



**Instituto Politécnico Nacional  
Escuela Superior de Cómputo**

**Práctica 3  
Administración de configuración**

**Integrantes del equipo:**

**Castro Flores Marcela  
Sánchez Cruz Rosa María  
Santiago Mancera Arturo Samuel**

M. en C. Tanibet Pérez de los Santos Mondragón

México, Ciudad de México a 05 de diciembre de 2018

<b>1. Introducción</b>	<b>4</b>
<b>2. Sensores implementados</b>	<b>5</b>
2.1. Supervisión de servidor de correo electrónico (SMTP)	5
2.1.1. Instalación	5
2.1.2. Sensor SMTP	9
2.1.2.1. Funcionamiento	10
2.2. Supervisión de servidor web (HTTP)	11
2.3. Supervisión de servidor de archivos (FTP/FTP Server File Count)	13
2.3.1. Sensor FTP	13
2.3.2. Sensor FTP Server File Count	16
2.4. Supervisión de servidor de impresión (SNMP)	16
2.5. Supervisión de servidor de servidor de acceso remoto (SSH)	16
2.5.1. Instalación	16
2.5.2. Sensor SSH	17
2.5.2.1. Funcionamiento	17
2.6. Administración de archivos de configuración	19
2.6.0.1. Importación	19
2.6.0.2. Exportación	20
<b>3. Conclusiones</b>	<b>21</b>

---

## Índice de figuras

---

2.1. Archivo de configuración /etc/host.conf. . . . .	5
2.2. Archivo de resolución de nombres de dominio. . . . .	6
2.3. Configuración final de Postfix. . . . .	6
2.4. Login de configuración de Webmail. . . . .	7
2.5. Configuración de dominio. . . . .	8
2.6. Bandeja de entrada. . . . .	9
2.7. Código en Python del sensor de correo. . . . .	10
2.8. Funcionamiento del sensor de correo. . . . .	11
2.9. Comando de instalación HTTP apache2. . . . .	11
2.10. Verificación de status servidor HTTP. . . . .	12
2.11. Visualización de ip. . . . .	12
2.12. Página de Apache2 en el navegador. . . . .	13
2.13. Comando de instalación FTP. . . . .	13
2.14. Archivo de configuración FTP. . . . .	14
2.15. Reinicio de servicio y status FTP. . . . .	15
2.16. Código del contador de archivos en el servidor FTP. . . . .	16
2.17. Contador de archivos en el servidor FTP. . . . .	16
2.18. Código del sensor SSH. . . . .	17
2.19. Funcionamiento del sensor SSH. . . . .	18
2.20. Conexiones SSH. . . . .	18
2.21. Archivo de configuracion para importaciones. . . . .	19
2.22. Código para importar archivos. . . . .	19
2.23. Funcionamiento de la importación de archivos. . . . .	19
2.24. Archivo de configuracion para exportaciones. . . . .	20
2.25. Código para exportar archivos. . . . .	20
2.26. Funcionamiento de la exportación de archivos. . . . .	20

# CAPÍTULO 1

---

## Introducción

---

Para la realización de la última práctica se implementaron diferentes servidores con los cuales por medio de la ejecución de diferentes sensores se obtenían tanto sus diferentes tiempos de ejecución como las diferentes respuestas que el servidor desplegaba según era el caso.

Los servidores implementados son los siguientes:

- Servidor de correo electrónico (SMTP)
- Servidor web (HTTP)
- Servidor de archivos (FTP/FTP Server File Count)
- Servidor de impresión (SNMP)
- Servidor de acceso remoto (SSH)

Esta práctica se dividió en las tres partes siguientes:

1. En la primera parte se requirió realizar la instalación y la configuración de los distintos servidores mencionados anteriormente mismos que fueron separados en diferentes máquinas virtuales con el fin de que en el paso posterior se pudiera dividir la topología dada.
2. La segunda parte fue la configuración de la topología que iba desde realizar la conexión correcta entre los dispositivos como también realizar las diferentes comprobaciones para verificar que si habia una conexión entre todos los dispositivos.
3. Por último, la tercera parte correspondió a la utilización de los diferentes sensores que verificaban la correcta respuesta y funcionamiento de cada servidor y de igual manera, se utilizó el administrador de archivos de configuración con el cual se tenia tanto la posibilidad de descargar como de subir los archivos correspondientes a cada uno de los routers disponibles.

En el capítulo mostrado a continuación se observa el desarrollo de la práctica indicando tanto el código utilizado para la implementación de los sensores de cada servidor, como las pantallas que muestran paso a paso el proceso que se realizó para la configuración de la topología dada.

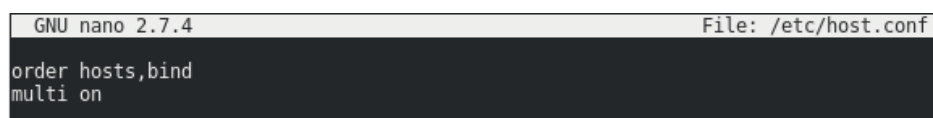
## 2.1. Supervisión de servidor de correo electrónico (SMTP)

### 2.1.1. Instalación

Para llevar a cabo la instalación de un servidor de correo se deben tomar en cuenta 3 elementos: el servidor SMTP (*Simple Mail Transfer Protocol*) encargado de encaminar los correos electrónicos, el servidor IMAP o POP3 encargado de administrar los correos electrónicos recibidos y un cliente de correo encargado de administrar el correo electrónico de las cuentas de usuario.

Para esta instalación se utilizó Postfix como servidor SMTP, Dovecot como servidor IMAP y Rainloop Webmail como cliente de correo electrónico a través de HTTP.

El primer paso es la preparación de los servicios de DNS del servidor. Sin embargo, para esta práctica no se implementó un servidor DNS. Por lo tanto, la resolución de nombres de dominio se realizó localmente en el servidor modificando el archivo `/etc/host.conf` como se muestra en la figura 2.1.



```
GNU nano 2.7.4 File: /etc/host.conf
order hosts,bind
multi on
```

Figura 2.1: Archivo de configuración `/etc/host.conf`.

Así mismo, se cambió el hostname del servidor a **server1** en el archivo `/etc/hostname` se agregó una resolución de nombre de dominio al archivo `/etc/hosts` para indicar que el servidor local es el host y dominio `server1.redes3.local`:

```
root@server1:/home/samuel# cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    ss

# The following lines are desirable for IPv6 capable hosts
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
10.100.70.56 redes3.local server1.redes3.local
```

Figura 2.2: Archivo de resolución de nombres de dominio.

Este último paso se debe realizar en cada cliente de la topología que utilice los servicios de correo con la finalidad de que pueda reconocer la dirección IP del dominio.

Una vez configurado la resolución de nombre de dominio y el nombre de host se procede a instalar **Postfix**. Para ello se utiliza el comando:

```
apt-get install postfix
```

Durante la instalación se solicitará ingresar el tipo de configuración general de correo (*General type of mail configuration*). Para lo cual se eligió **Internet Site**. De igual forma se solicitará el nombre de sistema de correo en el cual se establece el nombre de host más el nombre de dominio configurado anteriormente: **server1.redes3.local**.

El siguiente paso es editar la configuración de Postfix desde su archivo **/etc/postfix/main.cf**. Esta configuración se puede ver reflejada con el comando:

```
postconf -n
```

La configuración final se puede observar en la figura 2.3

```
root@server1:/home/samuel# postconf -n
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases
append_dot_mydomain = no
biff = no
compatibility_level = 2
home_mailbox = Maildir/
inet_interfaces = all
inet_protocols = all
mailbox_size_limit = 0
mydestination = $myhostname, $mydomain, localhost.$mydomain, , localhost
mydomain = redes3.local
myhostname = server1.redes3.local
mynetworks = 127.0.0.0/8 10.0.2.0/24 [::ffff:127.0.0.0]/104 [::1]/128
myorigin = $mydomain
readme_directory = no
recipient_delimiter = +
relayhost =
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtpd_banner = $myhostname ESMTP
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtpd_use_tls = yes
root@server1:/home/samuel#
```

Figura 2.3: Configuración final de Postfix.

El paso siguiente es instalar Dovecot como servidor IMAP. Para ello se utiliza el siguiente comando:

```
apt install dovecot-core dovecot-imapd
```

Una vez instalado deberemos editar algunos de los archivos de configuración. El primero es el archivo **/etc/dovecot/dovecot.conf** en el cual se debe descomentar la línea:

```
listen = *, ::
```

El segundo archivo a editar es **/etc/dovecot/conf.d/10-auth.conf** en el cual se deben tener líneas como las siguientes:

```
disable_plaintext_auth = no
```

`auth_mechanisms = plain login`

El tercer archivo a editar es `/etc/dovecot/conf.d/10-mail.conf` en la siguiente línea:

`mail_location = maildir: /Maildir`

Y el último archivo a editar es `/etc/dovecot/conf.d/10-master.conf` en el bloque de `smtp-auth`:

```
# Postfix smtp-auth
unix_listener /var/spool/postfix/private/auth {
mode = 0666
user = postfix
group = postfix
}
```

Reiniciamos Postfix y Dovecot:

```
service postfix restart
service dovecot restart
```

El último paso es la instalación de Rainloop Webmail como cliente de correo electrónico a través de HTTP. Para ello se instala un servidor apache y las dependencias necesarias para utilizar PHP. Para ello, usamos el siguiente comando:

```
apt install apache2 php7.0 libapache2-mod-php7.0 php7.0-curl php7.0-xml
```

Eliminamos el archivo índice del Apache y descargamos Webmail:

```
# cd /var/www/html/
```

```
# rm index.html
```

```
# curl -sL https://repository.rainloop.net/installer.php | php
```

Finalizada la instalación procedemos a ingresar al navegador web en la dirección `http://localhost/?admin` e ingresamos con las credenciales: **User: admin** y **Password: 12345** como se muestra en la figura 2.4.

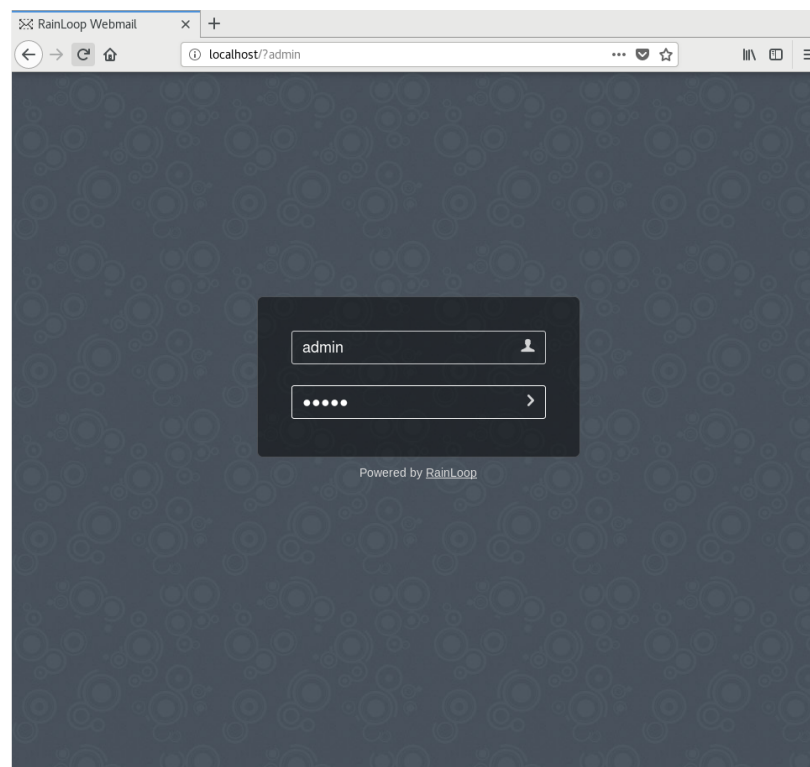


Figura 2.4: Login de configuración de Webmail.

Dentro del panel de configuración nos dirigimos a la pestaña **Domains** en donde se agrega el dominio configurado al inicio. Esto se puede ver en la figura 2.5.

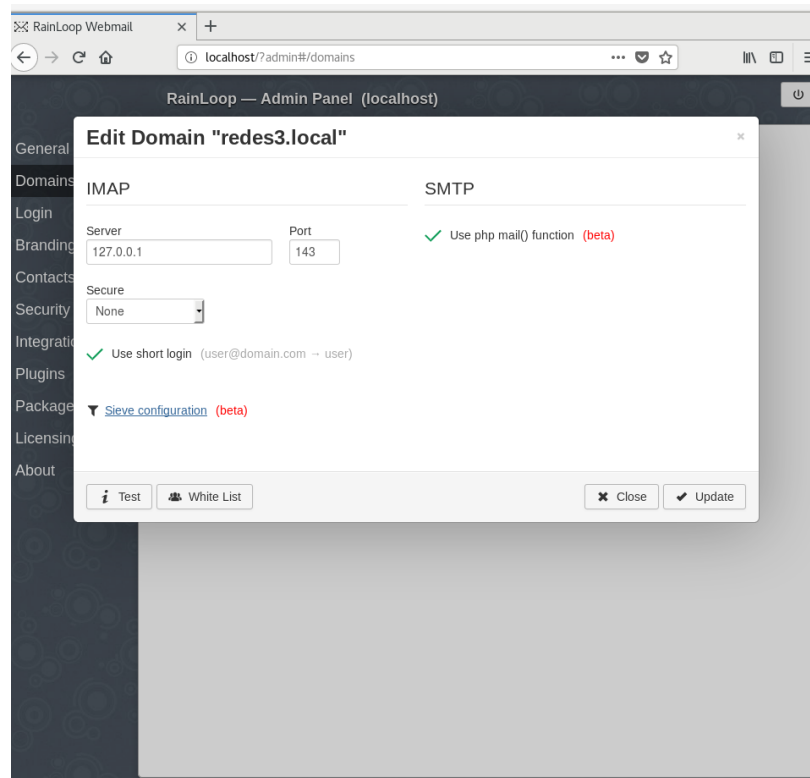


Figura 2.5: Configuración de dominio.

Finalmente, para crear un nuevo usuario en el servidor de correo (SMTP + IMAP) es necesario crear al usuario UNIX. Sin embargo, antes se debe definir una variable global que indique el directorio de almacenamiento de los correos para cada usuario. Para realizar esto se utilizan los siguientes comandos:

```
echo 'export MAIL=$HOME/Maildir' >>/etc/profile
useradd -m samuel
passwd samuel
```



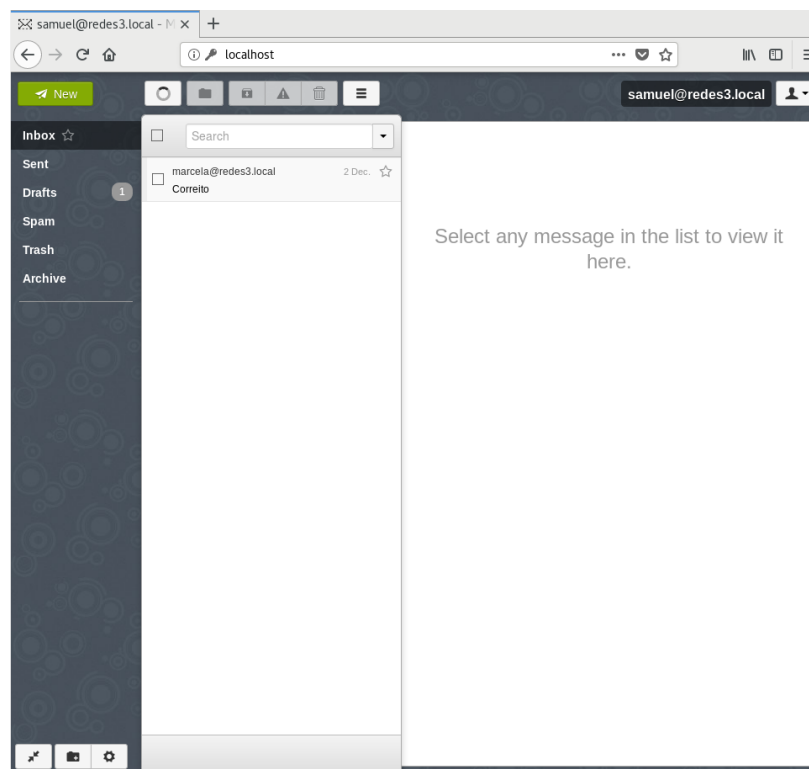


Figura 2.6: Bandeja de entrada.

### 2.1.2. Sensor SMTP

El sensor SMTP realiza mediciones de tiempo de respuesta tanto del servidor SMTP como del servidor IMAP. Para ello, se toma el tiempo (EPOCH) antes de realizar una solicitud SMTP, es decir, tiempo inicial. Después, se envía un correo de prueba a un usuario predefinido. Una vez que se terminó de ejecutar la función de solicitud de envío se vuelve a tomar el tiempo. Este tiempo será el tiempo de respuesta del servidor SMTP, puesto que el servidor SMTP ya terminó su tarea de enviar el correo al servidor IMAP. No obstante, para obtener el tiempo de respuesta del servidor IMAP se obtiene el nombre del último archivo modificado en la bandeja de entrada del usuario de prueba. Por defecto, Dovecot almacena los correos en el sistema de archivos concatenando en el nombre el tiempo (EPOCH) en el que fue recibido. Es por ello, que este valor se considera el tiempo de respuesta del servidor IMAP.

El proceso se muestra a detalle en la función `sensor_correo()` de la figura 2.7. Dicha función es parte del programa `cliente.py` escrito en Python que contiene la funcionalidad de todos los sensores solicitados en esta práctica.

```

106 def sensor_correo():
107     """
108     Antes de ejecutar la funcion se debe establecer una conexion
109     inicial con el servidor ssh para intercambiar firmas.
110     """
111     ssh_hostname = '192.168.0.32' # IP del servidor
112     ssh_port = 22
113     ssh_username = 'root'
114     ssh_password = 'holal23..'
115
116     sender = 'root'
117     sender_pass = 'holal23..'
118     receiver = 'marcela@redes3.local'
119     server = 'server1.redes3.local' # Cambiar la IP en /etc/hosts
120     email_port = 25
121
122     correo = MIMEText('')
123     correo['From'] = sender
124     correo['To'] = receiver
125     correo['Subject'] = "sensor-SMTP:redes3.local"
126     correo.attach(MIMEText('Correo de prueba - Hacer caso omiso.'))
127
128     mailServer = smtplib.SMTP(server,25)
129     mailServer.ehlo()
130     mailServer.starttls()
131     mailServer.ehlo()
132     mailServer.login(sender, sender_pass)
133     tiempo_inicio = int(round(time.time() * 1000)) # Milisegundos
134     #print "Tiempo inicial: "+str(tiempo_inicio)
135     mailServer.sendmail(sender,receiver,correo.as_string()) # No se por que se puede hacer esto sin autent
136     tiempo_fin_smtp = int(round(time.time() * 1000))
137     tiempo_smtp = abs(tiempo_fin_smtp - tiempo_inicio)
138     print "Tiempo de respuesta SMTP: "+str(tiempo_smtp) + 'ms'
139     mailServer.close()
140
141     paramiko.util.log_to_file('paramiko.log')
142     s = paramiko.SSHClient()
143     s.load_system_host_keys()
144     s.connect(ssh_hostname,ssh_port,ssh_username,ssh_password)
145     stdin, stdout, stderr = s.exec_command("find /home/marcela/Maildir/new/ -printf '%C\\n' | tail -1")
146     tiempo_imap = abs(int(round(float(stdout.read()) * 1000)) - tiempo_inicio)
147     print "Tiempo de respuesta IMAP: " + str(tiempo_imap) + 'ms'
148     print "Tiempo total: "+str(tiempo_smtp + tiempo_imap) + 'ms'
149
150     stdin, stdout, stderr = s.exec_command("find /home/marcela/Maildir/new/ | tail -1")
151     archivo = stdout.read()
152     stdin, stdout, stderr = s.exec_command("rm "+archivo)
153     s.close()

```

Figura 2.7: Código en Python del sensor de correo.

De la línea 122 a la línea 131 se realiza la preparación del correo de prueba. En la línea 133 se toma el tiempo inicial y posteriormente, en la línea 135 se realiza la solicitud de envío SMTP. Una vez que termina, en la línea 136 se vuelve a tomar el tiempo que indica la respuesta SMTP. A partir de la línea 141 a la línea 153 se realiza una conexión mediante SSH para obtener el tiempo EPOCH del nombre del último archivo modificado en la bandeja de entrada del usuario de prueba. Finalmente, utilizando SSH se elimina el correo de prueba.

#### 2.1.2.1. Funcionamiento

Inciamos corriendo el programa **cliente.py** de la siguiente manera:

```
python cliente.py
```

Se despliega el menú de opciones y elegimos la opción 1 que corresponde al sensor SMTP. Con ello, el sensor realizará todas las operaciones antes mencionadas y mostrará en pantalla los tiempos obtenidos como se puede observar en la figura 2.8

```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):1

---- SENSOR SMTP ----
Tiempo de respuesta SMTP: 498ms
Tiempo de respuesta IMAP: 328706ms
Tiempo total: 329204ms
```

Figura 2.8: Funcionamiento del sensor de correo.

## 2.2. Supervisión de servidor web (HTTP)

Para la supervisión del funcionamiento de este servidor, se realizó primero la instalación que es muy sencilla en una máquina virtual Ubuntu Server, en la cual se ingresó el comando **sudo apt-get install apache2**, mostrado en la figura 2.9 mismo que instalaba las librerías necesarias de este servidor.

```
march@march-Lenovo-Y50-70: ~
march@march-Lenovo-Y50-70:~$ sudo apt-get install apache2
[sudo] password for march:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0
0 upgraded, 9 newly installed, 0 to remove and 40 not upgraded.
Need to get 1 540 kB of archives.
After this operation, 6 373 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libapr1 amd64 1.5.2-3 [86,0 kB]
Get:2 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1 amd64 1.5.4-1build1 [77,1 kB]
Get:3 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1-dbd-sqlite3 amd64 1.5.4-1build1 [10,6 kB]
Get:4 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1-ldap amd64 1.5.4-1build1 [8 720 B]
Get:5 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 liblua5.1-0 amd64 5.1.5-8ubuntu1 [102 kB]
Get:6 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-bin amd64 2.4.18-2ubuntu3.9 [925 kB]
Get:7 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-utils amd64 2.4.18-2ubuntu3.9 [81,8 kB]
Get:8 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-data all 2.4.18-2ubuntu3.9 [162 kB]
Get:9 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2 amd64
```

Figura 2.9: Comando de instalación HTTP apache2.

Una vez que se realizó la instalación de los paquetes correspondientes a dicho servidor, se ejecutó el comando **sudo systemctl status apache2** con el cual se verificaba que el servidor estuviera activo como se observa en la figura 2.10.

```
march@march-Lenovo-Y50-70: ~  
march@march-Lenovo-Y50-70:~$ sudo systemctl status apache2  
● apache2.service - LSB: Apache2 web server  
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)  
   Drop-In: /lib/systemd/system/apache2.service.d  
            └─apache2-systemd.conf  
   Active: active (running) since lun. 2018-12-03 20:18:59 CST; 42s ago  
     Docs: man:systemd-sysv-generator(8)  
    CGroup: /system.slice/apache2.service  
            └─4987 /usr/sbin/apache2 -k start  
               4990 /usr/sbin/apache2 -k start  
               4991 /usr/sbin/apache2 -k start  
  
déc. 03 20:18:58 march-Lenovo-Y50-70 systemd[1]: Starting LSB: Apache2 web serve  
déc. 03 20:18:58 march-Lenovo-Y50-70 apache2[4965]: * Starting Apache httpd web  
déc. 03 20:18:58 march-Lenovo-Y50-70 apache2[4965]: AH00558: apache2: Could not  
déc. 03 20:18:59 march-Lenovo-Y50-70 apache2[4965]: *  
déc. 03 20:18:59 march-Lenovo-Y50-70 systemd[1]: Started LSB: Apache2 web server
```

Figura 2.10: Verificación de status servidor HTTP.

Y una vez que se verifica que su status es activo, se visualiza en terminal la IP de la máquina con el fin de ingresarla en el navegador mediante el cual se obtiene la página de inicio del servidor de Apache (figuras 2.11 y 2.12).

```
march@march-Lenovo-Y50-70: ~  
wlp8s0    Link encap:Ethernet  HWaddr 2c:33:7a:19:13:33  
          inet addr:192.168.1.66  Bcast:192.168.1.255  Mask:255.255.255.0  
          inet6 addr: fe80::3e0f:d2d:2445:8593/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:250759 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:161718 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:289029402 (289.0 MB)  TX bytes:21731661 (21.7 MB)  
  
march@march-Lenovo-Y50-70:~$
```

Figura 2.11: Visualización de ip.

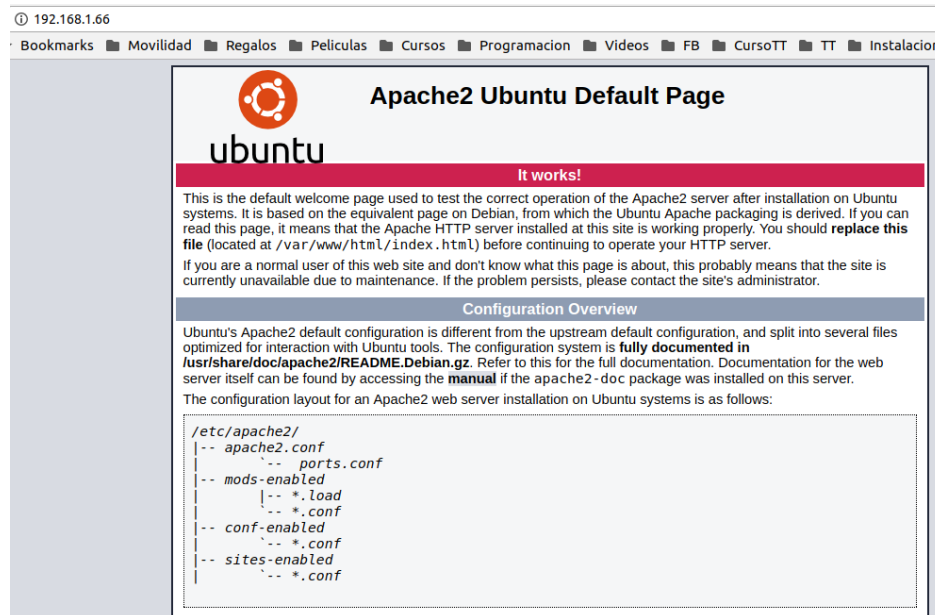


Figura 2.12: Página de Apache2 en el navegador.

## 2.3. Supervisión de servidor de archivos (FTP/FTP Server File Count)

### 2.3.1. Sensor FTP

Para la utilización de este sensor, se realizó primero la instalación del servidor FTP en una máquina virtual Ubuntu Server, en la cual se ingresó el comando **sudo apt-get install vsftpd**, mostrado en la figura 2.13 mismo que instalaba las librerías necesarias de este servidor.

```

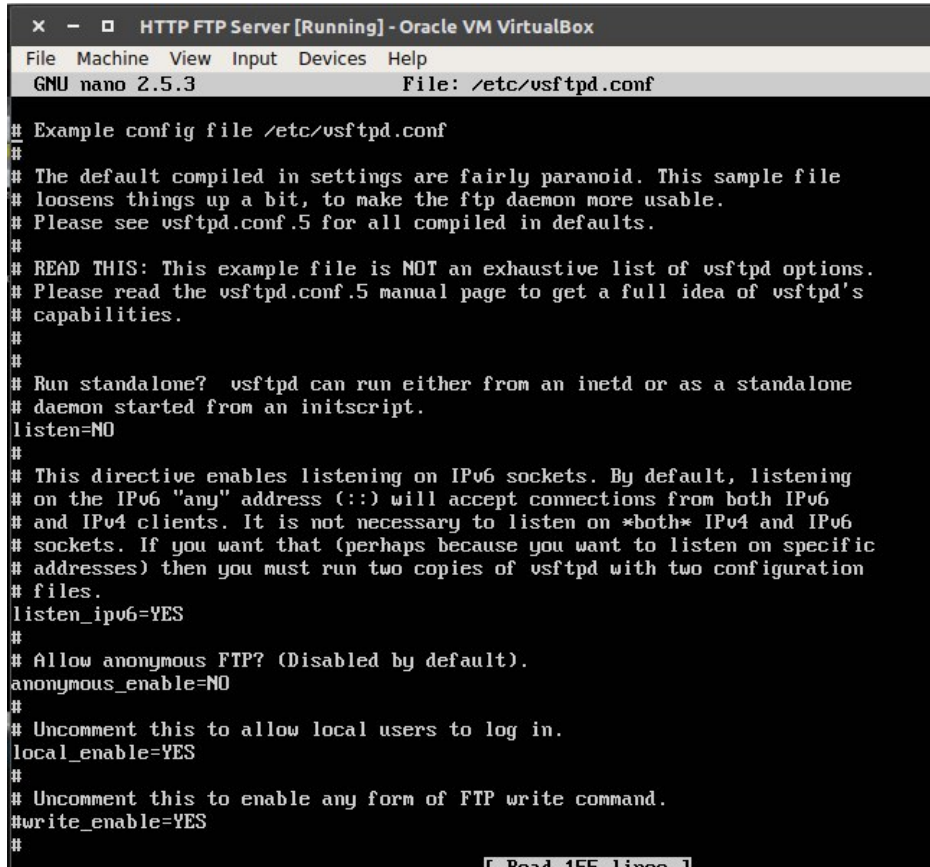
x - □ HTTP FTP Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
marce@httpUbuntu:~$ sudo apt-get install vsftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libwrap0 tcpd
The following NEW packages will be installed:
  libwrap0 tcpd vsftpd
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 185 kB of archives.
After this operation, 573 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libwrap0 amd64 7.6.q-25 [46.2 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 tcpd amd64 7.6.q-25 [23.0 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 vsftpd amd64 3.0.3-3ubuntu2 [115 kB]
Fetched 185 kB in 3s (53.1 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libwrap0:amd64.
(Reading database ... 90616 files and directories currently installed.)
Preparing to unpack .../libwrap0_7.6.q-25_amd64.deb ...
Unpacking libwrap0:amd64 (7.6.q-25) ...
Selecting previously unselected package tcpd.

```

Figura 2.13: Comando de instalación FTP.

Posteriormente, ya que se había realizado toda la descarga de paquetes, se ingreso mediante el comando **sudo nano /etc/vsftpd.conf**, como lo muestra la figura 2.14 con el fin de eliminar el comentario de la línea

**write\_enable = YES** y de esta manera permitir la escritura de archivos dentro del servidor.



```
X - □ HTTP FTP Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 2.5.3 File: /etc/vsftpd.conf

# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Run standalone? vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# 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
#
[ Read 155 Lines ]
```

Figura 2.14: Archivo de configuración FTP.

Por último, se ejecutó el comando **sudo service vsftpd restart** con el cual se reinicia el servicio FTP y posteriormente el comando **sudo service vsftpd status** para asegurarnos de que el estado del servicio sea activo. Por último únicamente se obtuvo la ip de la máquina virtual para saber el host al cual se enviarían y se recibirían los datos almacenados en dicho servidor (figura 2.15).

```

X - □ HTTP FTP Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
marce@httpUbuntu:~$ sudo service vsftpd restart
[sudo] password for marce:
marce@httpUbuntu:~$ sudo service vsftpd status
• vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor p
   Active: active (running) since Sun 2018-11-25 20:59:41 CST; 10s ago
   Process: 2214 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=e
   Main PID: 2220 (vsftpd)
   Tasks: 1
   Memory: 388.0K
   CPU: 3ms
   CGroup: /system.slice/vsftpd.service
           └─2220 /usr/sbin/vsftpd /etc/vsftpd.conf

Nov 25 20:59:41 httpUbuntu systemd[1]: Starting vsftpd FTP server...
Nov 25 20:59:41 httpUbuntu systemd[1]: Started vsftpd FTP server.
marce@httpUbuntu:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:d5:09:ba
          inet addr:192.168.1.69  Bcast:192.168.1.255  Mask:255.255.255.
          inet6 addr: fe80::a00:27ff:fed5:9ba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10047 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7089 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12621968 (12.6 MB)  TX bytes:525263 (525.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)

```

Figura 2.15: Reinicio de servicio y status FTP.



### 2.3.2. Sensor FTP Server File Count

Para la implementación del contador de archivos que almacena el servidor FTP se realiza una conexión SSH y se enlistan y se cuentan los archivos en el directorio de almacenamiento del servidor. Para ello se utiliza el siguiente comando:

```
ls -l /home/ftp/ | wc -l
```

El código se puede observar en la figura 2.16.

```
155 def ftp_counter():
156     ssh_hostname = '10.100.70.39' # IP del servidor
157     ssh_port = 22
158     ssh_username = 'root'
159     ssh_password = 'holal23.,'
160
161     paramiko.util.log_to_file('paramiko.log')
162     s = paramiko.SSHClient()
163     s.load_system_host_keys()
164     s.connect(sssh_hostname,ssh_port,ssh_username,ssh_password)
165     stdin, stdout, stderr = s.exec_command("ls -l /home/samuel/ | wc -l")
166     print 'Numero de archivos alojados en el servidor FTP: '+stdout.read()
167     s.close()
```

Figura 2.16: Código del contador de archivos en el servidor FTP.

El funcionamiento se puede ver en la figura 2.17.

```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):4

---- SENSOR FTP Server File Count ----
Numero de archivos alojados en el servidor FTP: 10
```

Figura 2.17: Contador de archivos en el servidor FTP.

## 2.4. Supervisión de servidor de impresión (SNMP)

## 2.5. Supervisión de servidor de acceso remoto (SSH)

### 2.5.1. Instalación



Para el desarrollo de esta práctica se optó por implementar el protocolo SSH *Secure Shell*. La instalación de este servidor de acceso remoto es muy simple. Basta con ejecutar el comando:

```
apt-get install openssh-server
```

En caso de que el servicio no sea iniciado automáticamente después de la instalación se ejecuta el comando:

```
service sshd start
```

### 2.5.2. Sensor SSH

Para la implementación del sensor se monitorizaron 4 aspectos: el número total de conexiones, dirección IP origen, dirección IP destino y usuario al que se encuentra conectada la sesión. Para obtener estos datos se ejecuta la función `sensor.ssh()` del programa `cliente.py`. En la figura 2.18 se puede observar a detalle el código.

```
169 def sensor_ssh():
170     ssh_hostname = '10.100.70.39' # IP del servidor
171     ssh_port = 22
172     ssh_username = 'root'
173     ssh_password = 'holal23..'
174
175     paramiko.util.log_to_file('paramiko.log')
176     s = paramiko.SSHClient()
177     s.load_system_host_keys()
178     s.connect(ssh_hostname, ssh_port, ssh_username, ssh_password)
179     stdin, stdout, stderr = s.exec_command('netstat -tnpa | grep 'ESTABLISHED.*sshd' | wc -l')
180     num = int(stdout.read())
181     print 'Número de conexiones SSH en el servidor: '+str(num)
182     if(num > 0):
183         print "Destino      Origen      Usuario\n"
184         stdin, stdout, stderr = s.exec_command('netstat -tnpa | grep 'ESTABLISHED.*sshd' | tr -s ' ' | cut -d' ' -f4,5,8)
185         print stdout.read()
186     s.close()
```

Figura 2.18: Código del sensor SSH.

En esta función podemos ver que se realiza una conexión SSH al servidor para ejecutar el comando `netstat -tnpa | grep 'ESTABLISHED.*sshd' | wc -l` para obtener el total de conexiones SSH establecidas. Si el número de conexiones es mayor a cero se ejecuta el comando `netstat -tnpa | grep 'ESTABLISHED.*sshd' | tr -s ' ' | cut -d' ' -f4,5,8` con lo cual se obtiene la información requerida por cada conexión establecida.

#### 2.5.2.1. Funcionamiento

La ejecución de este sensor se puede observar en la figura 2.19

```

root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

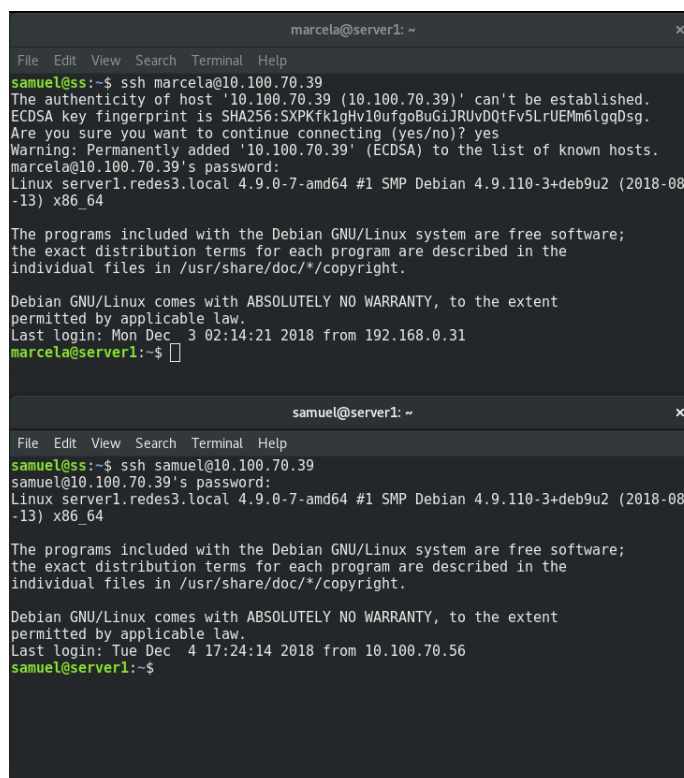
---- Menu de opciones ----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):6

---- SENSOR SSH ----
Numero de conexiones SSH en el servidor: 3
Destino      Origen      Usuario
10.100.70.39:22 10.100.70.56:44370 marcela
10.100.70.39:22 10.100.70.56:44378 root@not
10.100.70.39:22 10.100.70.56:44376 samuel

```

Figura 2.19: Funcionamiento del sensor SSH.

Este resultado se obtiene al tener iniciadas dos sesiones de SSH como se muestra en la figura 2.20.



```

marcela@server1: ~
File Edit View Search Terminal Help
samuel@ss:~$ ssh marcela@10.100.70.39
The authenticity of host '10.100.70.39 (10.100.70.39)' can't be established.
ECDSA key fingerprint is SHA256:5XPKfklgHv10ufgoBuG1JRuvDQtFv5LrUEMm6lgqDsg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.100.70.39' (ECDSA) to the list of known hosts.
marcela@10.100.70.39's password:
Linux server1.redes3.local 4.9.0-7-amd64 #1 SMP Debian 4.9.110-3+deb9u2 (2018-08-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 3 02:14:21 2018 from 192.168.0.31
marcela@server1:~$

samuel@server1: ~
File Edit View Search Terminal Help
samuel@ss:~$ ssh samuel@10.100.70.39
samuel@10.100.70.39's password:
Linux server1.redes3.local 4.9.0-7-amd64 #1 SMP Debian 4.9.110-3+deb9u2 (2018-08-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Dec 4 17:24:14 2018 from 10.100.70.56
samuel@server1:~$

```

Figura 2.20: Conexiones SSH.

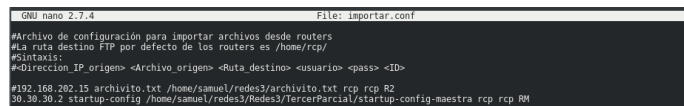
Es importante recalcar que el sensor reporta 3 sesiones ya que también cuenta la sesión que utiliza el mismo sensor para obtener los datos.

## 2.6. Administración de archivos de configuración

La administración de archivos de configuración se divide en dos partes. La importación y la exportación. Para ambos casos se utilizó FTP como protocolo de intercambio de archivos. Ambas funciones se encuentran definidas dentro del programa **cliente.py** y cada una de ellas necesita un archivo de configuración en el que se definen las direcciones IP destino u origen, rutas de archivos, usuarios y contraseñas, etc.

### 2.6.0.1. Importación

El archivo de configuración para la función de importación se puede observar en la figura 2.21. Podemos observar que en la parte superior se define la sintaxis de cada regla. Esto permite agregar los parámetros de diferentes routers y que la ejecución del programa sea fluido.



```
GNU nano 2.7.4 File: importar.conf
#Archivo de configuración para importar archivos desde routers
#La ruta destino FTP por defecto de los routers es /home/rcp/
#Sintaxis:
#<direction IP origen> <Archivo origen> <Ruta destino> <usuario> <pass> <ID>
#192.168.202.15 archivo1.txt /home/samuel/redes3/archivo1.txt rcp rcp R2
#9.39.38 > startup-config /home/samuel/redes3/Redes3/TercerParcial/startup-config-maestra rcp rcp RM
```

Figura 2.21: Archivo de configuración para importaciones.

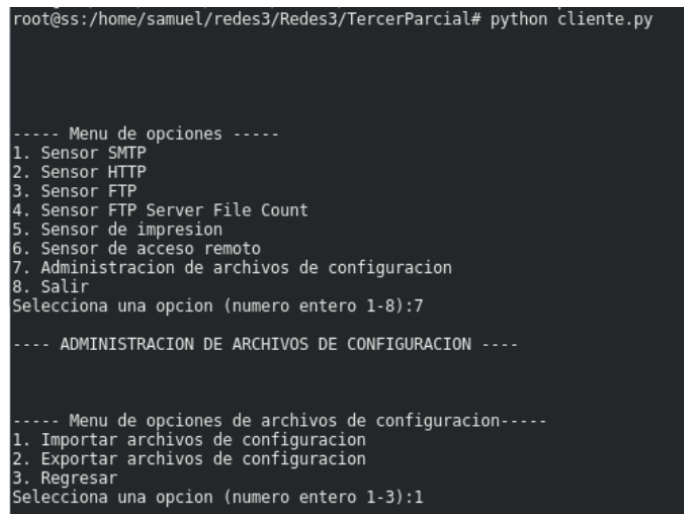
El código de la función **importar()** se puede ver en la figura 2.22. Que consiste básicamente en un ciclo que busca líneas que no comiencen con el carácter #, es decir, que no estén comentadas. Una vez que obtiene una línea descompone la cadena y obtiene los parámetros necesarios para realizar una conexión FTP y la respectiva descarga del archivo (también definido en el archivo de configuración).



```
199 def importar():
200     with open('importar.conf','r') as exportar:
201         for linea in exportar.readlines():
202             linea = linea.rstrip().split(' ')
203             if len(linea[0]) > 0:
204                 if linea[0][0] != '#':
205                     print linea
206                     ftp = FTP(linea[0]) # Connect
207                     ftp.login(linea[3],linea[4]) # Login
208                     file = open(linea[2]+'-'+linea[5]+'-'+str(time.time()),'wb') # Abre archivo local
209                     ftp.retrbinary('RETR %s' % linea[1],file.write) # Upload
210                     ftp.quit()
```

Figura 2.22: Código para importar archivos.

El funcionamiento se puede observar en la figura 2.23.



```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):7

---- ADMINISTRACION DE ARCHIVOS DE CONFIGURACION ----

----- Menu de opciones de archivos de configuracion-----
1. Importar archivos de configuracion
2. Exportar archivos de configuracion
3. Regresar
Selecciona una opcion (numero entero 1-3):1
```

Figura 2.23: Funcionamiento de la importación de archivos.

### 2.6.0.2. Exportación

El archivo de configuración para la función de exportación se puede observar en la figura 2.24. Podemos observar que en la parte superior se define la sintaxis de cada regla. Al igual que la importación, esto permite agregar los parámetros de diferentes routers y que la ejecución del programa sea fluido.

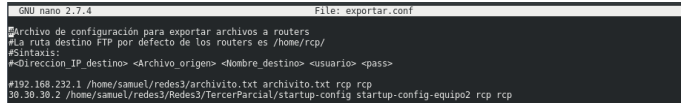


Figura 2.24: Archivo de configuración para exportaciones.

El código de la función **exportar()** se puede ver en la figura 2.25. Que, al igual que **importar()**, consiste básicamente en un ciclo que busca líneas que no comiencen con el carácter #, es decir, que no estén comentadas. Una vez que obtiene una línea descompone la cadena y obtiene los parámetros necesarios para realizar una conexión FTP y la respectiva subida del archivo (también definido en el archivo de configuración).



Figura 2.25: Código para exportar archivos.

El funcionamiento se puede observar en la figura 2.26.

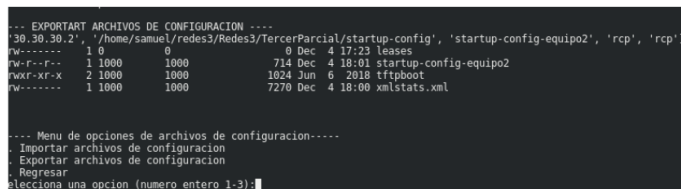


Figura 2.26: Funcionamiento de la exportación de archivos.

---

### Conclusiones

---

- **Castro Flores Marcela**
- **Sánchez Cruz Rosa María**
- **Santiago Mancera Arturo Samuel**

La práctica fue muy interesante porque nos permitió experimentar en GNS3 y enrutamiento. Lo cual son conocimientos básicos que deberían tomarse con más detalle. Así mismo, cabe resaltar que la implementación de servicios fue uno de los puntos más complicados. Sobre todo con respecto al servidor de correo.

Por otra parte, para la implementación de los sensores es importante señalar que se utilizaron soluciones simples como conexiones SSH y FTP. Sólo se tuvieron complicaciones con el sensor SSH para obtener los bytes transferidos por sesión y con el sensor de impresión al no contar con una impresora real.

---

## Referencias y bibliografías

---

- [1] JUAN C. HERNÁNDEZ M., *Método Holt Winters* (2017). Disponible en: [http://rstudio-pubs-static.s3.amazonaws.com/283175\\_1d0898ed1b704812a4eeb29b1fdcb213.html](http://rstudio-pubs-static.s3.amazonaws.com/283175_1d0898ed1b704812a4eeb29b1fdcb213.html) [Consultado el 01 Nov. 2018].
- [2] OMAR MAGUIÑA G., *El Método de Pronóstico de Holt Winters* (2016). Disponible en: <https://administration21.files.wordpress.com/2017/01/pronc3b3sticos-holt-winters-omr-nov2016.pdf> [Consultado el 01 Nov. 2018].
- [3] CISCO., *White Paper de las mejores prácticas del proceso de línea de base* (2015). Disponible en: [http://www.cisco.com/cisco/web/support/LA/102/1025/1025763\\_HAS\\_baseline.pdf](http://www.cisco.com/cisco/web/support/LA/102/1025/1025763_HAS_baseline.pdf) [Consultado el 01 Nov. 2018].
- [4] NET.SNMP., *HOST-RESOURCES-MIB* (2011). Disponible en: <http://www.netsnmp.org/docs/mibs/host.html> [Consultado el 01 Nov. 2018].