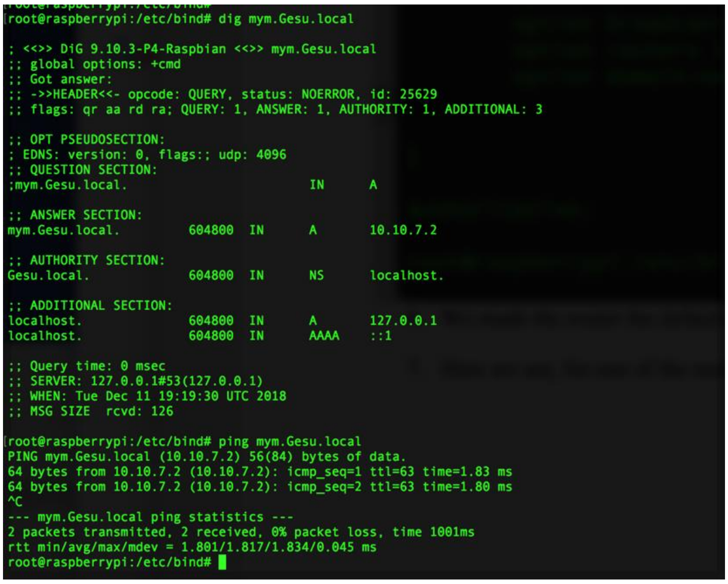


Figure 13: Using groper to test DNS server

The FTP server was also able to resolve the name we gave it.

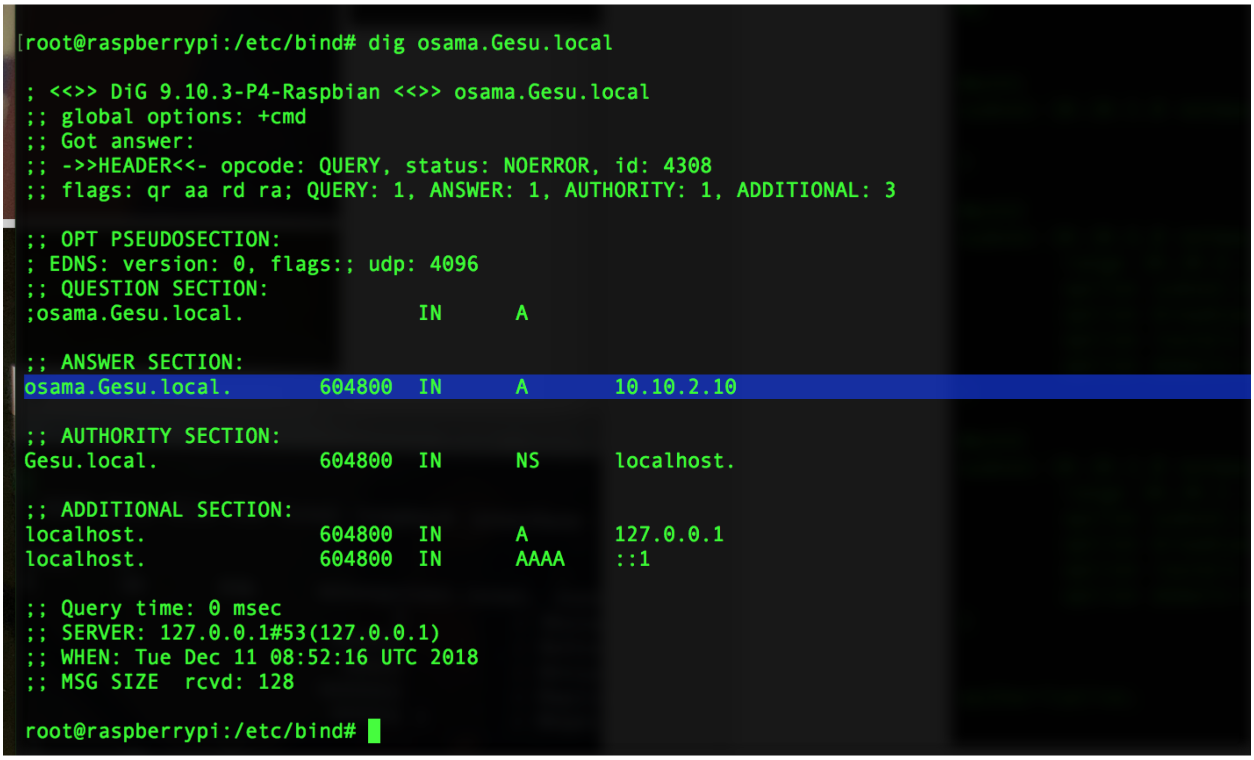


Figure 14: testing DNS on FTP / Web server

References:
[1] https://wiki.debian.org/DHCP_Server

[2] https://wiki.debian.org/Bind9

[3]https://www.ionos.com/digitalguide/server/configuration/how-to-make-your-raspberry-pi-into-a-dns-server

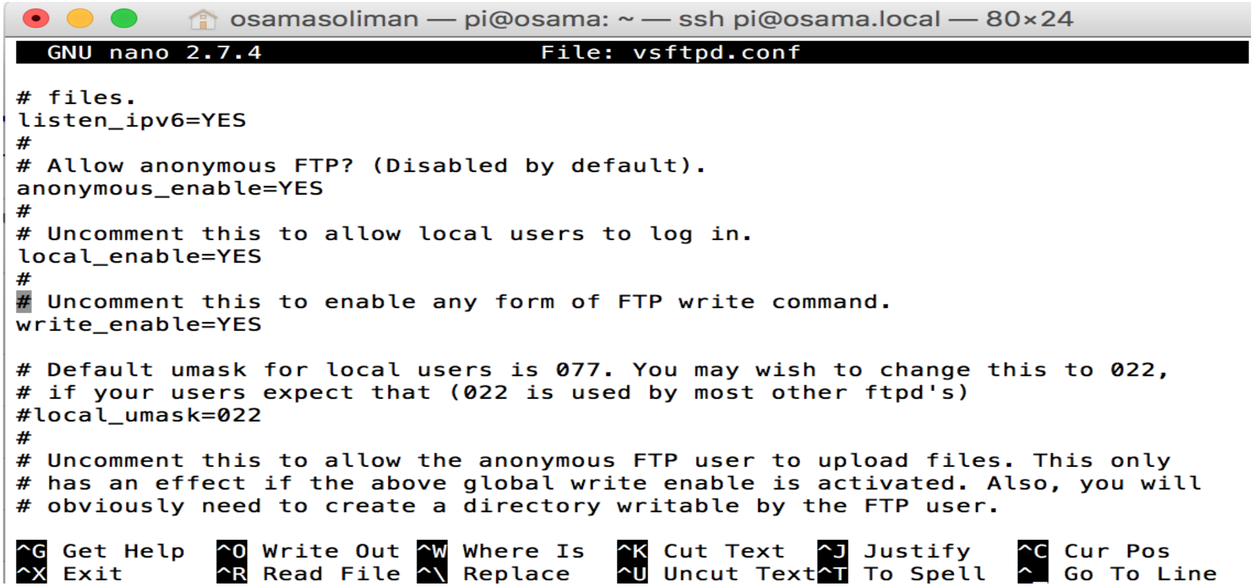
Difficulties:
The PIs turned out to be quite a nightmare to work with. Often times without changes on the interfaces we had to flash the cards and redo all work just to access them.

Configuring a Raspberry PI as a mail server and integration to router

Configuring a Raspberry PI as FTP & web server and integration to router

Installing and configuring FTP

We are using vsftpd as our FTP. Vsftpd was downloaded and installed. FTPs has two modes passive and active modes. In the active mode connections are initiated by the ftp server and the client just waits for server requests. In passive mode ftp server waits for client requests to initiate. In this project passive FTP was used.



```
osamasoliman — pi@osama: ~ — ssh pi@osama.local — 80×24
  GNU nano 2.7.4                  File: vsftpd.conf

# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES

# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Figure 1: ftp configuration file enable anonymous user

In the Figure 1 above, it shows the enabling of both the anonymous user enabling and the local users to use the FTP services. The Anonymous user can upload but can’t download from the FTP. The Pi user can upload and download from it as he is the owner of the file. I use the home directory of the Pi as the ftp container.

```
osamasoliman — pi@osama: ~ — ssh pi@osama.local — 80×24
  GNU nano 2.7.4                    File: vsftpd.conf

#
# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.
anon_mkdir_write_enable=YES
anon_root=/home/pi
#
# Activate directory messages - messages given to remote users when they
# go into a certain directory.
dirmessage_enable=YES
#
# If enabled, vsftpd will display directory listings with the time
# in  your  local  time  zone.  The default is to display GMT. The
# times returned by the MDTM FTP command are also affected by this
# option.
use_localtime=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Figure 2: setting anonymous directory to home of pi

```
osamasoliman — pi@osama: ~ — ssh pi@osama.local — 80×24
  GNU nano 2.7.4                    File: vsftpd.conf

# If enabled, vsftpd will display directory listings with the time
# in  your  local  time  zone.  The default is to display GMT. The
# times returned by the MDTM FTP command are also affected by this
# option.
use_localtime=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES
#
# If you want, you can arrange for uploaded anonymous files to be owned by
# a different user. Note! Using "root" for uploaded files is not
# recommended!
chown_uploads=YES
#chown_username=whoever
#
# You may override where the log file goes if you like. The default is shown

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Figure 3: enabling upload

Port 20 is used for connection with FTP server and only local hosts can download to their home directories.

Tested upload and download. The figure below shows a download successful attempt.

```
osamasoliman — ftp osama.local — 80×24
[Osama-2:~ osamasoliman$ ftp osama.local
Trying fe80::9064:1ba9:5187:510f%en5...
Connected to osama.local.
220 (vsFTPd 3.0.3)
Name (osama.local:osamasoliman): pi
331 Please specify the password.
[Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
[ftp> ls
229 Entering Extended Passive Mode (|||34724|)
150 Here comes the directory listing.
-rwxr-xr-x    1 0        114           477 Nov 12 05:58 osama.cpp
-rw-------    1 1000     1000            0 Nov 12 07:38 osama.hi
-rwxr-xr-x    1 1000     1000         1643 Nov 12 07:37 randombucket.h
226 Directory send OK.
[ftp> get osama.hi
local: osama.hi remote: osama.hi
229 Entering Extended Passive Mode (|||27911|)
150 Opening BINARY mode data connection for osama.hi (0 bytes).
     0        0.00 KiB/s
226 Transfer complete.
ftp>
```

Figure 4: ftp download

Installing and configuring web server

Apache web server was downloaded and configured to be used as the webserver for this project. On it two web pages a www.internal.com and a www.external.com .

Internal configuration file:

```
osamasoliman — pi@osama: ~ — ssh pi@osama.local — 80×24
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port th
at
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ServerName internal.com
        ServerAlias www.example.com



        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn
```

Figure 5: internal web server

External configuration file:

```
osamasoliman — pi@osama: ~ — ssh pi@osama.local — 80×24
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port th
at
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/external.com
        ServerName external.com
        ServerAlias www.external.com
        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Figure 6: external web server

On the apache2.conf configuration file in the /etc/apache2 folder using the require command you can make only certain people access the web server instance you want.

```
<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/html>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require ip 169.254.224.243
</Directory>
<Directory /var/www/external.com>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

Figure 7: allowing only ip 169.254.224.243 to access internal

If it is opened for everyone to access anyone can access both webpages.

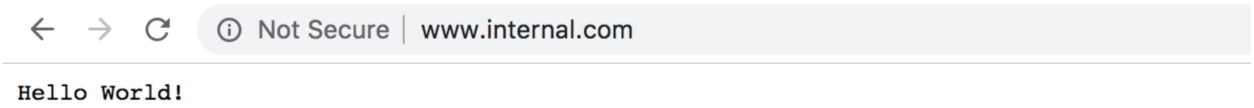


Figure 8: internal

Figure 9: external web server

After requiring a certain ip address to enter internal this is what ips from out of the subnet get if they want to access the internal web page.

Figure 10: an ip not in the list trying to access internal.com