



UNIVERSIDAD DE MURCIA

SERVICIOS TELEMÁTICOS

Práctica Final

3º Grado de Ingeniería Informática

Luis Miguel García Macías – luismiguel.garcia@um.es

Profesor: Rafael Marín López – rafa@um.es

Versión 2.0 (entrega Post-Febrero)

Convocatoria de Junio 2016

ÍNDICE

1. INTRODUCCIÓN	PÁG. 04
2. ESCENARIO DE DESARROLLO	PÁG. 05
3. FUNCIONALIDADES OBLIGATORIAS	PÁG. 06
3.1 Desplegar Servicio DNS	PÁG. 06
<u>3.1.1 Trazas en Wireshark</u>	PÁG. 10
3.2 Desplegar Servicios SMTP/POP	PÁG. 12
<u>3.2.1 Trazas en Wireshark</u>	PÁG. 15
3.3 Desplegar Servicios HTTP/HTTPS	PÁG. 24
<u>3.3.1 Apache HTTP(S) Server</u>	PÁG. 24
3.3.1.1 Autenticación de usuario HTTP básica	PÁG. 28
3.3.1.2 Trazas en Wireshark	PÁG. 29
<u>3.3.2 Servicio HTTPS</u>	PÁG. 31
<u>3.3.3 Web-SSTT HTTP Server</u>	PÁG. 42
3.3.3.1 Funcionamiento del Software	PÁG. 42
3.3.3.2 Código Implementado	PÁG. 42
3.3.3.3 Trazas en Wireshark	PÁG. 47
3.4 Desplegar Servicio NTP	PÁG. 48
<u>3.4.1 Configuración del Servidor</u>	PÁG. 48
<u>3.4.2 Configuración del Cliente</u>	PÁG. 49
<u>3.4.3 Trazas en Wireshark</u>	PÁG. 49
3.5 IPsec	PÁG. 51
<u>3.5.1 Configuración</u>	PÁG. 51
3.5.1.1 Cabeceras AH	PÁG. 51
3.5.1.2 Cabeceras ESP	PÁG. 54
<u>3.5.2 Implementando IKE</u>	PÁG. 55

4. COMUNICAR LAS MAQUINAS VIRTUALES	PÁG. 58
5. PROBLEMAS ENCONTRADOS	PÁG. 59
6. APARTADO EXTRA PARA VERSION 2 (Webserver en C)	PÁG. 60
6.1 Código función debug	PÁG. 60
6.2 Código para Cookie	PÁG. 61
7. CONCLUSIONES Y VALORACIÓN PERSONAL	PÁG. 62
7.1 Conclusiones	PÁG. 62
7.2 Valoración Personal	PÁG. 62

1. INTRODUCCIÓN

Este documento se corresponde con la memoria de la práctica final de la asignatura. Esta práctica tiene como base las diferentes prácticas que han sido desarrolladas en las sesiones de la asignatura. El objetivo de este trabajo es el desarrollo, configuración y puesta en funcionamiento de un entorno de distintos Servicios Telemáticos.

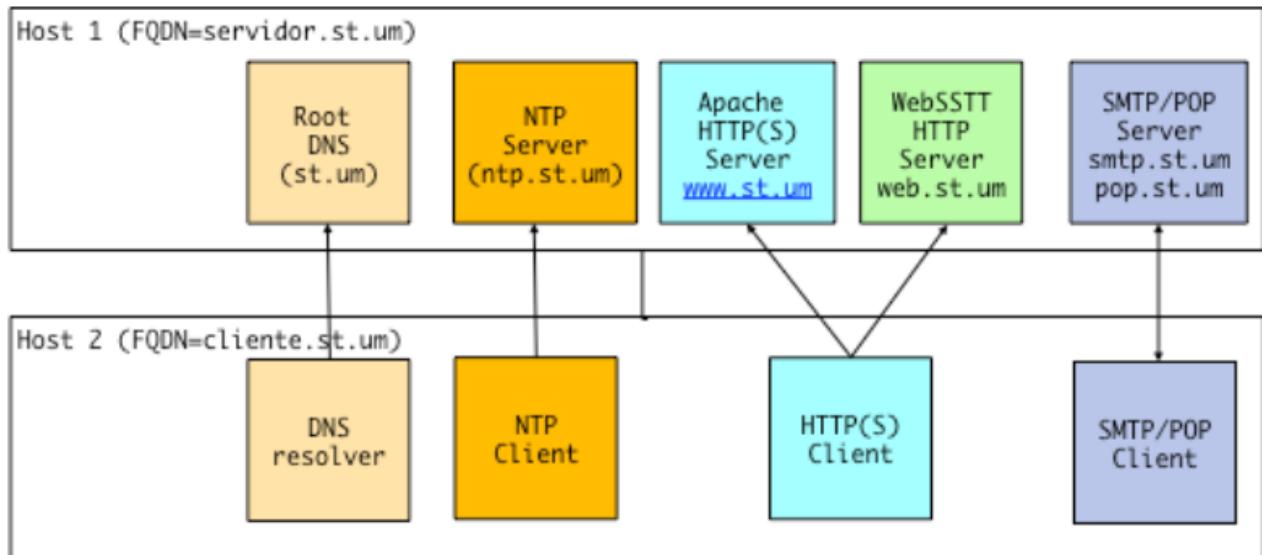


Ilustración 1: Escenario objetivo SSTT

Como podemos ver en la ilustración, contaremos con un equipo **Servidor** (servidor.st.um) y otro **Cliente** (cliente.st.um), donde existirá comunicación entre los mismos.

2. ESCENARIO DE DESARROLLO

Para poder realizar este proyecto hemos recurrido al software de virtualización de máquinas **Oracle VM Virtual Box**. Gracias a este entorno hemos podido crear dos máquinas independientes, donde una de ellas realiza la labor de Servidor mientras que la otra es el Cliente.

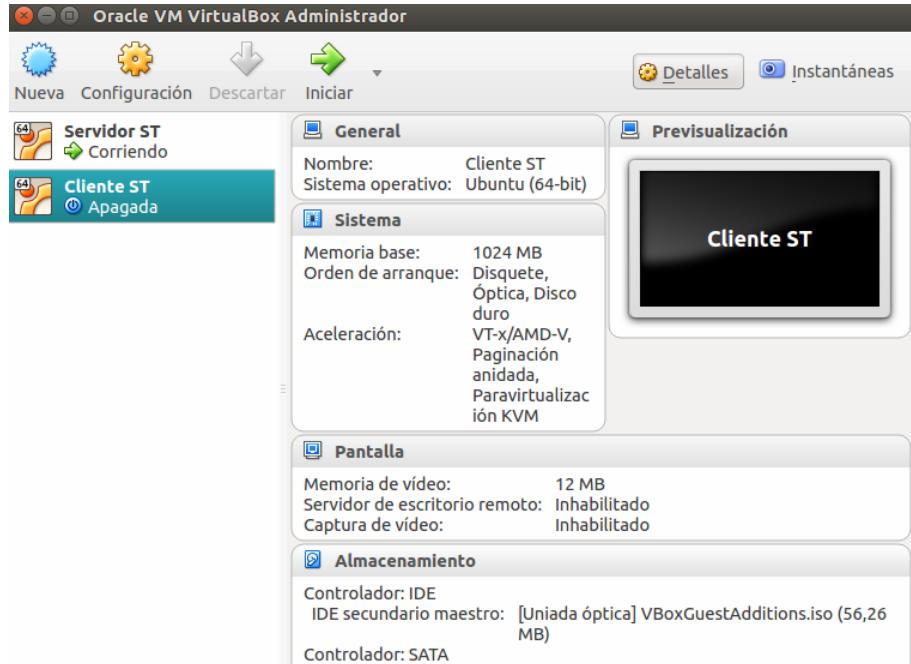


Ilustración 2: Panel administrador de máquinas Virtual Box

La máquina anfitrión es un **Portátil Asus A52J** y funciona con el **sistema operativo Ubuntu 14.04 LTS**. Para las máquinas virtualizadas hemos utilizado el sistema operativo **Ubuntu 12.04 LTS**, ya que las sesiones prácticas estaban orientadas para dicho sistema. Para el servidor podríamos haber empleado una versión de Ubuntu Server, pero por la gran comodidad que proporciona el entorno gráfico hemos optado por emplear una distribución normal.

3. FUNCIONALIDADES OBLIGATORIAS

Conjunto de funcionalidades mínimas con las que debe de contar la práctica final para su correcto funcionamiento.

3.1 Desplegar Servicio DNS

Domain Name Services (DNS) es un servicio de conversión de nombres de máquinas en direcciones IP. Se emplea para poder utilizar nombres de máquinas con nombres en lugar de números, ya que esto hace que sea mas sencillo de recordar. DNS nació de la necesidad de recordar fácilmente los nombres de todos los servidores conectados a Internet.

“Este sistema asocia información variada con nombres de dominios asignado a cada uno de los participantes. Su función más importante es traducir (resolver) nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.”

Este servicio funcionará como servidor DNS del cliente, de manera que las consultas DNS primero intentará resolverlas el servidor DNS. *“El DNS raíz en servidor.st.um deberá gestionar un único dominio de nivel principal llamado st.um. Para verificar el correcto funcionamiento se lanzará el resolver del DNS en cliente.st.um y se solicitará la resolución de diferentes nombres de equipo”.*

Si las consultas no puede resolverlas, las reenviará al servidor DNS de la Universidad de Murcia (cuya dirección IP es la 155.54.1.1) o al DNS esclavo de la Universidad de Murcia (155.54.15.240). *“El servicio DNS deberá hacer un reenvío a los servidores DNS de la UMU en el caso de que no sepa realizar una resolución”.*

Para la disponer del servicio DNS vamos a tener que instalar **BIND** (Berkeley Internet Name Domain software). Este software es utilizado aproximadamente por el 75% de los servidores DNS. Contiene un servidor de nombres (llamado *named*) y dos librerías resolver.

Para instalarlo podemos hacerlo gracias al siguiente comando:

```
sudo apt-get install bind9
```

- Ejecutable:

```
/usr/sbin/named
```

- Los comandos de control de Bind son los siguientes:

```
sudo etc/init.d/bind9 start | stop | restart | status
```

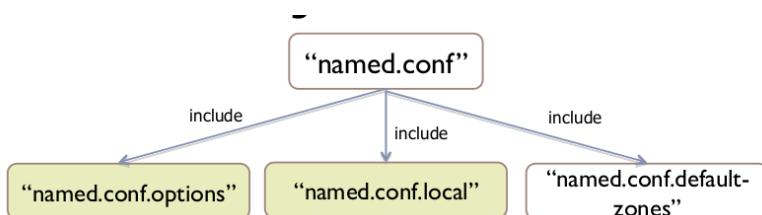


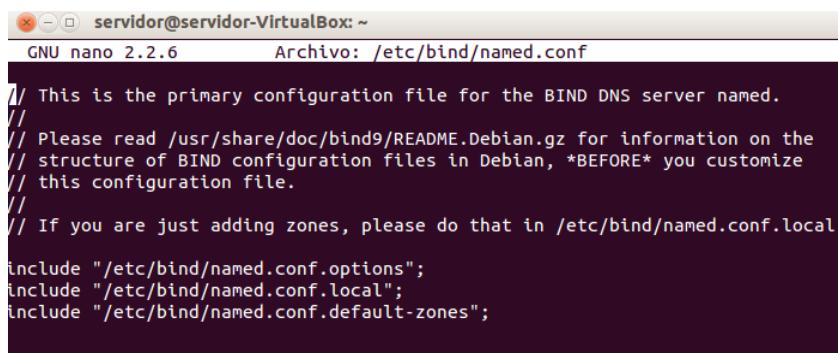
Ilustración 3: ficheros de configuración de BIND

Los ficheros más importantes y que deben de modificarse para el correcto funcionamiento son:

/etc/bind/named.conf: colección de declaraciones usando opciones anidadas que permite separar las secciones en diferentes ficheros utilizando *include*.

Declaraciones principales:

- logging: opciones de logeo para el servidor.
- options: opciones de configuración globales.
- zone: definición de las características de una zona.
- include: permite incluir archivos adicionales.



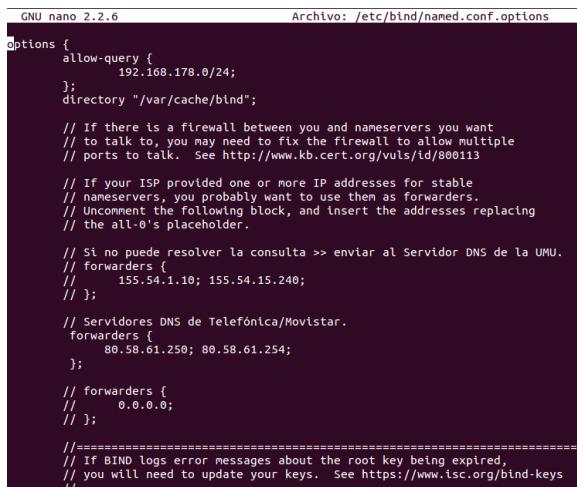
```
servidor@servidor-VirtualBox: ~
GNU nano 2.2.6          Archivo: /etc/bind/named.conf

// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

Ilustración 4: fichero named.conf

/etc/bind/named.conf.options: contiene las opciones de configuración globales del servidor. Se especifica la ubicación del directorio de trabajo, donde se guarda la copia local de resoluciones aprendidas. Las opciones más importantes son:

- allow-query: host que tienen permitido realizar consultas al servidor DNS (por defecto, todos los host tienen permiso).
- directory: establece el directorio de trabajo.
- forwarders: direcciones IP de servidores DNS donde reenviar las peticiones para ser resueltas (si no puede hacerlo este servidor).
- notify: Notificaciones a los servidores esclavos cuando una zona es actualizada { yes, no}.



```
GNU nano 2.2.6          Archivo: /etc/bind/named.conf.options

options {
    allow-query {
        192.168.178.0/24;
    };
    directory "/var/cache/bInd";

    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk. See http://www.kb.cert.org/vuls/id/800113

    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.
    // Uncomment the following block, and insert the addresses replacing
    // the all-0's placeholder.

    // Si no puede resolver la consulta >> enviar al Servidor DNS de la UMU.
    // forwarders {
    //     155.54.1.10; 155.54.15.240;
    // };

    // Servidores DNS de Telefónica/Movistar.
    forwarders {
        80.58.61.250; 80.58.61.254;
    };

    // forwarders {
    //     0.0.0.0;
    // };

    //=====
    // If BIND logs error messages about the root key being expired,
    // you will need to update your keys. See https://www.isc.org/bind-keys
    //=====
```

Ilustración 5: fichero named.conf.options

/etc/bind/named.conf.local: contiene las características de una zona administrada por este servidor DNS. Aquí definimos la zona “st.um”. Las opciones más importantes son:

- **allow-query:** host autorizados para pedir información en esta zona (por defecto, todas las peticiones son autorizadas).
- **allow-transfer:** servidores DNS esclavos que están autorizados para pedir una transferencia de información de la zona (por defecto, todas las peticiones son autorizadas).
- **file:** nombre del archivo, dentro del directorio de configuración, que contiene los datos de configuración de esta zona. En nuestro caso es **db.st.um.zone**.
- **type:** Define el tipo de zona:

master: El servidor actual tiene la autoridad para esta zona.

slave: Servidor esclavo para esta zona.

forward: Todas las peticiones a esta zona serán redirigidas a otros servidores DNS.

hint: Zona especial donde se definen los DNS raíz que sirven para resolver peticiones de una zona no conocida.

>> En nuestro caso es **master**, ya que el servidor DNS es el que tiene la autoridad para la zona.

- **masters:** Direcciones IP donde solicitar información autorizada para esta zona (solamente se usa si type=slave).

```
GNU nano 2.2.6                               Archivo: /etc/bind/named.conf.local

// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "st.um." IN {
    type master;
    file "/etc/bind/db/db.st.um.zone";
    allow-query { 192.168.178.0/24; };
    // allow-transfer { 192.168.178.84; }; Para DNS secundario. Emplearía esa IP.
};
```

Ilustración 6: fichero named.conf.local

/etc/bind/db.st.um.zone: fichero de zona de dominio que vamos a crear (contiene información sobre el dominio). Contiene el registro de recursos SOA (Start of Authority), que es el que indica las Directivas y los Registros de Recursos (RR).

- **Directivas:** contienen configuraciones especiales para la zona.

- **Registros de recursos (RR):** definen los parámetros de la zona y asignan identidades a host individuales.

```

GNU nano 2.2.6                               Archivo: /etc/bind/db.st.um.zone

;
; Archivo de configuracion de la zona st.um.
;
$TTL    86400
$ORIGIN st.um.
@      IN      SOA    dns.st.um.  usuario1.st.um (
                        1           ; Serial
                        3600        ; Refresh
                        1800        ; Retry
                        604800      ; Expire
                        3600 )      ; Negative Cache TTL
                IN  NS  dns
www      3600    IN      A      192.168.178.81
www1     3600    IN      A      192.168.178.81
www2     3600    IN      A      192.168.178.81
web      3600    IN      A      192.168.178.81
dns      3600    IN      A      192.168.178.81
smtp     3600    IN      A      192.168.178.81
pop      3600    IN      A      192.168.178.81
ntp      3600    IN      A      192.168.178.81

```

Ilustración 7: fichero de configuracion de zona db.st.um.zone

Explicación del fichero:

- **Directiva \$TTL:** ajusta el valor Time To Live (TTL) predeterminado para toda la zona, en segundos (tiempo durante el que un RR de zona es válido). En este caso 86400.

- **Directiva \$ORIGIN:** anexa el nombre del dominio a registros no cualificados. Cualquier nombre que NO termine en un punto (“.”) tendrá domain-name anexada. En este caso “st.um.”.

- Registros de Recursos:

IN: identifica la clase del registro como de Internet.

SOA: indica el inicio de los datos para una zona y define parámetros que afectan a todos los registros de la zona. Indicamos el servidor primario (dns.st.um) y el email del hostmaster (usuario1@st.um):

- **serial-number (Serial):** incrementarlo cada vez que el archivo de zona cambia para que el demonio named (o bind) recargue la zona. En este caso 1 segundo.
- **time-to-refresh (Refresh):** tiempo que un servidor esclavo debe esperar antes de preguntar al maestro si se han realizado cambios. En este caso 1 hora.
- **time-to-retry (Retry):** intervalo de tiempo que un servidor esclavo debe esperar antes de reemitter una petición de actualización de datos. En este caso 30 minutos.
- **time-to-expire (Expire):** si el servidor maestro no responde durante este tiempo, el servidor esclavo debe dejar de responder solicitudes. En este caso 1 semana.
- **minimum-TTL (Negative cache TTL):** tiempo mínimo que otros servidores deben almacenar en caché esta información de zona. En este caso 1 hora.

NS: identifica el servidor de nombres para el dominio.

Seguidamente aparecen las asociaciones de nombres de máquina con dirección IP que nos interesan. Una vez editado todo lo anterior tenemos que reiniciar BIND:

```
sudo /etc/init.d/bind9 restart
```

Para finalizar el trabajo en el Servidor tenemos que editar el fichero **/etc/resolv.conf** del Server e indicar la IP de nuestro Servidor (192.168.178.81) y quitar los otros que puedan aparecer:

```
GNU nano 2.2.6                               Archivo: /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#      DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.178.81
#search fritz.box
#nameserver 127.0.0.1
```

Ilustración 8: fichero resolv.conf del Servidor

En la máquina **Cliente** tenemos también que realizar la misma tarea anteriormente descrita en el archivo **/etc/resolv.conf**.

```
GNU nano 2.2.6                               Archivo: /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#      DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
#nameserver 127.0.0.1
#search fritz.box
nameserver 192.168.178.81
```

Ilustración 9: fichero resolv.conf del Cliente

3.1.1 Trazas en Wireshark

A continuación se van a mostrar las trazas que hemos obtenido con Wireshark.

1º Vamos a probar con una dirección no gestionada por el *servidor.st.um*.

No.	Time	Source	Destination	Protocol	Length	Info
27	21.849801	192.168.178.80	192.168.178.81	ICMP	318	Destination unreachable (Port unreachable)
28	21.850202	192.168.178.81	192.168.178.80	DNS	290	Standard query response A 91.189.89.240 A 91.189.90.40 A 91.189.90.41 A 91.
29	21.850672	192.168.178.80	192.168.178.81	DNS	76	Standard query A start.ubuntu.com
30	21.851235	192.168.178.81	192.168.178.80	DNS	290	Standard query response A 91.189.89.88 A 91.189.89.240 A 91.189.90.40 A 91.
36	23.829854	192.168.178.80	192.168.178.81	DNS	73	Standard query A www.google.es
46	28.831695	192.168.178.80	192.168.178.81	DNS	73	Standard query A www.google.es
47	33.835106	192.168.178.80	192.168.178.81	DNS	73	Standard query A www.aoole.es

Frame 36: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)
 ▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ User Datagram Protocol, Src Port: 56361 (56361), Dst Port: domain (53)
 ▶ Domain Name System (query)
 [Response In: 58]
 Transaction ID: 0xc64d
 Flags: 0x0100 (Standard query)
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0
 ▶ Queries
 ▶ www.google.es: type A, class IN
 Name: www.google.es
 Type: A (Host address)
 Class: IN (0x0001)

Ilustración 10: Cliente envía la petición www.google.es

Filter: dns Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
55	43.841326	192.168.178.80	192.168.178.81	DNS	73	Standard query A www.google.es
56	43.965278	192.168.178.81	192.168.178.80	DNS	135	Standard query response A 74.125.136.94
58	43.965346	192.168.178.80	192.168.178.81	ICMP	163	Destination unreachable (Port unreachable)
59	43.965552	192.168.178.81	192.168.178.80	DNS	135	Standard query response A 74.125.136.94
60	43.965569	192.168.178.80	192.168.178.81	ICMP	163	Destination unreachable (Port unreachable)
61	43.965865	192.168.178.81	192.168.178.80	DNS	135	Standard query response A 74.125.136.94
65	44.226676	192.168.178.80	192.168.178.81	DNS	73	Standard query A www.google.es
66	44.221892	192.168.178.81	192.168.178.80	DNS	135	Standard query response A 74.125.136.94
86	45.270211	192.168.178.80	192.168.178.81	DNS	79	Standard query A clients1.google.com
95	47.314949	192.168.178.80	192.168.178.81	DNS	76	Standard query A daisy.ubuntu.com
96	47.315696	192.168.178.81	192.168.178.80	DNS	258	Standard query response A 91.189.92.57 A 91.189.92.55
164	47.638306	192.168.178.80	192.168.178.81	DNS	75	Standard query A ssl.gstatic.com
165	48.169112	192.168.178.80	192.168.178.81	DNS	74	Standard query A www.gstatic.com
166	48.226267	192.168.178.80	192.168.178.81	DNS	75	Standard query A www.gstatic.com
167	48.226934	192.168.178.81	192.168.178.80	DNS	170	Standard query response A 216.58.211.3
188	48.319505	192.168.178.80	192.168.178.81	DNS	76	Standard query A daisy.ubuntu.com

▼ www.google.es: type A, class IN
 Name: www.google.es
 Type: A (Host address)
 Class: IN (0x0001)

Ilustración 11: Servidor responde a la petición www.google.es

2º Ahora vamos a probar una que sí es gestionada por servidor.st.um.

Filter: dns Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
2478	1174.389732	192.168.178.81	192.168.178.80	DNS	258	Standard query response A 91.189.92.55 A 91.189.92.57
2487	1200.668385	192.168.178.77	224.0.0.251	MDNS	87	Standard query PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM"
2488	1201.151623	fe80::1e4b:d6ff:fe8c::ff02::fb		MDNS	187	Standard query PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM"
2489	1202.155102	192.168.178.80	192.168.178.81	DNS	76	Standard query A daisy.ubuntu.com
2492	1207.160790	192.168.178.80	192.168.178.81	DNS	76	Standard query A daisy.ubuntu.com
2496	1212.163771	192.168.178.80	192.168.178.81	DNS	86	Standard query A daisy.ubuntu.com.fritz.box
2497	1212.164827	192.168.178.81	192.168.178.80	DNS	161	Standard query response, No such name
2501	1217.342023	fe80::a00:27ff:fe9e:7ff02::fb		MDNS	143	Standard query SRV MusicBox Share on MusicBox._smb._tcp.local, "QM" question SRV Share._smb._tcp.local, "QM"
2504	1217.535123	fe80::76da:38ff:fe0e::ff02::fb		MDNS	189	Standard query response AAAA, cache flush fe80::76da:38ff:fe0e:420d SRV, cache flushed
2505	1222.201387	192.168.178.81	192.168.178.80	DNS	258	Standard query response A 91.189.92.55 A 91.189.92.57
2506	1222.201460	192.168.178.80	192.168.178.81	ICMP	286	Destination unreachable (Port unreachable)
2516	1250.470504	192.168.178.80	192.168.178.81	DNS	69	Standard query A www.st.um
2517	1250.471183	192.168.178.81	192.168.178.80	DNS	119	Standard query response A 192.168.178.81
2525	1254.661533	192.168.178.81	224.0.0.251	MDNS	123	Standard query SRV MusicBox Share on MusicBox._smb._tcp.local, "QM" question SRV Share._smb._tcp.local, "QM"
2526	1254.823321	192.168.178.24	224.0.0.251	MDNS	169	Standard query response AAAA, cache flush fe80::76da:38ff:fe0e:420d SRV, cache flushed
2533	1259.290496	192.168.178.80	192.168.178.81	DNS	69	Standard query A www.st.um

AUTHORITY RRs: 0
 Additional RRs: 0
 ▼ Queries
 ▼ www.st.um: type A, class IN
 Name: www.st.um
 Type: A (Host address)
 Class: IN (0x0001)

Ilustración 12: Cliente envía la petición www.st.um

Filter: dns Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
2526	1254.823321	192.168.178.24	224.0.0.251	MDNS	169	Standard query response AAAA, cache flush fe80::76da:38ff:fe0e:420d SRV, cache flushed
2533	1259.290496	192.168.178.80	192.168.178.81	DNS	69	Standard query A www.st.um
2536	1259.291254	192.168.178.81	192.168.178.80	DNS	119	Standard query response A 192.168.178.81
2572	1259.670667	192.168.178.80	192.168.178.81	DNS	80	Standard query A fonts.googleapis.com
2576	1259.711286	192.168.178.80	192.168.178.81	DNS	77	Standard query A fonts.gstatic.com
2586	1260.043643	192.168.178.80	192.168.178.81	DNS	70	Standard query A www1.st.um
2587	1260.044278	192.168.178.81	192.168.178.80	DNS	120	Standard query response A 192.168.178.81
2588	1260.044559	192.168.178.80	192.168.178.81	DNS	70	Standard query A www2.st.um
2589	1260.044924	192.168.178.81	192.168.178.80	DNS	120	Standard query response A 192.168.178.81
2590	1260.045169	192.168.178.80	192.168.178.81	DNS	80	Standard query A luismiguelgarcia.com
2597	1263.310161	192.168.178.80	192.168.178.81	DNS	76	Standard query A daisy.ubuntu.com
2598	1263.311023	192.168.178.81	192.168.178.80	DNS	258	Standard query response A 91.189.92.57 A 91.189.92.55
2599	1264.312488	192.168.178.80	192.168.178.81	DNS	76	Standard query A daisy.ubuntu.com
2600	1264.313469	192.168.178.81	192.168.178.80	DNS	258	Standard query response A 91.189.92.55 A 91.189.92.57
2601	1264.673376	192.168.178.80	192.168.178.81	DNS	80	Standard query A fonts.googleapis.com
2602	1264.716881	192.168.178.80	192.168.178.81	DNS	77	Standard query A fonts.gstatic.com
2612	1265.040266	192.168.178.80	192.168.178.81	DNS	80	Standard query A luismiguelgarcia.com

Ilustración 13: Servidor responde a la petición www.st.um

3.2 Desplegar Servicios SMTP/POP

Para poder dar soporte **SMTP** tenemos que preparar el equipo. Para ello es necesario la instalación de **Exim4**. Exim4 es un agente de transporte de correo (Mail Transport Agent, usualmente MTA) desarrollado por la Universidad de Cambridge. Para su instalación tenemos que:

- Actualizar los paquetes:

```
sudo apt-get update
```

- Introducir el comando de instalación de Exim4 en la terminal:

```
sudo apt-get install exim4
```

- Realizar la configuración rápida del mismo:

```
sudo dpkg-reconfigure exim4-config
```

- ▶ **Tipo general del servidor:** Escoger primera opción (Internet site)
- ▶ **Nombre del sistema de correo:** "st.um"
- ▶ **Lista de IP desde las que se aceptan mensajes:** (en blanco → desde cualquier IP)
- ▶ **Destinos de los que se acepta correo:** "st.um"
- ▶ **Dominio para que se puede reenviar correo:** "st.um; um.es"
- ▶ **Lista de IPs para reenvío incondicional:** 192.168.56.0/24
- ▶ **Limitar consultas DNS:** NO
- ▶ **Formato de buzón de correo:** Maildir
- ▶ **Dividir ficheros de configuración:** NO

Ilustración 14: parámetros de configuración Exim4

NOTA: Nosotros hemos usado la IP 192.168.178.0/24.

- Tras la configuración reiniciamos Exim4:

```
sudo service exim4 restart
```

- Comprobamos el servicio:

```
netstat -a | more  
netstat -a | grep smtp  
netstat -a --numeric-ports | grep 25
```

Cuando hacemos la comprobación debemos de ver que nos aparece el puerto número 25.

```

servidor@servidor-VirtualBox:~$ sudo netstat -a --numeric-ports | grep 25
tcp      0      0 0.0.0.0:25                      0.0.0.0:*                  ESCUCHAR
tcp      1      0 192.168.178.81:41489          91.189.94.25:80            CLOSE_WAIT
tcp6     0      0 ::::25                         ::::*                  ESCUCHAR
udp      0      0 192.168.178.81:40936          80.58.61.254:53            ESTABLECIDO
udp      0      0 192.168.178.81:9039           80.58.61.254:53            ESTABLECIDO
unix    2      [ ACC ]   FLUJO    ESCUCHANDO    10825      /tmp/keyring-35ARsc/control
unix    3      [ ]       FLUJO    CONECTADO    12585
unix    3      [ ]       FLUJO    CONECTADO    12584      @/tmp/dbus-moRpmGy5h3
unix    3      [ ]       FLUJO    CONECTADO    12527
unix    3      [ ]       FLUJO    CONECTADO    12025
unix    3      [ ]       FLUJO    CONECTADO    12589      @/tmp/dbus-moRpmGy5h3
unix    3      [ ]       FLUJO    CONECTADO    12586      @/tmp/.X11-unix/X0
unix    3      [ ]       FLUJO    CONECTADO    12593      /var/run/dbus/system_bus_socket
unix    3      [ ]       FLUJO    CONECTADO    7250      @/com/ubuntu/upstart
unix    3      [ ]       FLUJO    CONECTADO    11925      /var/run/dbus/system_bus_socket
unix    3      [ ]       FLUJO    CONECTADO    12514      /var/run/dbus/system_bus_socket
unix    3      [ ]       FLUJO    CONECTADO    12513
unix    3      [ ]       FLUJO    CONECTADO    12591      @/tmp/dbus-moRpmGy5h3
unix    3      [ ]       FLUJO    CONECTADO    12523      @/tmp/dbus-moRpmGy5h3
unix    3      [ ]       FLUJO    CONECTADO    12528      @/tmp/dbus-moRpmGy5h3
unix    3      [ ]       FLUJO    CONECTADO    11825
unix    3      [ ]       FLUJO    CONECTADO    12588
unix    3      [ ]       FLUJO    CONECTADO    12583
unix    3      [ ]       FLUJO    CONECTADO    9825      /var/run/dbus/system_bus_socket
unix    3      [ ]       FLUJO    CONECTADO    12522
unix    3      [ ]       FLUJO    CONECTADO    11725      @/tmp/dbus-moRpmGy5h3
unix    3      [ ]       FLUJO    CONECTADO    12325
unix    2      [ ]       DGRAM
unix    3      [ ]       FLUJO    CONECTADO    12590
unix    3      [ ]       FLUJO    CONECTADO    12592
servidor@servidor-VirtualBox:~$ sudo netstat -a | grep smtp
tcp      0      0 *:smtp                     *:*                  ESCUCHAR
tcp6     0      0 [::]:smtp                   [::]:*                ESCUCHAR

```

Ilustración 15: comprobando puerto 25 SMTP

Una vez hemos terminado con el soporte a SMTP nos disponemos a dar soporte **POP**. Para ello es necesario instalar **Dovecot**. Dovecot es un servidor de IMAP y POP3 de código abierto desarrollado por Timo Sirainen. Para su instalación tenemos que:

- Instalar el Servidor POP con el siguiente comando en la Terminal:

```
sudo apt-get install dovecot-pop3d
```

Una vez tenemos instalado el servidor hay que editar los siguientes ficheros:

/etc/dovecot/conf.d/10-auth.conf:

- disable plaintext auth = no (pedir autenticación débil basada en texto plano)
- auth mechanisms = plain (activar autenticación débil basada en texto plano)

/etc/dovecot/conf.d/10-mail.conf:

- mail location = maildir:~/Maildir (especifica formato de los buzones de correo)

/etc/dovecot/conf.d/auth-system.conf.ext:

- Configura POP para que lea la información de los usuarios de sus cuentas en el SO (no es necesario modificarlo).

Una vez que hemos finalizado con la configuración tenemos que proceder a dar de alta a los nuevos usuarios en el sistema operativo:

- usuario1 – usuario1@st.um
 - usuario2 – usuario2@st.um
- Contraseña de ambos: mypassword

- Para darlos de alta simplemente tendremos que teclear lo siguiente en una terminal:

```
sudo useradd usuario1 -m  
sudo useradd usuario2 -m  
  
-  
sudo passwd usuario1  
sudo passwd usuario2
```

+ El buzón Maildir se crea automáticamente al acceder por primera vez. En caso contrario, usar el comando **maildirmake.dovecot ./Maildir** como el usuario correspondiente.

- Una vez finalizada la creación de usuarios, tenemos que reiniciar el Servicio POP:

```
sudo service dovecot restart
```

- Finalmente comprobamos el servicio:

```
netstat -a | more  
netstat -a | grep pop  
netstat -a --numeric-ports | grep 110
```

Cuando hacemos la comprobación debemos de ver que nos aparece el puerto número 110

```
servidor@servidor-VirtualBox:~$ sudo netstat -a --numeric-ports | grep 110  
[sudo] password for servidor:  
tcp        0      0 0.0.0.0:110          0.0.0.0:*          ESCUCHAR  
tcp        0      0 192.168.178.81:110        192.168.178.81:34047    TIME_WAIT  
tcp6       0      0 :::::110            ::::*              ESCUCHAR  
unix  3      [ ]        FLUJO      CONECTADO    11067  
unix  3      [ ]        FLUJO      CONECTADO    11065  @/tmp/dbus-moRpmGy5h3  
unix  3      [ ]        FLUJO      CONECTADO    11073  
unix  3      [ ]        FLUJO      CONECTADO    11024  @/tmp/dbus-moRpmGy5h3  
unix  3      [ ]        FLUJO      CONECTADO    11030  @/tmp/dbus-moRpmGy5h3  
unix  3      [ ]        FLUJO      CONECTADO    11029  
unix  3      [ ]        FLUJO      CONECTADO    11074  @/tmp/dbus-moRpmGy5h3  
unix  3      [ ]        FLUJO      CONECTADO    11064  
unix  3      [ ]        FLUJO      CONECTADO    11068  @/tmp/dbus-moRpmGy5h3  
unix  3      [ ]        FLUJO      CONECTADO    12110  @/tmp/dbus-moRpmGy5h3  
unix  3      [ ]        FLUJO      CONECTADO    11023  
servidor@servidor-VirtualBox:~$ sudo netstat -a | grep pop  
tcp        0      0 *:pop3             *:*              ESCUCHAR  
tcp        0      0 *:pop3s            *:*              ESCUCHAR  
tcp        0      0 www.st.um:pop3      www.st.um:34047    TIME_WAIT  
tcp6       0      0 [::]:pop3          [::]:*            ESCUCHAR  
tcp6       0      0 [::]:pop3s         [::]:*            ESCUCHAR  
unix  2      [ ACC ]     FLUJO      ESCUCHANDO   9438  /var/run/dovecot/login/pop3
```

Ilustración 16: comprobando puerto 110 POP

- Podemos hacer una prueba de conexión manual (para nosotros sería con la IP 192.168.178.81):

- ▶ Desde equipo cliente:
 - ▶ telnet 192.168.56.2 110
 - ▶ Realizar una consulta básica mediante comandos POP
 - ▶ telnet 192.168.56.2 25
 - ▶ Realizar un envío básico mediante comandos SMTP
 - ▶ Para salir : QUIT

Ilustración 17: comandos conexión manual

Una vez tenemos creados los usuarios del sistema, tenemos que configurar un Cliente de Correo Electrónico, en nuestro caso **Thunderbird**. Para ello, a la hora de dar de alta los emails, debemos de introducir la siguiente configuración por cada correo electrónico:

Configuración POP

- Nombre del servidor POP: pop.st.um
- Puerto: 110
- Nombre de usuario: usuarioX
- Seguridad de la conexión (SSL): ninguna
- Método de identificación: Contraseña, transmitida de manera insegura (Contraseña normal)

Configuración SMTP

- Nombre del servidor: smtp.st.um
- Puerto: 25
- Seguridad de la conexión (SSL): ninguna
- Método de autenticación: Contraseña, transmitida de manera insegura (Contraseña normal)
- Nombre de usuario: usuarioX@st.um

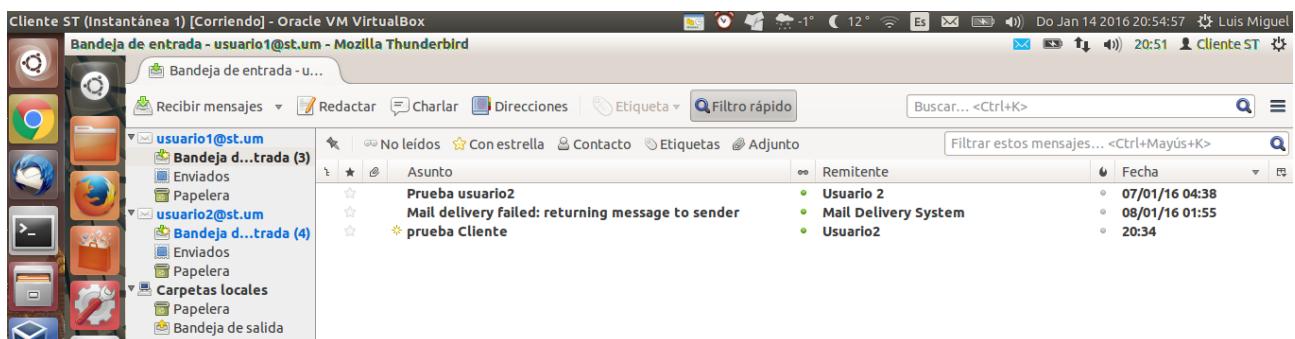


Ilustración 18: probando funcionamiento Thunderbird

Como paso final debemos de dar de alta en el Servidor DNS a los servidores SMTP y POP.

3.2.1 Trazas en Wireshark

A continuación se van a mostrar las trazas que hemos obtenido con Wireshark. Para ello vamos a realizar un envío de un mensaje entre los dos usuarios para poder realizar comprobaciones.

Vamos a dividir por SMTP y POP el análisis de trazas.

1º SMTP.

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3566	2352.947880	192.168.178.81	192.168.178.80	SMTP	139 S:	220 servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100		
3568	2352.955091	192.168.178.80	192.168.178.81	SMTP	89 C:	EHLO [192.168.178.80]		
3570	2352.956216	192.168.178.81	192.168.178.80	SMTP	176 S:	250-servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80] 250-SI		
3571	2352.957290	192.168.178.80	192.168.178.81	SMTP	183 C:	MAIL FROM:<usuario2@st.um> SIZE=442		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74 S:	250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92 C:	RCPT TO:<usuario1@st.um>		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA		
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself		
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes		
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from:	Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)		
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJvIC-0001C8-L8		
3581	2353.071756	192.168.178.80	192.168.178.81	SMTP	72 C:	QUIT		
3582	2353.072451	192.168.178.81	192.168.178.80	SMTP	110 S:	221 servidor-VirtualBox closing connection		
Frame 3566: 139 bytes on wire (1112 bits), 139 bytes captured (1112 bits)								
‣	Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)		Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)					
‣	Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81)		Dst: 192.168.178.80 (192.168.178.80)					
‣	Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 1, Ack: 1, Len: 73							
‣	Simple Mail Transfer Protocol							
▼	Response: 220 servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100\r\n							
	Response code: <domain> Service ready (220)							
	Response parameter: servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100							

Ilustración 19: el Servidor se presenta

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3566	2352.947880	192.168.178.81	192.168.178.80	SMTP	139 S:	220 servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100		
3568	2352.955091	192.168.178.80	192.168.178.81	SMTP	89 C:	EHLO [192.168.178.80]		
3570	2352.956216	192.168.178.81	192.168.178.80	SMTP	176 S:	250-servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80] 250-SI		
3571	2352.957290	192.168.178.80	192.168.178.81	SMTP	183 C:	MAIL FROM:<usuario2@st.um> SIZE=442		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74 S:	250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92 C:	RCPT TO:<usuario1@st.um>		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA		
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself		
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes		
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from:	Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)		
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJvIC-0001C8-L8		
3581	2353.071756	192.168.178.80	192.168.178.81	SMTP	72 C:	QUIT		
3582	2353.072451	192.168.178.81	192.168.178.80	SMTP	110 S:	221 servidor-VirtualBox closing connection		
Frame 3568: 89 bytes on wire (712 bits), 89 bytes captured (712 bits)								
‣	Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62)		Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)					
‣	Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80)		Dst: 192.168.178.81 (192.168.178.81)					
‣	Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 1, Ack: 74, Len: 23							
‣	Simple Mail Transfer Protocol							
▼	Command: EHLO [192.168.178.80]\r\n							
	Command: EHLO							
	Request parameter: [192.168.178.80]							

Ilustración 20: el Cliente se presenta

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3566	2352.947880	192.168.178.81	192.168.178.80	SMTP	139	S: 220 servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100		
3568	2352.955091	192.168.178.80	192.168.178.81	SMTP	89	C: EHLO [192.168.178.80]		
3570	2352.956216	192.168.178.81	192.168.178.80	SMTP	176	S: 250-servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80] 250-SI		
3571	2352.957290	192.168.178.80	192.168.178.81	SMTP	103	C: MAIL FROM:<usuario2@st.um> SIZE=442		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74	S: 250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92	C: RCPT TO:<usuario1@st.um>		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	90	S: 250 Accepted		
► Frame 3570: 176 bytes on wire (1408 bits), 176 bytes captured (1408 bits)								
► Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)								
► Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)								
► Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 74, Ack: 24, Len: 110								
▼ Simple Mail Transfer Protocol								
▼ Response: 250-servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80]\r\n								
Response code: Requested mail action okay, completed (250)								
Response parameter: servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80]								
▼ Response: 250-SIZE 52428800\r\n								
Response code: Requested mail action okay, completed (250)								
Response parameter: SIZE 52428800								
▼ Response: 250-PIPELINING\r\n								
Response code: Requested mail action okay, completed (250)								
Response parameter: PIPELINING								
▼ Response: 250 HELP\r\n								
Response code: Requested mail action okay, completed (250)								
Response parameter: HELP								

Ilustración 21: el Servidor responde Hello, pleased to meet you

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3566	2352.947880	192.168.178.81	192.168.178.80	SMTP	139	S: 220 servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100		
3568	2352.955091	192.168.178.80	192.168.178.81	SMTP	89	C: EHLO [192.168.178.80]		
3570	2352.956216	192.168.178.81	192.168.178.80	SMTP	176	S: 250-servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80] 250-SI		
3571	2352.957290	192.168.178.80	192.168.178.81	SMTP	103	C: MAIL FROM:<usuario2@st.um> SIZE=442		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74	S: 250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92	C: RCPT TO:<usuario1@st.um>		
► Frame 3571: 103 bytes on wire (824 bits), 103 bytes captured (824 bits)								
► Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)								
► Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)								
► Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 24, Ack: 184, Len: 37								
▼ Simple Mail Transfer Protocol								
▼ Command: MAIL FROM:<usuario2@st.um> SIZE=442\r\n								
Command: MAIL								
Request parameter: FROM:<usuario2@st.um> SIZE=442								

Ilustración 22: el Cliente le dice quién es

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3566	2352.947880	192.168.178.81	192.168.178.80	SMTP	139	S: 220 servidor-VirtualBox ESMTP Exim 4.76 Fri, 15 Jan 2016 04:25:42 +0100		
3568	2352.955091	192.168.178.80	192.168.178.81	SMTP	89	C: EHLO [192.168.178.80]		
3570	2352.956216	192.168.178.81	192.168.178.80	SMTP	176	S: 250-servidor-VirtualBox Hello [192.168.178.80] [192.168.178.80] 250-SI		
3571	2352.957290	192.168.178.80	192.168.178.81	SMTP	103	C: MAIL FROM:<usuario2@st.um> SIZE=442		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74	S: 250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92	C: RCPT TO:<usuario1@st.um>		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80	S: 250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72	C: DATA		
► Frame 3572: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)								
► Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)								
► Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)								
► Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 184, Ack: 61, Len: 8								
▼ Simple Mail Transfer Protocol								
▼ Response: 250 OK\r\n								
Response code: Requested mail action okay, completed (250)								
Response parameter: OK								

Ilustración 23: el Servidor responde OK 250

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74 S:	250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92 C:	RCPT TO:<usuario1@st.um>		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA		
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself		
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes		

▶ Frame 3573: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 53572 (53572), Dst Port: smtp (25), Seq: 61, Ack: 192, Len: 26
 ▶ Simple Mail Transfer Protocol
 ▶ Command: RCPT TO:<usuario1@st.um>\r\n
 Command: RCPT
 Request parameter: TO:<usuario1@st.um>

Ilustración 24: el Cliente indica para quién es el envío

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3572	2352.957698	192.168.178.81	192.168.178.80	SMTP	74 S:	250 OK		
3573	2352.971129	192.168.178.80	192.168.178.81	SMTP	92 C:	RCPT TO:<usuario1@st.um>		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA		
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself		
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes		

▶ Frame 3574: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 192, Ack: 87, Len: 14
 ▶ Simple Mail Transfer Protocol
 ▶ Response: 250 Accepted\r\n
 Response code: Requested mail action okay, completed (250)
 Response parameter: Accepted

Ilustración 25: el Servidor acepta el envío

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA		
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself		
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes		

▶ Frame 3575: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
 ▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 53572 (53572), Dst Port: smtp (25), Seq: 87, Ack: 206, Len: 6
 ▶ Simple Mail Transfer Protocol
 ▶ Command: DATA\r\n
 Command: DATA

Ilustración 26: el Cliente indica que procede a realizar el envío

Filter: smtp						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted		
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA		
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself		
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes		

▶ Frame 3576: 122 bytes on wire (976 bits), 122 bytes captured (976 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 206, Ack: 93, Len: 56
 ▶ Simple Mail Transfer Protocol
 ▶ Response: 354 Enter message, ending with "." on a line by itself\r\n
 Response code: Start mail input; end with <CRLF>.<CRLF> (354)
 Response parameter: Enter message, ending with "." on a line by itself

Ilustración 27: el Servidor responde

Filter: **smtp** Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from:	Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJv1C-0001C8-L8

Frame 3577: 508 bytes on wire (4064 bits), 508 bytes captured (4064 bits)
Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
Transmission Control Protocol, Src Port: 53572 (53572), Dst Port: smtp (25), Seq: 93, Ack: 262, Len: 442
Simple Mail Transfer Protocol
Reassembled DATA in frame: 3578

Filter: **smtp** Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
3574	2352.972054	192.168.178.81	192.168.178.80	SMTP	80 S:	250 Accepted
3575	2352.972307	192.168.178.80	192.168.178.81	SMTP	72 C:	DATA
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from:	Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJv1C-0001C8-L8

Frame 3578: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)
Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
Transmission Control Protocol, Src Port: 53572 (53572), Dst Port: smtp (25), Seq: 535, Ack: 262, Len: 3
Simple Mail Transfer Protocol
C: .
[1 DATA fragment (442 bytes): #3577(442)]
[Frame: 3577, payload: 0-441 (442 bytes)]
[DATA fragment count: 1]
[Reassembled DATA length: 442]
Internet Message Format

Ilustración 28: el Cliente envía las dos partes del mensaje

Filter: **smtp** Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from:	Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJv1C-0001C8-L8
3581	2353.071756	192.168.178.80	192.168.178.81	SMTP	72 C:	QUIT
3582	2353.072451	192.168.178.81	192.168.178.80	SMTP	110 S:	221 servidor-VirtualBox closing connection

Frame 3580: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 262, Ack: 538, Len: 28
Simple Mail Transfer Protocol
Response: 250 OK id=laJv1C-0001C8-L8\r\n
Response code: Requested mail action okay, completed (250)
Response parameter: OK id=laJv1C-0001C8-L8

Ilustración 29: el Servidor responde

Filter: **smtp** Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from:	Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJv1C-0001C8-L8
3581	2353.071756	192.168.178.80	192.168.178.81	SMTP	72 C:	QUIT
3582	2353.072451	192.168.178.81	192.168.178.80	SMTP	110 S:	221 servidor-VirtualBox closing connection

Frame 3581: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
Transmission Control Protocol, Src Port: 53572 (53572), Dst Port: smtp (25), Seq: 538, Ack: 290, Len: 6
Simple Mail Transfer Protocol
Command: QUIT\r\n
Command: QUIT

Ilustración 30: el Cliente cierra la conexión

Filter: smtp					Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info	
3576	2352.972645	192.168.178.81	192.168.178.80	SMTP	122 S:	354 Enter message, ending with "." on a line by itself	
3577	2352.973726	192.168.178.80	192.168.178.81	SMTP	508 C:	DATA fragment, 442 bytes	
3578	2352.973906	192.168.178.80	192.168.178.81	IMF	69 from: Usuario2 <usuario2@st.um>, subject: Probando Wireshark, (text/plain)		
3580	2353.046119	192.168.178.81	192.168.178.80	SMTP	94 S:	250 OK id=laJvIC-0001C8-L8	
3581	2353.071756	192.168.178.80	192.168.178.81	SMTP	72 C:	QUIT	
3582	2353.072451	192.168.178.81	192.168.178.80	SMTP	110 S:	221 servidor-VirtualBox closing connection	

Frame 3582: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: smtp (25), Dst Port: 53572 (53572), Seq: 290, Ack: 544, Len: 44
 ▶ Simple Mail Transfer Protocol
 ▶ Response: 221 servidor-VirtualBox closing connection\r\n
 Response code: <domain> Service closing transmission channel (221)
 Response parameter: servidor-VirtualBox closing connection

Ilustración 31: el Servidor responde al cierre de conexión

2º POP

Filter: pop					Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info	
7382	7104.023295	192.168.178.81	192.168.178.80	POP	86 S:	+OK Dovecot ready.	
7384	7104.023928	192.168.178.80	192.168.178.81	POP	72 C:	CAPA	
7386	7104.024715	192.168.178.81	192.168.178.80	POP	139 S:	+OK	
7387	7104.025051	192.168.178.80	192.168.178.81	POP	78 C:	AUTH PLAIN	
7388	7104.025972	192.168.178.81	192.168.178.80	IMF	70 +		
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96 C:	AHVzdWFyaW8xAG15cGFzc3dvcmQ=	

Frame 7382: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 58683 (58683), Seq: 1, Ack: 1, Len: 20
 ▶ Post Office Protocol
 ▶ +OK Dovecot ready.\r\n

Ilustración 32: el Servidor se presenta

Filter: pop					Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info	
7382	7104.023295	192.168.178.81	192.168.178.80	POP	86 S:	+OK Dovecot ready.	
7384	7104.023928	192.168.178.80	192.168.178.81	POP	72 C:	CAPA	
7386	7104.024715	192.168.178.81	192.168.178.80	POP	139 S:	+OK	
7387	7104.025051	192.168.178.80	192.168.178.81	POP	78 C:	AUTH PLAIN	
7388	7104.025972	192.168.178.81	192.168.178.80	IMF	70 +		
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96 C:	AHVzdWFyaW8xAG15cGFzc3dvcmQ=	

Frame 7384: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
 ▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 1, Ack: 21, Len: 6
 ▶ Post Office Protocol
 ▶ CAPA\r\n

Ilustración 33: el Cliente solicita lista de Capacidades

Filter: pop					Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info	
7382	7104.023295	192.168.178.81	192.168.178.80	POP	86 S:	+OK Dovecot ready.	
7384	7104.023928	192.168.178.80	192.168.178.81	POP	72 C:	CAPA	
7386	7104.024715	192.168.178.81	192.168.178.80	POP	139 S:	+OK	
7387	7104.025051	192.168.178.80	192.168.178.81	POP	78 C:	AUTH PLAIN	
7388	7104.025972	192.168.178.81	192.168.178.80	IMF	70 +		
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96 C:	AHVzdWFyaW8xAG15cGFzc3dvcmQ=	

Frame 7386: 139 bytes on wire (1112 bits), 139 bytes captured (1112 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 58683 (58683), Seq: 21, Ack: 7, Len: 73
 ▶ Post Office Protocol
 ▶ +OK\r\n
 CAPA\r\n
 TOP\r\n
 UIDL\r\n
 RESP-CODES\r\n
 PIPELINING\r\n
 STLS\r\n
 USER\r\n
 SASL PLAIN\r\n
 .\r\n

Ilustración 34: el Servidor responde OK

Filter: pop							Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info			
7382	7104.023295	192.168.178.81	192.168.178.80	POP	86	S: +OK Dovecot ready.			
7384	7104.023928	192.168.178.80	192.168.178.81	POP	72	C: CAPA			
7386	7104.024715	192.168.178.81	192.168.178.80	POP	139	S: +OK			
7387	7104.025051	192.168.178.80	192.168.178.81	POP	78	C: AUTH PLAIN			
7388	7104.025972	192.168.178.81	192.168.178.80	IMF	70	+			
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96	C: AHVzdWFyaW8xAG15cGFzc3dvcnQ=			
▶ Frame 7387: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)									
▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)									
▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)									
▶ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 7, Ack: 94, Len: 12									
▼ Post Office Protocol									
▼ AUTH PLAIN\r\n									
Request command: AUTH									
Request parameter: PLAIN									
Filter: pop							Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info			
7382	7104.023295	192.168.178.81	192.168.178.80	POP	86	S: +OK Dovecot ready.			
7384	7104.023928	192.168.178.80	192.168.178.81	POP	72	C: CAPA			
7386	7104.024715	192.168.178.81	192.168.178.80	POP	139	S: +OK			
7387	7104.025051	192.168.178.80	192.168.178.81	POP	78	C: AUTH PLAIN			
7388	7104.025972	192.168.178.81	192.168.178.80	IMF	70	+			
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96	C: AHVzdWFyaW8xAG15cGFzc3dvcnQ=			
▶ Frame 7389: 96 bytes on wire (768 bits), 96 bytes captured (768 bits)									
▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)									
▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)									
▶ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 19, Ack: 98, Len: 30									
▼ Post Office Protocol									
▼ AHVzdWFyaW8xAG15cGFzc3dvcnQ=\r\n									
Request command: AHVzdWFyaW8xAG15cGFzc3dvcnQ=									

Ilustración 35: el Cliente se autentica

Filter: pop							Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info			
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96	C: AHVzdWFyaW8xAG15cGFzc3dvcnQ=			
7390	7104.047915	192.168.178.81	192.168.178.80	POP	82	S: +OK Logged in.			
7391	7104.048389	192.168.178.80	192.168.178.81	POP	72	C: STAT			
7392	7104.048813	192.168.178.81	192.168.178.80	POP	78	S: +OK 4 4138			
7393	7104.050560	192.168.178.80	192.168.178.81	POP	72	C: LIST			
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:			
▶ Frame 7390: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)									
▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)									
▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)									
▶ Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 58683 (58683), Seq: 98, Ack: 49, Len: 16									
▼ Post Office Protocol									
▼ +OK Logged in.\r\n									
Response indicator: +OK									
Response description: Logged in.									

Ilustración 36: el Servidor da el OK al Cliente autenticado

Filter: pop							Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info			
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96	C: AHVzdWFyaW8xAG15cGFzc3dvcnQ=			
7390	7104.047915	192.168.178.81	192.168.178.80	POP	82	S: +OK Logged in.			
7391	7104.048389	192.168.178.80	192.168.178.81	POP	72	C: STAT			
7392	7104.048813	192.168.178.81	192.168.178.80	POP	78	S: +OK 4 4138			
7393	7104.050560	192.168.178.80	192.168.178.81	POP	72	C: LIST			
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:			
▶ Frame 7391: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)									
▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)									
▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)									
▶ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 49, Ack: 114, Len: 6									
▼ Post Office Protocol									
▼ STAT\r\n									
Request command: STAT									

Ilustración 37: el Cliente ejecuta STAT

Filter: pop						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96	C: AHVzdWFyaW8xAG15cGFzc3dvcmQ=		
7390	7104.047915	192.168.178.81	192.168.178.80	POP	82	S: +OK Logged in.		
7391	7104.048389	192.168.178.80	192.168.178.81	POP	72	C: STAT		
7392	7104.048813	192.168.178.81	192.168.178.80	POP	78	S: +OK 4 4138		
7393	7104.050560	192.168.178.80	192.168.178.81	POP	72	C: LIST		
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:		

▶ Frame 7392: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 58683 (58683), Seq: 114, Ack: 55, Len: 12
 ▶ Post Office Protocol
 ▼ +OK 4 4138\r\n
 Response indicator: +OK
 Response description: 4 4138

Ilustración 38: el Servidor responde OK

Filter: pop						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
7389	7104.026341	192.168.178.80	192.168.178.81	POP	96	C: AHVzdWFyaW8xAG15cGFzc3dvcmQ=		
7390	7104.047915	192.168.178.81	192.168.178.80	POP	82	S: +OK Logged in.		
7391	7104.048389	192.168.178.80	192.168.178.81	POP	72	C: STAT		
7392	7104.048813	192.168.178.81	192.168.178.80	POP	78	S: +OK 4 4138		
7393	7104.050560	192.168.178.80	192.168.178.81	POP	72	C: LIST		
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:		

▶ Frame 7393: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
 ▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 55, Ack: 126, Len: 6
 ▶ Post Office Protocol
 ▼ LIST\r\n
 Request command: LIST

Ilustración 39: el Cliente hace LIST

Filter: pop						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
7392	7104.048813	192.168.178.81	192.168.178.80	POP	78	S: +OK 4 4138		
7393	7104.050560	192.168.178.80	192.168.178.81	POP	72	C: LIST		
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:		
7395	7104.051406	192.168.178.80	192.168.178.81	POP	72	C: UIDL		
7396	7104.051893	192.168.178.81	192.168.178.80	POP	154	S: +OK		
7397	7104.053974	192.168.178.80	192.168.178.81	POP	72	C: QUIT		
7398	7104.055226	192.168.178.81	192.168.178.80	POP	84	S: +OK Logging out.		

▶ Frame 7394: 115 bytes on wire (920 bits), 115 bytes captured (920 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 58683 (58683), Seq: 126, Ack: 61, Len: 49
 ▶ Post Office Protocol
 ▼ +OK 4 messages:\r\n
 Response indicator: +OK
 Response description: 4 messages:
 1 667\r\n
 2 1995\r\n
 3 734\r\n
 4 742\r\n
 .\r\n

Ilustración 40: el Servidor lista cuatro mensajes

Filter: pop						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info		
7392	7104.048813	192.168.178.81	192.168.178.80	POP	78	S: +OK 4 4138		
7393	7104.050560	192.168.178.80	192.168.178.81	POP	72	C: LIST		
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:		
7395	7104.051406	192.168.178.80	192.168.178.81	POP	72	C: UIDL		
7396	7104.051893	192.168.178.81	192.168.178.80	POP	154	S: +OK		
7397	7104.053974	192.168.178.80	192.168.178.81	POP	72	C: QUIT		
7398	7104.055226	192.168.178.81	192.168.178.80	POP	84	S: +OK Logging out.		

▶ Frame 7395: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
 ▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 61, Ack: 175, Len: 6
 ▶ Post Office Protocol
 ▼ UIDL\r\n
 Request command: UIDL

Ilustración 41: el Cliente hace UIDL

Filter: pop				Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:
7395	7104.051406	192.168.178.80	192.168.178.81	POP	72	C: UIDL
7396	7104.051893	192.168.178.81	192.168.178.80	POP	154	S: +OK
7397	7104.053974	192.168.178.80	192.168.178.81	POP	72	C: QUIT
7398	7104.055226	192.168.178.81	192.168.178.80	POP	84	S: +OK Logging out.
7731	7625.975722	192.168.178.81	192.168.178.80	POP	86	S: +OK Dovecot ready.

▷ Frame 7397: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
 ▷ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▷ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▷ Transmission Control Protocol, Src Port: 58683 (58683), Dst Port: pop3 (110), Seq: 67, Ack: 263, Len: 6
 ▷ Post Office Protocol
 ▷ QUIT\r\n
 Request command: QUIT

Ilustración 42: el Cliente cierra la conexión

Filter: pop				Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info
7394	7104.051119	192.168.178.81	192.168.178.80	POP	115	S: +OK 4 messages:
7395	7104.051406	192.168.178.80	192.168.178.81	POP	72	C: UIDL
7396	7104.051893	192.168.178.81	192.168.178.80	POP	154	S: +OK
7397	7104.053974	192.168.178.80	192.168.178.81	POP	72	C: QUIT
7398	7104.055226	192.168.178.81	192.168.178.80	POP	84	S: +OK Logging out.
7731	7625.975722	192.168.178.81	192.168.178.80	POP	86	S: +OK Dovecot ready.

▷ Frame 7398: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
 ▷ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▷ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▷ Transmission Control Protocol, Src Port: pop3 (110), Dst Port: 58683 (58683), Seq: 263, Ack: 73, Len: 18
 ▷ Post Office Protocol
 ▷ +OK Logging out.\r\n
 Response indicator: +OK
 Response description: Logging out.

Ilustración 43: el Servidor confirma el cierre de la conexión

3.3 Desplegar Servicios HTTP/HTTPS

Para desplegar los servicios HTTP/HTTPS se ha desarrollado con dos servicios completamente independientes, uno tomando como base **Apache HTTP(S) Server** y otro implementando el servidor en C, teniendo como base el fichero que nos ha sido proporcionado por los profesores de la asignatura (**web_ssst.c**).

3.3.1 Apache HTTP(S) Server

Para poder implementar el servicio web mediante Apache hemos tenido que instalar Apache2 Webserver. Como deseamos que nuestro servidor sea capaz de interpretar el lenguaje Script PHP, se ha instalado de forma conjunta el módulo de PHP5, de esta forma nuestro servidor es capaz de mostrar/interpretar webs (y aplicaciones web) en dicho lenguaje.

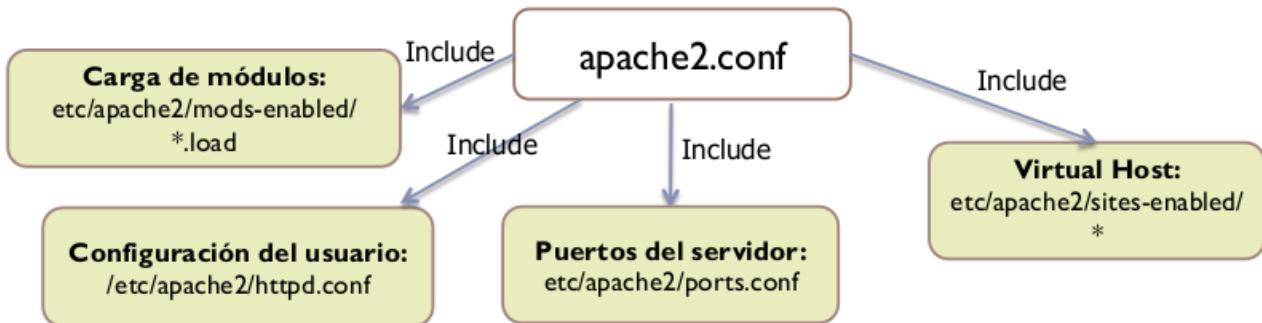


Ilustración 44: Estructura de configuración de Apache2

- Para la instalación de Apache2 hemos requerido del siguiente comando:

```
sudo apt-get install apache2
```

- Para incorporar el módulo de PHP5 es necesario ejecutar el siguiente comando en una Terminal:

```
sudo apt-get install php5
```

- Para que funcione PHP ahora tendremos que reiniciar el servidor Apache y para ello podemos ejecutar el siguiente comando:

```
sudo /etc/init.d/apache2 restart
```

- También podemos hacerlo con el siguiente:

```
sudo service apache2 restart
```

- Los diferentes comandos de control básicos de Apache2 son los siguientes:

```
sudo /etc/init.d/apache2 start
sudo /etc/init.d/apache2 stop
sudo /etc/init.d/apache2 restart

sudo service apache2 start
sudo service apache2 stop
sudo service apache2 restart
```

Apache2 emplea lo que se conoce como **Virtual Host**. Tal y como se puede deducir por su nombre, apache dispone de la capacidad de mantener varios servidores virtuales distintos en una sola máquina. La forma de distinguirlos es a través de:

- Nombre de la máquina.
- Dirección IP.

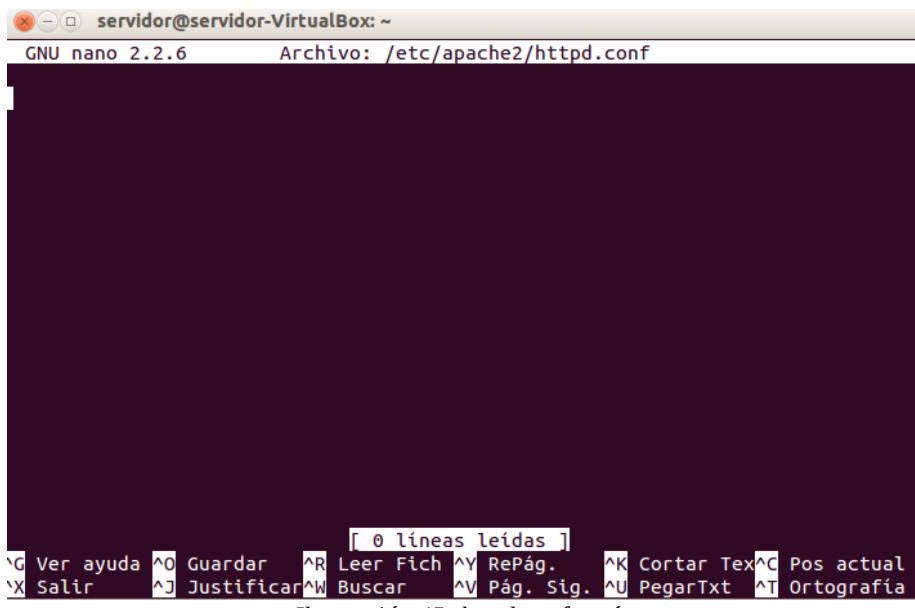
Aún estando los servidores de forma conjunta en una misma máquina, para el cliente es totalmente transparente.

En Ubuntu, se ha pensado que se puedan tener distintas configuraciones posibles de Virtual Host:

- Teniendo diferentes ficheros en *sites-available*.
- Estarán activos aquellos **enlazados** en *sites-enabled*.

Para esta práctica final se desea contar con la utilización de Virtual Hosts. Para configurar Apache2 para la utilización de Virtual Hosts tenemos que tener habilitada la opción, y ello se consigue con una serie de pasos:

1. Debemos de asegurarnos de que el fichero **httpd.conf** está vacío.

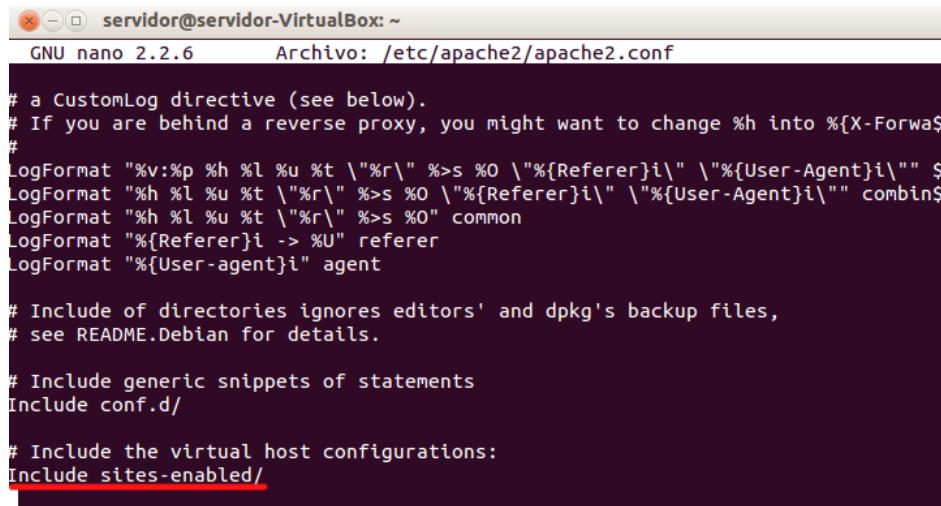


The screenshot shows a terminal window titled "servidor@servidor-VirtualBox: ~". The title bar also displays "GNU nano 2.2.6" and "Archivo: /etc/apache2/httpd.conf". The main area of the terminal is a dark grey rectangle, indicating that the file is empty. At the bottom of the terminal window, there is a status bar with the following text: "[0 líneas leídas]" (0 lines read), followed by a series of keyboard shortcuts: '^G Ver ayuda', '^O Guardar', '^R Leer Fich.', '^Y RePág.', '^K Cortar Tex', '^C Pos actual', '^X Salir', '^J Justificar', '^W Buscar', '^V Pág. Sig.', '^U PegarTxt', and '^T Ortografía'.

Ilustración 45: *httpd.conf vacío*

2. En el fichero **apache2.conf** tiene que contener la línea:

```
Include /etc/apache2/sites-enabled/
```



```
servidor@servidor-VirtualBox: ~
GNU nano 2.2.6      Archivo: /etc/apache2/apache2.conf

# a CustomLog directive (see below).
# If you are behind a reverse proxy, you might want to change %h into %{X-Forwarded-For}i
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\""
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-Agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

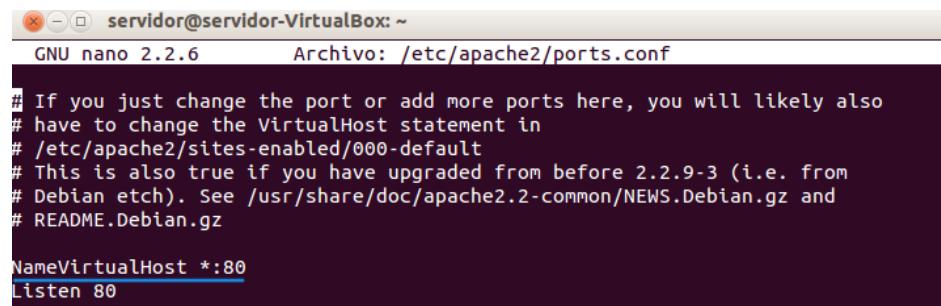
# Include generic snippets of statements
Include conf.d/

# Include the virtual host configurations:
Include sites-enabled/
```

Ilustración 46: apache2.conf

3. Tenemos que tener habilitada la opción de Virtual Hosts en el fichero “/etc/apache2/ports.conf”.

- Ha de contener la linea NameVirtualHost *:80



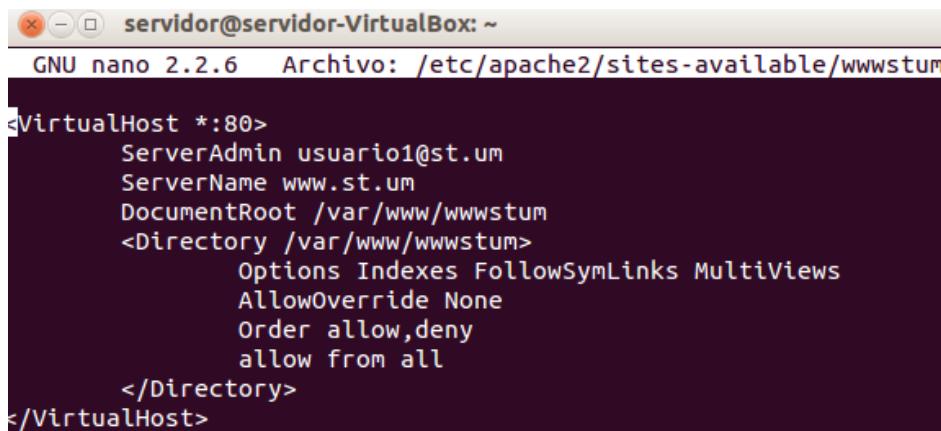
```
servidor@servidor-VirtualBox: ~
GNU nano 2.2.6      Archivo: /etc/apache2/ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default
# This is also true if you have upgraded from before 2.2.9-3 (i.e. from
# Debian etch). See /usr/share/doc/apache2.2-common/NEWS.Debian.gz and
# README.Debian.gz

NameVirtualHost *:80
Listen 80
```

Ilustración 47: ports.conf

4. Por cada Virtual Host que queramos usar tendremos que crear un fichero de configuración correspondiente.



```
servidor@servidor-VirtualBox: ~
GNU nano 2.2.6      Archivo: /etc/apache2/sites-available/wwwstum

<VirtualHost *:80>
    ServerAdmin usuario1@st.um
    ServerName www.st.um
    DocumentRoot /var/www/wwwstum
    <Directory /var/www/wwwstum>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
</VirtualHost>
```

Ilustración 48: fichero de configuración de www.st.um

5. Tendremos que activar el Virtual Host. `sudo a2ensite [nombre_fichero_virtualHost]`

```
sudo a2ensite wwwstum
```

NOTA: para desactivarlo: `sudo a2dissite [nombre_fichero_virtualHost]`.

6. Ahora tendremos que desactivar el sitio por defecto:

```
sudo a2dissite default
```

7. Ahora tenemos que reiniciar/recargar el servicio del Servidor Apache:

```
sudo service apache2 reload / sudo service apache2 restart
```

8. Las direcciones del virtual Host deben ser resolubles. Para ello tendremos que dar de alta en el servidor DNS las direcciones que queremos emplear (cuando se realiza este trabajo sin servidor DNS lo hacemos en “/etc/hosts”). En nuestro caso www.st.um, pero contamos con varios ejemplos más.

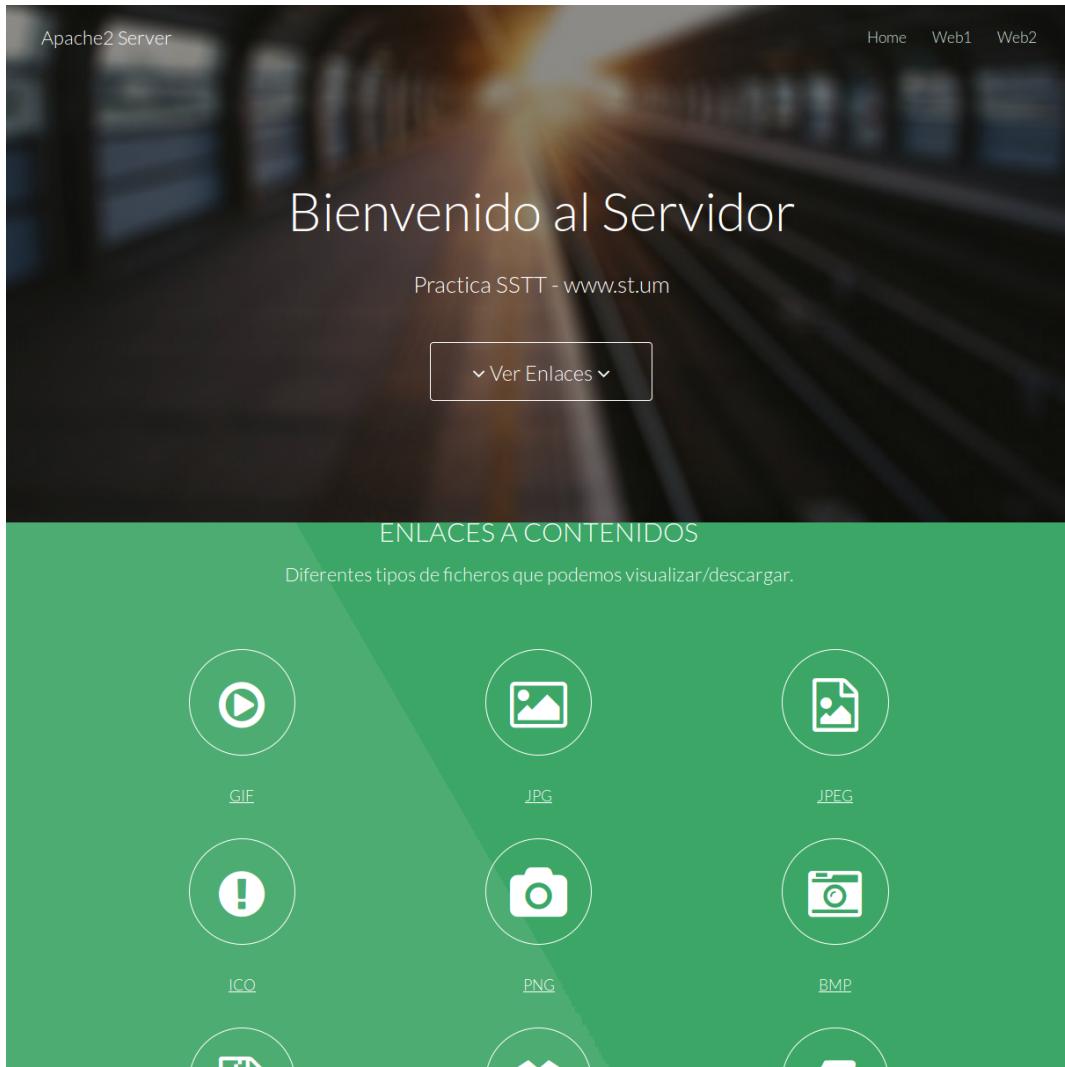


Ilustración 49: www.st.um

3.3.1.1 Autenticación de usuario HTTP básica

Puesto que este aspecto ha sido desarrollado en las sesiones prácticas, se ha considerado de interés para añadir de forma adicional a esta práctica final. Vamos a restringir el acceso para el usuario “usuario1” con autenticación basada en texto plano. Para ello vamos a crear un hash de una clave y se va a guardar en un archivo que llamaremos **passwords**.

Paso1:

Tenemos que introducir el siguiente comando en una Terminal:

```
htpasswd -c /etc/apache2/passwords usuario1
```

Nos solicitará una nueva contraseña y volver a introducirla. Hemos empleado la contraseña “**mypassword**”.

```
# htpasswd -c /etc/apache2/passwords usuario1
      New password: mypassword
      Re-type new password: mypassword
      Adding password for user usuario1
```

Ilustración 50: creando el fichero de contraseña

NOTA: para añadir nuevos usuarios es el mismo comando pero sin la opción -c.

Paso2:

Tras haber generado el fichero de contraseñas, tenemos que realizar modificaciones en el fichero de configuración del Virtual Host correspondiente para que utilice la autenticación para el usuario1 que hemos generado. Hay que dar de nuevo de alta el fichero wwwstum y reiniciar el servicio de Apache.

```
<VirtualHost *:80>
    ServerAdmin usuario1@st.um
    ServerName www.st.um
    DocumentRoot /var/www/wwwstum
    <Directory /var/www/wwwstum>
        AllowOverride AuthConfig
        AuthType Basic
        AuthName "Acceso restringido a usuario1"
        AuthBasicProvider file
        AuthUserFile /etc/apache2/passwords
        Require user usuario1
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
</VirtualHost>
```

Ilustración 51: fichero de configuración modificado

- Podemos observar que lo que se ha añadido es lo siguiente:

```
AllowOverride AuthConfig
AuthType Basic
AuthName "Acceso restringido a usuario1"
AuthBasicProvider file
AuthUserFile /etc/apache2/passwords
Require user usuario1
```

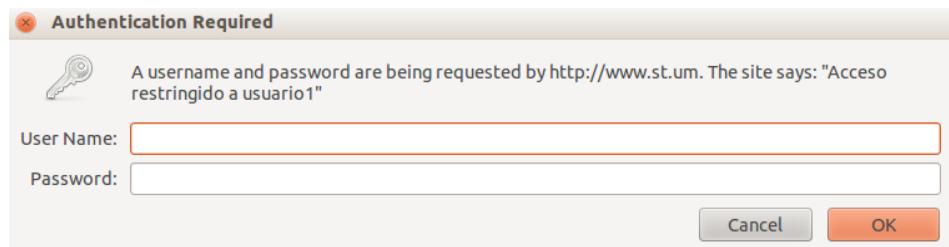


Ilustración 52: mensaje de autenticación

3.3.1.2 Trazas en Wireshark

A continuación se van a mostrar las trazas que hemos obtenido con Wireshark.

Filter: http

No.	Time	Source	Destination	Protocol	Length	Info
14	4.576810	192.168.178.80	91.189.89.88	HTTP	472	GET /12.04/Google/?sourceid=hp HTTP/1.1
16	4.630150	91.189.89.88	192.168.178.80	HTTP	408	HTTP/1.0 304 Not Modified
27	6.552798	192.168.178.1	239.255.255.250	SSDP	165	M-SEARCH * HTTP/1.1
28	6.553650	fe80::a96:d7ff:feee:b1ff02::c		SSDP	179	M-SEARCH * HTTP/1.1
39	10.179756	192.168.178.80	192.168.178.81	HTTP	445	GET / HTTP/1.1
41	10.180580	192.168.178.81	192.168.178.80	HTTP	745	HTTP/1.1 401 Authorization Required (text/html)
75	22.765570	192.168.178.80	192.168.178.81	HTTP	496	GET / HTTP/1.1
78	22.766983	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified
80	22.900983	192.168.178.80	192.168.178.81	HTTP	481	GET /js/jquery.min.js HTTP/1.1
84	22.902066	192.168.178.80	192.168.178.81	HTTP	478	GET /js/skel.min.js HTTP/1.1
90	22.904072	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified
92	22.904092	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified

▼ Hypertext Transfer Protocol
► HTTP/1.1 401 Authorization Required\r\nDate: Fri, 15 Jan 2016 05:27:33 GMT\r\nServer: Apache/2.2.22 (Ubuntu)\r\nWWW-Authenticate: Basic realm="Acceso restringido a usuario1"\r\nVary: Accept-Encoding\r\nContent-Encoding: gzip\r\nContent-Length: 339\r\nKeep-Alive: timeout=5, max=100\r\nConnection: Keep-Alive\r\nContent-Type: text/html; charset=iso-8859-1\r\n

Ilustración 53: el Servidor solicita autenticación

Filter: http

No.	Time	Source	Destination	Protocol	Length	Info
14	4.576810	192.168.178.80	91.189.89.88	HTTP	472	GET /12.04/Google/?sourceid=hp HTTP/1.1
16	4.630150	91.189.89.88	192.168.178.80	HTTP	408	HTTP/1.0 304 Not Modified
27	6.552798	192.168.178.1	239.255.255.250	SSDP	165	M-SEARCH * HTTP/1.1
28	6.553650	fe80::a96:d7ff:feee:b1ff02::c		SSDP	179	M-SEARCH * HTTP/1.1
39	10.179756	192.168.178.80	192.168.178.81	HTTP	445	GET / HTTP/1.1
41	10.180580	192.168.178.81	192.168.178.80	HTTP	745	HTTP/1.1 401 Authorization Required (text/html)
75	22.765570	192.168.178.80	192.168.178.81	HTTP	496	GET / HTTP/1.1
78	22.766983	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified
80	22.900983	192.168.178.80	192.168.178.81	HTTP	481	GET /js/jquery.min.js HTTP/1.1
84	22.902066	192.168.178.80	192.168.178.81	HTTP	478	GET /js/skel.min.js HTTP/1.1
90	22.904072	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified
92	22.904092	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified

▼ Hypertext Transfer Protocol
► GET / HTTP/1.1\r\nHost: www.st.um\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\nIf-Modified-Since: Mon, 04 Jan 2016 02:39:17 GMT\r\nIf-None-Match: "b95e-11b6-528790709d966"\r\nAuthorization: Basic dXNlYXlnb2FhbXlwYXNzd29vZA==\r\n

Ilustración 54: el Cliente responde a la solicitud de autorización

No.	Time	Source	Destination	Protocol	Length	Info
14	4.576810	192.168.178.80	91.189.89.88	HTTP	472	GET /12.04/Google/?sourceid=hp HTTP/1.1
16	4.630150	91.189.89.88	192.168.178.80	HTTP	408	HTTP/1.0 304 Not Modified
27	6.552798	192.168.178.1	239.255.255.250	SSDP	165	M-SEARCH * HTTP/1.1
28	6.553650	fe80:::96:d7ff:feee:bff02::c		SSDP	179	M-SEARCH * HTTP/1.1
39	10.179756	192.168.178.80	192.168.178.81	HTTP	445	GET / HTTP/1.1
41	10.180588	192.168.178.81	192.168.178.80	HTTP	745	HTTP/1.1 401 Authorization Required (text/html)
75	22.765570	192.168.178.80	192.168.178.81	HTTP	496	GET / HTTP/1.1
78	22.766983	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified
80	22.900983	192.168.178.80	192.168.178.81	HTTP	481	GET /js/jquery.min.js HTTP/1.1
84	22.902066	192.168.178.80	192.168.178.81	HTTP	478	GET /js/skel.min.js HTTP/1.1
90	22.904072	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified
92	22.904092	192.168.178.81	192.168.178.80	HTTP	276	HTTP/1.1 304 Not Modified

▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: http (80), Dst Port: 40434 (40434), Seq: 1, Ack: 431, Len: 210
 ▶ Hypertext Transfer Protocol
 ▶ HTTP/1.1 304 Not Modified
 Date: Fri, 15 Jan 2016 05:27:46 GMT\r\n
 Server: Apache/2.2.22 (Ubuntu)\r\n
 Connection: Keep-Alive\r\n
 Keep-Alive: timeout=5, max=100\r\n
 ETag: "b95e-11b6-528790709d96"\r\n
 Vary: Accept-Encoding\r\n
 \r\n

Ilustración 55: el Servidor indica que la página no fue modificada desde última consulta

3.3.2 Servicio HTTPS

Para este servicio deseamos que se emplee autenticación de cliente basada en certificados X.509. Como se desea realizar una consulta a la página desde el navegador con “<https://www.st.um>” tenemos que crear un nuevo fichero de configuración para el Virtual Host (/etc/apache2/sites-available/httpswwwstum).

- Pero tenemos que asegurarnos que el nuevo fichero de configuración no cuenta con la configuración hecha en el apartado anterior de Autenticación de Usuario Básica.

```
GNU nano 2.2.6                               Archivo: /etc/apache2/sites-available/httpswwwstum

VirtualHost *:443>
    ServerAdmin usuario1@st.um
    ServerName www.st.um
    DocumentRoot /var/www/wwwstum
    <Directory /var/www/wwwstum>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
    SSLEngine on
        SSLCertificateFile      /home/servidor/demoCA/servercert.pem
        SSLCertificateKeyFile   /home/servidor/demoCA/serverkey.pem
        SSLCACertificateFile    /home/servidor/demoCA/cacert.pem
        SSLVerifyClient         require
        SSLVerifyDepth          1
            <FilesMatch "\.(cgi|shtml|phtml|php)$">
                SSLOptions +StdEnvVars
            </FilesMatch>
        BrowserMatch "MSIE [2-6]" \
            nokeepalive ssl-unclean-shutdown \
            downgrade-1.0 force-response-1.0
        #MSIE 7 and newer should be able to use keepalive
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
```

Ilustración 56: fichero de configuración httpswwwstum

Para la comunicación segura (https) entre Cliente y Servidor vamos a emplear **OpenSSL**. Se trata de la librería criptográfica más popular. En 1995 Eric A. Young y Tim J. Hudson crean SSLeay, para posteriormente en 1998 llamarse OpenSSL. Es una librería de código abierto que incluye:

- **libcrypto**: librería de funciones criptográficas.
- **libssl**: librería para la gestión del protocolo SSL.

Esta librería incluye la implementación completa de SSLv2, SSLv3 y TLSv1. SSL ofrece autenticación y privacidad basados en la **autenticación con certificados digitales**. En nuestro caso, se va a usar la autenticación basada en **certificados x.509**.

A la hora de comenzar con la correcta implantación de OpenSSL en nuestro sistema debemos de comentar con la configuración de la Autoridad de Certificación (CA). Para ello, como paso inicial, debemos crear un árbol de directorio concreto y de modificar el fichero /usr/lib/ssl/**openssl.cfg**.

OpenSSL cuenta con una configuración CA por defecto. Tendremos que modificar dicha configuración para adaptarla a nuestras necesidades.

```

▶ Configuración CA por defecto: sección [ CA_default ]
  ▶ dir           = ./demoCA ← Actualizar a /home/alumno/demoCA
  ▶ certs         = $dir/certs
  ▶ crl_dir       = $dir/crl
  ▶ database      = $dir/index.txt
  ▶ new_certs_dir = $dir/newcerts
  ▶ certificate   = $dir/cacert.pem
  ▶ serial        = $dir/serial
  ▶ crlnumber     = $dir/crlnumber
  ▶ crl           = $dir/crl.pem
  ▶ private_key   = $dir/private/cakey.pem
  ▶ default_days  = 365
  ▶ default_crl_days = 30

```

Ilustración 57: configuración CA por defecto

Para poder realizar la modificación tendremos que realizar, en primera instancia , el siguiente *árbol de directorios* (con el aspecto que tiene un directorio de CA de OpenSSL):

Generamos el **directorio demoCA** dentro de la carpeta de nuestro usuario (en nuestro caso es servidor):

```
mkdir /home/servidor/demoCA
```

Nos situamos dentro del directorio que hemos creado:

```
cd /home/servidor/demoCA
```

Una vez dentro, necesitamos el archivo **crlnumber**. Este ha de contener en su interior 00. Para ello ejecutamos:

```
echo 0 > crlnumber
```

Necesitamos que **index.txt** esté vacío:

```
touch index.txt
```

Ahora es necesario un fichero **serial** que contenga 01:

```
echo 01 > serial
```

Una vez creado estos archivos tendremos que crear las carpetas “**private**”, “**newcerts**” y “**certs**”.

```
mkdir private
mkdir newcerts
mkdir certs
```

```

servidor@servidor-VirtualBox:~$ tree demoCA/
demoCA/
├── certs
├── crlnumber
├── index.txt
├── newcerts
└── private
    └── serial

```

Ilustración 58: arbol de directorio demoCA

Una vez creado por completo el anterior árbol de directorio podemos modificar el fichero /usr/lib/ssl/**openssl.cfg**. Las modificaciones las realizaremos en la sección [CA_default]:

- ▶ dir = /home/alumno/demoCA
- ▶ (demás valores por defecto)
- ▶ countryName_default = ES
- ▶ stateOrProvinceName_default = Murcia
- ▶ organizationalName_default = UMU
- ▶ organizationalUnitName_default = SSTT
- ▶ Ojo, por defecto está comentado, descomentar.

Ilustración 59: modificaciones en sección [CA_default]

- **OJO:** en nuestro caso es **dir = /home/servidor/demoCA**.

Una vez configurado tenemos que proceder con la inicialización. Para ello vamos a generar el certificado de la CA (Autoridad de Certificación) y su clave privada (CA genera su propio certificado auto-firmado). Hay que tener en cuenta de forma cuidadosa los valores indicados en el fichero /usr/lib/ssl/**openssl.cfg** y cuando nos solicite Common Name, introducir **ca** y recordar la palabra de paso para usarla posteriormente (hemos usado **palabradepaso**). Tenemos que ejecutar el siguiente comando en una Terminal:

```
openssl req -x509 -newkey rsa:1024 -keyout cakey.pem -out cacert.pem
```

```
servidor@servidor-VirtualBox:~$ openssl req -x509 -newkey rsa:1024 -keyout cakey.pem -out cacert.pem
Generating a 1024 bit RSA private key
.....+++++
.....+++++
unable to write 'random state'
writing new private key to 'cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:ES
State or Province Name (full name) [Murcia]:Murcia
Locality Name (eg, city) []:
Organization Name (eg, company) [UMU]:UMU
Organizational Unit Name (eg, section) [SSTT]:SSTT
Common Name (e.g. server FQDN or YOUR name) []:ca
Email Address []:
```

Ilustración 60: generando el certificado de la CA

- Podemos apreciar entre [] los valores que ya fueron introducidos en el fichero openssl.cfg y a la derecha repetimos los mismos, salvo en Common Name que hemos introducido **ca**.

Podemos comprobar el certificado generado con los siguientes comandos:

```
cat cacert.pem  
cat cakey.pem  
openssl x509 -in cacert.pem -text  
openssl rsa -in cakey.pem -text
```

Ahora tenemos que mover a demoCA/private la clave privada de la CA.

```
mv cakey.pem demoCA/private/
```

En este momento ya disponemos del certificado del CA y su KS. Ya podemos generar el certificado para el Servicio Web (para el **Servidor**). Una vez más debemos de tener cuidado con los valores que introdujimos anteriormente en /usr/lib/ssl/**openssl.cfg**, pero ahora en Common Name introduciremos **www.st.um**. Tenemos que ejecutar el siguiente comando en una Terminal:

```
openssl req -new -nodes -newkey rsa:512 -keyout serverkey.pem -out servercsr.pem
```

```
servidor@servidor-VirtualBox:~$ openssl req -new -nodes -newkey rsa:512 -keyout serverkey.pem -out servercsr.pem  
Generating a 512 bit RSA private key  
.....  
.....  
unable to write 'random state'  
writing new private key to 'serverkey.pem'  
-----  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [ES]:ES  
State or Province Name (full name) [Murcia]:Murcia  
Locality Name (eg, city) []:  
Organization Name (eg, company) [UMU]:UMU  
Organizational Unit Name (eg, section) [SSTT]:SSTT  
Common Name (e.g. server FQDN or YOUR name) []:www.st.um  
Email Address []:  
  
Please enter the following 'extra' attributes  
to be sent with your certificate request  
A challenge password []:  
An optional company name []:
```

Ilustración 61: generando el certificado Servicio Web

- Pedirá un password para proteger la clave privada, **lo dejamos en blanco**.
- Podemos apreciar entre [] los valores que ya fueron introducidos en el fichero **openssl.cfg** y a la derecha repetimos los mismos, salvo en Common Name que hemos introducido **www.st.um**.

Una vez que tenemos generado el certificado que usaremos en el Servicio Web debemos de firmarlo con su KS. Para ello tendremos que situarnos en el directorio de **demoCA** y ejecutar un comando concreto en la Terminal:

```
cd /home/servidor/demoCA  
openssl ca -keyfile private/cakey.pem -in servercsr.pem -out servercert.pem
```

Debemos de asegurarnos de que **cacert.pem**, **servercsr.pem** y **serverkey.pem** se encuentran también en el directorio **demoCA**.

- Pedirá la palabra de paso (**palabradepaso**) para descifrar la clave privada de la CA y poder firmar la solicitud de certificación.
- Pedirá confirmación para firmar el certificado (yes).
- Pedirá confirmación para añadirlo al repositorio interno de certificados (demoCA/newcerts/) (yes).
- Con –days <días> se puede especificar el número de días para los cuales se genera el certificado.

```

servidor@servidor-VirtualBox:~/demoCA$ openssl ca -keyfile private/cakey.pem -in servercsr.pem -out servercert.pem
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for private/cakey.pem:
check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 1 (0x1)
    Validity
        Not Before: Jan 16 17:47:20 2016 GMT
        Not After : Jan 15 17:47:20 2017 GMT
    Subject:
        countryName            = ES
        stateOrProvinceName    = Murcia
        organizationName       = UMU
        organizationalUnitName = SSS
        commonName              = www.st.um
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
        D0:D8:AC:C8:A4:D7:D5:B7:1B:6A:41:94:A7:F0:DE:6D:47:DD:76:A5
X509v3 Authority Key Identifier:
        keyid:0B:AE:F1:AD:7B:7F:E3:F7:45:74:30:5D:E4:63:22:DE:94:6F:24:99
Certificate is to be certified until Jan 15 17:47:20 2017 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]
Write out database with 1 new entries
Data Base Updated
unable to write 'random state'
```

Ilustración 62: certificado generado y firmado

+ Ahora ya está el certificado del servicio web generado y firmado por el CA (autofirmado).

NOTA: como revocar un certificado por si hubiese que repetir el proceso:

```
openssl ca -revoke /etc/ssl/newcerts/certificado.pem (nosotros en demoCA)
```

Probamos el certificado servidor.

Para probarlo podemos utilizar el comando **s_server**:

- Implementa un pequeño servidor seguro SSL/TLS.
- Devuelve información de sesión al navegador con la opción -www .
- Especifica el puerto en el que escucha con la opción -accept (4433 por defecto).

```
openssl s_server -cert servercert.pem -key serverkey.pem -www
```

> Navegador: <https://localhost:4433/>
 > Añadimos excepción con el certificado.

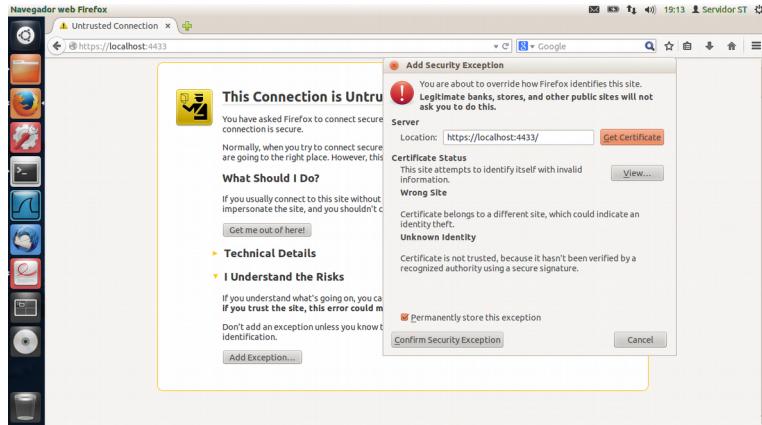


Ilustración 63: mensaje de alerta certificado y añadiendo excepción

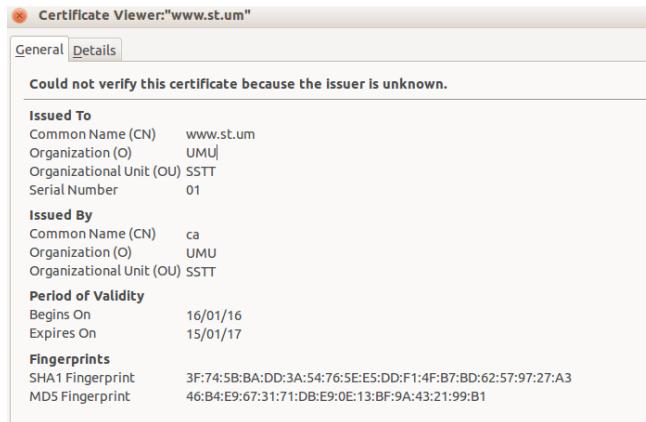


Ilustración 64: datos del certificado

Certificado del cliente.

Debemos generar el certificado del **Cliente**. Para ello tenemos que generar un nuevo par de claves pública/privada para el cliente del mismo modo que para el servidor (hay que repetir todo el proceso anterior, salvo cambiar server por client).

```
openssl req -new -nodes -newkey rsa:512 -keyout clientkey.pem -out clientcsr.pem
```

Ahora tenemos que ejecutar el comando en la Terminal:

```
openssl ca -keyfile private/cakey.pem -in clientcsr.pem -out clientcert.pem
```

+ Una vez repetido este proceso, ya está el certificado firmado por el CA para el cliente, pero tenemos que exportarlo a formato PKCS12. Ejecutar el siguiente comando:

```
openssl pkcs12 -export -in clientcert.pem -certfile cacert.pem -inkey clientkey.pem -out clientcert.pfx
```

- Hemos usado en Email: usuario1@st.um.

Con este comando hemos generado `clientcert.pfx`. Este certificado cuenta con la KS y el certificado de la CA. Ahora se tiene que importar en el navegador del cliente el certificado en formato PKCS12 y ya el usuario se autenticará cuando sea necesario.

- Para **Importar el Certificado** tenemos que ir a Menú >> Preferencias >> Avanzado >> Certificados >> Ver Certificados >> Tus Certificados >> Importar >> Seleccionamos el fichero.

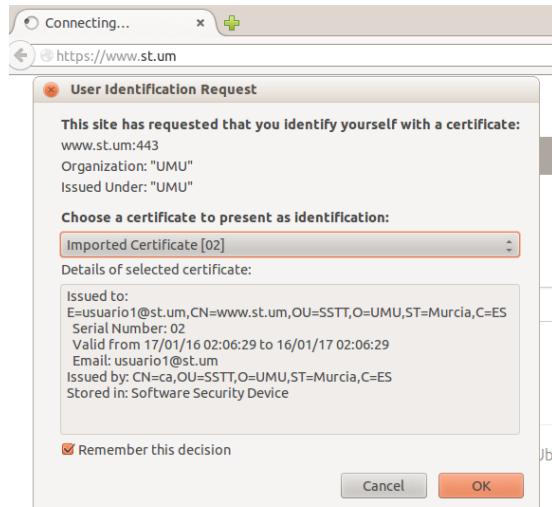


Ilustración 65: solicitud autenticación certificado

Configuración de ApacheSSL.

Una vez hecha la prueba manual podemos proceder a la configuración de **ApacheSSL**. Como ya disponemos del certificado generado y firmado, simplemente debemos de configurar el nuevo Virtual Host. Es requisito que no disponga de autenticación básica de usuario como ya mencionamos en el inicio del apartado, es por ello que vamos a crear uno nuevo eliminando las opciones relacionadas con ello y añadiendo las que son necesarias para la **configuración SSL** (autenticación de servidor y cliente), que son las siguientes:

```
SSLEngine on
    SSLCertificateFile      /home/servidor/demoCA/servercert.pem
    SSLCertificateKeyFile   /home/servidor/demoCA/serverkey.pem
    SSLCACertificateFile    /home/servidor/demoCA/cacert.pem
    SSLVerifyClient         require
    SSLVerifyDepth          10
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
            SSLOptions +StdEnvVars
        </FilesMatch>

    BrowserMatch "MSIE [2-6]" \
        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0
    #MSIE 7 and newer should be able to use keepalive
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
```

- En la **Ilustración 56** puede verse como queda finalmente el fichero de configuración del Virtual Host.

Las siguientes líneas son las que corresponden con la autenticación de cliente:

- SSLVerifyClient require
- SSLVerifyDepth 10

Ahora debemos de habilitar el nuevo sitio:

```
sudo a2ensite httpswwwstum
```

Tenemos que añadir en apache el módulo SSL:

```
sudo a2enmod ssl
```

Debemos de asegurarnos de que está habilitado puerto 443 en /etc/apache2/ports.conf

- namevirtualhost *:443
- Listen 443 (suele estar activado por defecto)

Ahora nos queda reiniciar el servicio de apache2:

```
sudo service apache2 restart
```

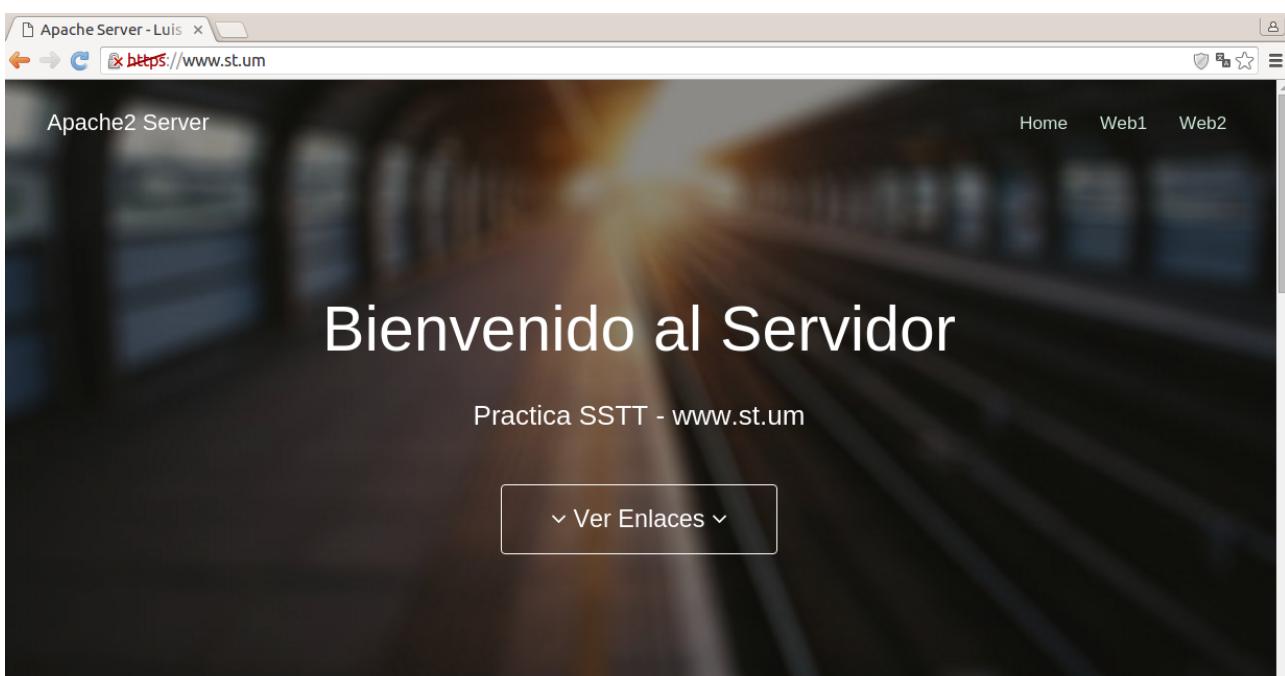


Ilustración 66: <https://www.st.um>

Trazas obtenidas con Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
119	12.579844	192.168.178.81	192.168.178.80	TLSv1.1	1587	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Name, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
121	12.672370	192.168.178.80	192.168.178.81	TLSv1.1	921	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
122	12.674754	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
125	12.685938	192.168.178.81	192.168.178.80	TLSv1.1	119	Encrypted Alert
132	12.696445	192.168.178.80	192.168.178.81	TLSv1.1	236	Client Hello
134	12.698307	192.168.178.81	192.168.178.80	TLSv1.1	1514	Server Hello, Certificate, Server Key Exchange
136	12.701998	192.168.178.81	192.168.178.80	TLSv1.1	139	Certificate Request, Server Hello Done

▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 58155 (58155), Dst Port: https (443), Seq: 1, Ack: 1, Len: 170
 ▶ Secure Sockets Layer
 ▶ TLSv1.1 Record Layer: Handshake Protocol: Client Hello
 Content Type: Handshake (22)
 Version: TLS 1.0 (0x0301)
 Length: 165
 ▶ Handshake Protocol: Client Hello
 Handshake Type: Client Hello (1)
 Length: 161
 Version: TLS 1.1 (0x0302)
 ▶ Random
 Session ID Length: 0
 Cipher Suites Length: 28
 ▶ Cipher Suites (14 suites)

Ilustración 67: el Cliente se presenta (Client Hello) - Fase 1

No.	Time	Source	Destination	Protocol	Length	Info
119	12.579844	192.168.178.81	192.168.178.80	TLSv1.1	1587	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Name, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
121	12.672370	192.168.178.80	192.168.178.81	TLSv1.1	921	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
122	12.674754	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
125	12.685938	192.168.178.81	192.168.178.80	TLSv1.1	119	Encrypted Alert
132	12.696445	192.168.178.80	192.168.178.81	TLSv1.1	236	Client Hello
134	12.698307	192.168.178.81	192.168.178.80	TLSv1.1	1514	Server Hello, Certificate, Server Key Exchange
136	12.701998	192.168.178.81	192.168.178.80	TLSv1.1	139	Certificate Request, Server Hello Done

▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: https (443), Dst Port: 58155 (58155), Seq: 1, Ack: 171, Len: 1448
 ▶ Secure Sockets Layer
 ▶ TLSv1.1 Record Layer: Handshake Protocol: Server Hello
 Content Type: Handshake (22)
 Version: TLS 1.1 (0x0302)
 Length: 65
 ▶ Handshake Protocol: Server Hello
 Handshake Type: Server Hello (2)
 Length: 61
 Version: TLS 1.1 (0x0302)
 ▶ Random
 Session ID Length: 0
 Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
 Compression Method: null (0)
 Extensions Length: 21

Ilustración 68: el Servidor responde (Server Hello) e indica que es necesario certificado para autenticarse

Filter: ssl		Expression... Clear Apply				
No.	Time	Source	Destination	Protocol	Length	Info
122	12.07.17.81	192.168.178.81	192.168.178.80	TLSv1.1	572	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
125	12.685938	192.168.178.81	192.168.178.80	TLSv1.1	119	Encrypted Alert
132	12.696445	192.168.178.80	192.168.178.81	TLSv1.1	236	Client Hello
134	12.698307	192.168.178.81	192.168.178.80	TLSv1.1	1514	Server Hello, Certificate, Server Key Exchange
136	12.701998	192.168.178.81	192.168.178.80	TLSv1.1	139	Certificate Request, Server Hello Done
138	12.706070	192.168.178.80	192.168.178.81	TLSv1.1	921	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, E
139	12.708094	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Messag
140	12.720693	192.168.178.80	192.168.178.81	TLSv1.1	471	Application Data

► Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ► Transmission Control Protocol, Src Port: https (443), Dst Port: 58155 (58155), Seq: 1449, Ack: 171, Len: 73
 ► [2 Reassembled TCP Segments (95 bytes): #134(22), #136(73)]
 ▾ Secure Sockets Layer
 ▾ TLSv1.1 Record Layer: Handshake Protocol: Multiple Handshake Messages
 Content Type: Handshake (22)
 Version: TLS 1.1 (0x0302)
 Length: 90
 ▾ Handshake Protocol: Certificate Request
 Handshake Type: Certificate Request (13)
 Length: 82
 Certificate types count: 3
 ► Certificate types (3 types)
 Distinguished Names Length: 76
 ► Distinguished Names (76 bytes)
 ▾ Handshake Protocol: Server Hello Done

Ilustración 69: el Servidor solicita el certificado (Certificate Request) - Fase 3

Filter: ssl		Expression... Clear Apply				
No.	Time	Source	Destination	Protocol	Length	Info
122	12.07.17.81	192.168.178.81	192.168.178.80	TLSv1.1	572	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
125	12.685938	192.168.178.81	192.168.178.80	TLSv1.1	119	Encrypted Alert
132	12.696445	192.168.178.80	192.168.178.81	TLSv1.1	236	Client Hello
134	12.698307	192.168.178.81	192.168.178.80	TLSv1.1	1514	Server Hello, Certificate, Server Key Exchange
136	12.701998	192.168.178.81	192.168.178.80	TLSv1.1	139	Certificate Request, Server Hello Done
138	12.706070	192.168.178.80	192.168.178.81	TLSv1.1	921	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, E
139	12.708094	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Messag
140	12.720693	192.168.178.80	192.168.178.81	TLSv1.1	471	Application Data

► Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ► Transmission Control Protocol, Src Port: 58155 (58155), Dst Port: https (443), Seq: 171, Ack: 1522, Len: 855
 ▾ Secure Sockets Layer
 ▾ TLSv1.1 Record Layer: Handshake Protocol: Certificate
 Content Type: Handshake (22)
 Version: TLS 1.1 (0x0302)
 Length: 625
 ▾ Handshake Protocol: Certificate
 Handshake Type: Certificate (11)
 Length: 621
 Certificates Length: 618
 ► Certificates (618 bytes)
 ▾ TLSv1.1 Record Layer: Handshake Protocol: Client Key Exchange
 Content Type: Handshake (22)
 Version: TLS 1.1 (0x0302)
 Length: 70

Ilustración 70: el Cliente envía su certificado (Certificate, Client KeyExchange, Certificate Verify, CCSP, Encrypted Handshake Message)

Filter: ssl Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
122	12.07.77.57	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
125	12.685938	192.168.178.81	192.168.178.80	TLSv1.1	119	Encrypted Alert
132	12.696445	192.168.178.80	192.168.178.81	TLSv1.1	236	Client Hello
134	12.698307	192.168.178.81	192.168.178.80	TLSv1.1	1514	Server Hello, Certificate, Server Key Exchange
136	12.701998	192.168.178.81	192.168.178.80	TLSv1.1	139	Certificate Request, Server Hello Done
138	12.706070	192.168.178.80	192.168.178.81	TLSv1.1	921	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
139	12.708094	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
140	12.720693	192.168.178.80	192.168.178.81	TLSv1.1	471	Application Data

► Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▷ Transmission Control Protocol, Src Port: https (443), Dst Port: 58155 (58155), Seq: 1522, Ack: 1026, Len: 906
 Source port: https (443)
 Destination port: 58155 (58155)
 [Stream index: 31]
 Sequence number: 1522 (relative sequence number)
 [Next sequence number: 2428 (relative sequence number)]
 Acknowledgement number: 1026 (relative ack number)
 Header length: 32 bytes
 ► Flags: 0x018 (PSH, ACK)
 Window size value: 248
 [Calculated window size: 31744]
 [Window size scaling factor: 128]
 ► Checksum: 0x89e6 [validation disabled]
 ► Options: (12 bytes)
 ► FIN/ACK analysis

Ilustración 71: el Servidor finaliza la Fase 4.

Filter: ssl Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
125	12.685938	192.168.178.81	192.168.178.80	TLSv1.1	119	Encrypted Alert
132	12.696445	192.168.178.80	192.168.178.81	TLSv1.1	236	Client Hello
134	12.698307	192.168.178.81	192.168.178.80	TLSv1.1	1514	Server Hello, Certificate, Server Key Exchange
136	12.701998	192.168.178.81	192.168.178.80	TLSv1.1	139	Certificate Request, Server Hello Done
138	12.706070	192.168.178.80	192.168.178.81	TLSv1.1	921	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
139	12.708094	192.168.178.81	192.168.178.80	TLSv1.1	972	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
140	12.720693	192.168.178.80	192.168.178.81	TLSv1.1	471	Application Data

► Frame 140: 471 bytes on wire (3768 bits), 471 bytes captured (3768 bits)
 ► Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ► Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▷ Transmission Control Protocol, Src Port: 58155 (58155), Dst Port: https (443), Seq: 1026, Ack: 2428, Len: 405
 Source port: 58155 (58155)
 Destination port: https (443)
 [Stream index: 31]
 Sequence number: 1026 (relative sequence number)
 [Next sequence number: 1431 (relative sequence number)]
 Acknowledgement number: 2428 (relative ack number)
 Header length: 32 bytes
 ► Flags: 0x018 (PSH, ACK)
 Window size value: 274
 [Calculated window size: 35072]
 [Window size scaling factor: 128]

Ilustración 72: se produce el intercambio de los datos cifrados

3.3.3 Web-SSTT HTTP Server

Este servidor HTTP ha sido desarrollado en el **lenguaje de programación C**. Esta sección de la práctica está basada en la primera sesión vista en la asignatura y se disponía para ello de un fichero base, mediante el cual se ha montado todo el trabajo desarrollado. Para que funcione correctamente debemos de darlo de alta en el Servidor DNS, como se hace con todos los servicios.

Apache funciona en el puerto 80. Es por ello que, cuando queremos ejecutar nuestro software debemos de hacerlo en un puerto diferente ya que si no es así se generaría conflicto.

3.3.3.1 Funcionamiento del Software

Compilar:

```
gcc web_sstt.c -o web_sstt
```

Ejecución:

```
./web_sstt número-de-puerto directorio-web (un . Indica el mismo del código)  
./web_sstt 8080 .
```

Luego simplemente, como tenemos dado de alta **web.st.um** en el servidor DNS, tendremos que acceder desde el navegador del cliente a la dirección especificando el puerto.

3.3.3.2 Código Implementado

```
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <errno.h>  
#include <string.h>  
#include <fcntl.h>  
#include <signal.h>  
#include <sys/types.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <arpa/inet.h>  
#define VERSION 23  
#define BUFFER_SIZE 8096  
#define ERROR 42  
#define LOG 44  
#define FORBIDDEN 403  
#define NOTFOUND 404  
  
struct {  
    char *ext;  
    char *filetype;  
} extensions [] = {  
    {"gif", "image/gif"},  
    {"jpg", "image/jpg"},  
    {"jpeg", "image/jpeg"},  
    {"png", "image/png"},  
    {"ico", "image/ico"},
```

```

{"zip", "image/zip" },
 {"gz", "image/gz" },
 {"tar", "image/tar" },
 {"htm", "text/html" },
 {"html","text/html" },
 {0,0} };

/* Función ejecutada por el proceso hijo para procesar la conexión web */
/***********************/
void servidor_web(int fd)
{
    int j, file_fd, buflen;
    long i, ret, len;
    char * fstr;

    /* Definir buffer para leer las peticiones */
    static char buffer[BUFFER_SIZE+1];

    /* Leer la petición HTTP */
    ret =read(fd,buffer,BUFFER_SIZE);

    /* Comprobación de errores de lectura*/
    if(ret == 0 || ret == -1) {
        printf("Error de lectura de petición");
    }

    /* Si la lectura tiene datos válidos terminar el buffer con un \0 */
    if(ret > 0 && ret < BUFFER_SIZE)
        buffer[ret]=0;           /* terminar el buffer */
    else buffer[0]=0;

    /* Imprimir línea a línea la petición HTTP en pantalla */
    printf("Recibido: %s \n",buffer);

    /* Realizar un "parsing" del HTTP request */
    for(i=0;i<ret;i++)
        if(buffer[i] == '\r' || buffer[i] == '\n')
            buffer[i]='*';

    /* Si no es un GET rechazar porque solo se permite esta operación */
    if( strncmp(buffer,"GET ",4) && strncmp(buffer,"get ",4) ) {
        printf("ERROR: solo se permite la operación GET \n");
    }

    /* Rellenamos con 0's el resto del buffer */
    for(i=4;i<BUFFER_SIZE;i++) {
        if(buffer[i] == ' ') {
            buffer[i] = 0;
            break;
        }
    }

    /* Comprobación del uso de rutas de directorio padre no permitidas */
    for(j=0;j<i-1;j++)
        if(buffer[j] == '.' && buffer[j+1] == '.') {
            printf("Directorio padre (...) ruta incompatible \n");
        }
}

```

```

/* Si no se especifica el index.html añadirlo por defecto a la petición */
if(      !strncmp(&buffer[0],"GET          /\0",6)           ||      !
strncmp(&buffer[0],"get /\0",6) )
    (void)strcpy(buffer,"GET /index.html");

/* Verificar si se soporta el tipo de fichero de la petición. Ver tabla
"extensiones" y responder con el código de error en caso de que no*/
buflen=strlen(buffer);
fstr = (char *)0;
for(i=0;extensions[i].ext != 0;i++) {
    len = strlen(extensions[i].ext);
    if( !strncmp(&buffer[buflen-len], extensions[i].ext, len)) {
        fstr =extensions[i].filetype;
        break;
    }
}
if(fstr == 0) printf("Tipo de fichero no soportado \n");

/* Abrir el contenido del fichero especificado en la URL*/
file_fd = open(&buffer[5],O_RDONLY);

/* Crear una respuesta y enviar los datos en bloques de 8KB */
if (file_fd == -1) {
    printf("No se puede abrir el archivo correctamente \n");
    long fileError_fd = open("404.html",O_RDONLY);
    sprintf(buffer,"HTTP/1.0 404 Not Found \r\nContent-Type:\r\n\r\n");
    write(fd,buffer,strlen(buffer));
    while ((ret = read(fileError_fd, buffer, BUFFER_SIZE)) > 0)
        write(fd,buffer,ret);
}
else {
    printf("Fichero abierto correctamente \n");
    sprintf(buffer,"HTTP/1.0 200 OK\r\nContent-Type: %s\r\n\r\n",
fstr);
    write(fd,buffer,strlen(buffer));

    /* Enviar el resto del contenido */
    while ( (ret = read(file_fd, buffer, BUFFER_SIZE)) > 0)
        write(fd,buffer,ret);
}

printf("Se ha procesado todo correctamente, hacer nuevas peticiones \n");
sleep(1);      /* esperar un poco a que se vacie el buffer */
exit(1);
}

int main(int argc, char **argv)
{
    int i, port, pid, listenfd, socketfd;
    socklen_t length;
    static struct sockaddr_in cli_addr;      /* static = inicializa a ceros
*/

```

```

static struct sockaddr_in serv_addr;      /* static = inicializa a ceros
*/
if( argc < 3 || argc > 3 ) {
    printf("Uso: web-sstt numero-puerto directorio-web\n");
    printf("No se soporta: URLs que incluyan \".\", Java,
Javascript, CGI\n");
    exit(0);
}

if(chdir(argv[2]) == -1)
{
    printf("No se puede cambiar al directorio %s\n", argv[2]);
    exit(4);
}

/* Hacer un demonio que no se pueda parar y que no genere zombies (no
wait()) */
if(fork() != 0)
    return 0;

(void)signal(SIGCLD, SIG_IGN); /* ignora muertes de procesos hijo */
(void)signal(SIGHUP, SIG_IGN); /* ignora cuelges de terminal */

for(i=0;i<32;i++)
    (void)close(i);           /* cerrar ficheros abiertos */
(void)setpgrp();             /* romper grupo de procesos */
printf("INFO: iniciando Servidor \n");

/* Establecer el socket con el puerto pasado como parámetro */
if((listenfd = socket(AF_INET, SOCK_STREAM, 0)) <0)
    printf("ERROR en la creación del socket \n");
port = atoi(argv[1]);

/* Comprobación de puertos inválidos */
if(port < 0 || port >60000)
    printf("ERROR, número de puerto inválido (pruebe 1->60000) \n");

/* Asignación de los valores al socket de puerto y de direcciones */
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(port);

/* Hacer un bind */
if(bind(listenfd, (struct sockaddr *)&serv_addr,sizeof(serv_addr)) <0)
    printf("ERROR al hacer bind sobre el socket: %d \n",errno);

/* Hacer un listen sobre el socket */
if( listen(listenfd,64) <0)
    printf("ERROR al hacer el listen \n");

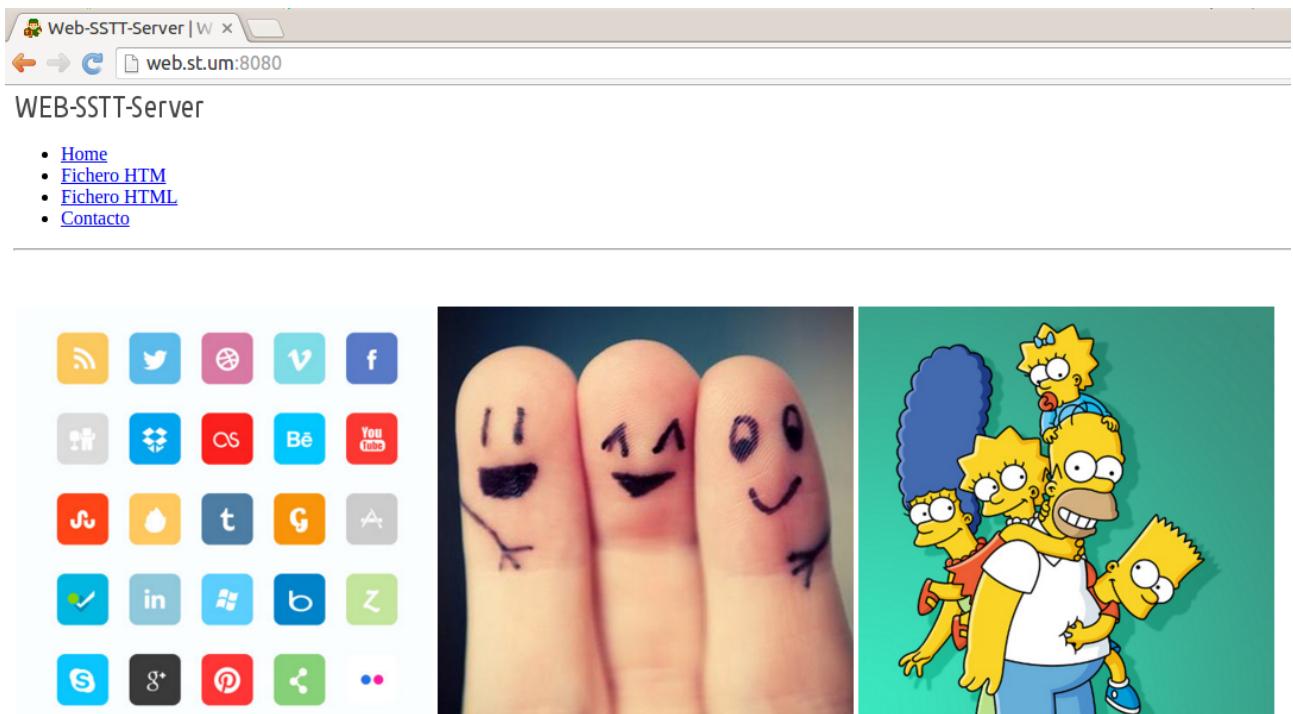
while(1) {
    length = sizeof(cli_addr);
    if((socketfd = accept(listenfd, (struct sockaddr *)&cli_addr,
&length)) < 0)
        printf("Error al hacer accept %d \n", errno);
    if((pid = fork()) < 0) {

```

```

        printf("Error al hacer fork %d \n",errno);
    }
    else {
        if(pid == 0) { /* hijo */
            (void)close(listenfd);
            servidor_web(socketfd);
        } else { /* padre */
            (void)close(socketfd);
        }
    }
}
}

```



[Presiona para ir al fichero ICO](#) [Presiona para ir al fichero JPG](#) [Presiona para ir al fichero JPEG](#)

Ilustración 73: web.st.um:8080 desde máquina Cliente

NOTA: este código corresponde con la Versión1 implementada donde no era necesario generar un fichero de log y no existia una gestión básica de Cookies.

>> Se ha generado un apartado nuevo en el documento para tratar la nueva implementación.

3.3.3.3 Trazas en Wireshark

Filter: http						
No.	Time	Source	Destination	Protocol	Length	Info
78	6.031733	192.168.178.80	192.168.178.81	HTTP	434	GET / HTTP/1.1
92	6.092077	192.168.178.80	192.168.178.81	HTTP	378	GET /js/jquery.js HTTP/1.1
93	6.092253	192.168.178.80	192.168.178.81	HTTP	376	GET /js/main.js HTTP/1.1
95	6.092655	192.168.178.80	192.168.178.81	HTTP	403	GET /img/logo.png HTTP/1.1
118	6.179308	192.168.178.80	192.168.178.81	HTTP	402	GET /img/ico.jpg HTTP/1.1
119	6.179515	192.168.178.80	192.168.178.81	HTTP	402	GET /img/jpg.jpg HTTP/1.1
122	6.179801	192.168.178.80	192.168.178.81	HTTP	403	GET /img/jpeg.jpg HTTP/1.1
164	6.204285	192.168.178.80	192.168.178.81	HTTP	402	GET /img/gif.jpg HTTP/1.1

Frame 78: 434 bytes on wire (3472 bits), 434 bytes captured (3472 bits)
 ▶ Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
 ▶ Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
 ▶ Transmission Control Protocol, Src Port: 60677 (60677), Dst Port: http-alt (8080), Seq: 1, Ack: 1, Len: 368
 ▶ Hypertext Transfer Protocol
 ▶ GET / HTTP/1.1\r\n
 Host: web.st.um:8080\r\n
 Connection: keep-alive\r\n
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
 Upgrade-Insecure-Requests: 1\r\n
 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36\r\n
 Accept-Encoding: gzip, deflate, sdch\r\n
 Accept-Language: es-ES,es;q=0.8\r\n
 \r\n
 [Full request URI: http://web.st.um:8080/]

Ilustración 74: el Cliente realiza GET

Filter: http						
No.	Time	Source	Destination	Protocol	Length	Info
164	6.204285	192.168.178.80	192.168.178.81	HTTP	402	GET /img/gif.jpg HTTP/1.1
190	7.032907	192.168.178.81	192.168.178.80	HTTP	66	HTTP/1.0 200 OK (text/html)
196	7.037959	192.168.178.80	192.168.178.81	HTTP	402	GET /img/hmn.ico HTTP/1.1

\r\n
 ▶ Line-based text data: text/html
 <!DOCTYPE html>\r\n
 <html lang="es">\r\n
 <head>\r\n
 <t>title>Web-SSTT-Server | Web de Pruebas</title>\r\n
 <t>meta charset="utf-8">\r\n
 <t>meta name="author" content="luismiguelgarcia.com">\r\n
 <t>meta name="description" content="Web Ejemplo SSTT"/>\r\n
 <t>meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0" />\r\n
 <t>link rel="icon" type="image/ico" href="img/link.ico" />\r\n
 <script type="text/javascript" src="js/jquery.js"></script>\r\n
 <script type="text/javascript" src="js/main.js"></script>\r\n
 </head>\r\n
 <body>\r\n
 <r>\n
 <t>header>\r\n
 <r>\n
 <t>t<div class="logo">\r\n
 <t>t\r\n
 </t>

Ilustración 75: el Servidor responde al GET

Filter: http						
No.	Time	Source	Destination	Protocol	Length	Info
93	6.092253	192.168.178.80	192.168.178.81	HTTP	376	GET /js/main.js HTTP/1.1
95	6.092655	192.168.178.80	192.168.178.81	HTTP	403	GET /img/logo.png HTTP/1.1
118	6.179308	192.168.178.80	192.168.178.81	HTTP	402	GET /img/ico.jpg HTTP/1.1
119	6.179515	192.168.178.80	192.168.178.81	HTTP	402	GET /img/jpg.jpg HTTP/1.1
122	6.179801	192.168.178.80	192.168.178.81	HTTP	403	GET /img/jpeg.jpg HTTP/1.1
164	6.204285	192.168.178.80	192.168.178.81	HTTP	402	GET /img/gif.jpg HTTP/1.1
190	7.032907	192.168.178.81	192.168.178.80	HTTP	66	HTTP/1.0 200 OK (text/html)
196	7.037959	192.168.178.80	192.168.178.81	HTTP	402	GET /img/bmp.jpg HTTP/1.1
208	7.092919	192.168.178.81	192.168.178.80	HTTP	66	HTTP/1.0 404 Not Found ()

Frame 208: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ▶ Transmission Control Protocol, Src Port: http-alt (8080), Dst Port: 60678 (60678), Seq: 1561, Ack: 314, Len: 0
 ▶ [4 Reassembled TCP Segments (1560 bytes): #97(42), #100(1032), #206(486), #208(0)]
 ▶ Hypertext Transfer Protocol
 ▶ HTTP/1.0 404 Not Found \r\n
 Content-Type:\r\n
 \r\n
 ▶ Media Type

Ilustración 76: Error 404, se solicita un recurso que no esta disponible

3.4 Desplegar Servicio NTP

Network Time Protocol (NTP) es un protocolo de Internet para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable (mantiene el reloj sincronizado en un rango de milisegundos respecto a Coordinated Universal Time). NTP utiliza UDP como su capa de transporte, usando el puerto 123. Está diseñado para resistir los efectos de la latencia variable.

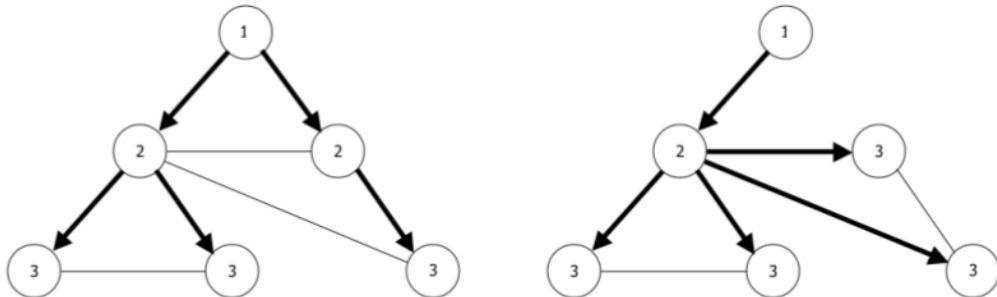


Ilustración 77: ejemplo de sincronización de jerarquía de relojes

3.4.1 Configuración del Servidor

Para dar soporte NTP a nuestro Servidor debemos de realizar una serie de sencillos pasos:

- Actualización del sistema:

```
sudo apt-get update
```

- Instalación de NTP mediante comando en la Terminal:

```
sudo apt-get install ntp
```

Una vez lo tenemos instalado tenemos que realizar la configuración básica en el fichero **/etc/ntp.conf**. Para ello tenemos que añadir los siguientes servidores:

- server 0.ubuntu.pool.ntp.org
- server 1.ubuntu.pool.ntp.org
- server 2.ubuntu.pool.ntp.org
- server 3.ubuntu.pool.ntp.org

Una vez que tenemos los servidores, tenemos que reiniciar el Servicio NTP:

```
sudo service ntp restart/reload
```

Una vez realizado todo esto tan solo nos queda dar de alta el servicio en el Servidor DNS.

3.4.2 Configuración del Cliente

Todo lo anteriormente descrito corresponde con la parte del Servidor. Ahora toca hablar de la configuración que tenemos que realizar por parte del **Cliente**. Para poder llevarla a cabo tenemos que crear el fichero **/etc/cron.daily/ntpdate**. A este archivo tenemos que:

- Añadir el contenido `ntpdate 192.168.178.81` (IP del Servidor).
- Dar permiso 775.

```
sudo chmod 755 /etc/cron.daily/ntpdate
```

Para probar el funcionamiento de NTP, se realiza una consulta al servidor NTP que se acaba de crear.

```
sudo ntpdate 192.168.178.81
```

```
cliente@cliente-VirtualBox:/etc$ sudo ntpdate 192.168.178.81
16 Jan 03:47:56 ntpdate[2189]: adjust time server 192.168.178.81 offset 0.068760 sec
```

Ilustración 78: probando ntpdate

3.4.3 Trazas en Wireshark

A continuación se van a mostrar las trazas que hemos obtenido con Wireshark.

Filter: ntp							Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info			
13	21.655378	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client			
14	21.656119	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server			
19	23.655266	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client			
20	23.656124	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server			
23	25.655194	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client			
24	25.655880	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server			
27	27.655173	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client			
28	27.655874	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server			

► Frame 13: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
► Ethernet II, Src: CadmusCo_92:d7:62 (08:00:27:92:d7:62), Dst: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33)
► Internet Protocol Version 4, Src: 192.168.178.80 (192.168.178.80), Dst: 192.168.178.81 (192.168.178.81)
► User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
▼ Network Time Protocol
► Flags: 0xe3
 Peer Clock Stratum: unspecified or invalid (0)
 Peer Polling Interval: invalid (3)
 Peer Clock Precision: 0,015625 sec
 Root Delay: 1,0000 sec
 Root Dispersion: 1,0000 sec
 Reference ID: NULL
 Reference Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
 Origin Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
 Receive Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
 Transmit Timestamp: Jan 16, 2016 03:01:42.388958000 UTC

Ilustración 79: consulta NTP desde Cliente al Servidor

Filter: **ntp** Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
13	21.655378	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client
14	21.656119	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server
19	23.655266	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client
20	23.656124	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server
23	25.655194	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client
24	25.655880	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server
27	27.655173	192.168.178.80	192.168.178.81	NTP	90	NTP Version 4, client
28	27.655874	192.168.178.81	192.168.178.80	NTP	90	NTP Version 4, server

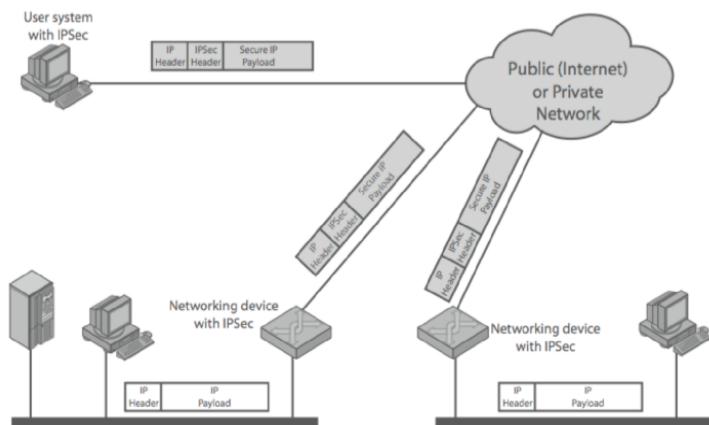
► Frame 14: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
 ► Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ► Internet Protocol Version 4, Src: 192.168.178.81 (192.168.178.81), Dst: 192.168.178.80 (192.168.178.80)
 ► User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
 ▾ Network Time Protocol
 ► Flags: 0x24
 Peer Clock Stratum: secondary reference (3)
 Peer Polling Interval: invalid (3)
 Peer Clock Precision: 0,00000 sec
 Root Delay: 0,0291 sec
 Root Dispersion: 0,1336 sec
 Reference ID: 212.18.3.18
 Reference Timestamp: Jan 16, 2016 02:53:12.745716000 UTC
 Origin Timestamp: Jan 16, 2016 03:01:42.388958000 UTC
 Receive Timestamp: Jan 16, 2016 03:02:23.655118000 UTC
 Transmit Timestamp: Jan 16, 2016 03:02:23.655302000 UTC

Ilustración 80: respuesta NTP del Servidor a el Cliente

3.5 IPsec

Este apartado es adicional a la primera práctica entregada en la convocatoria de Febrero 2016, donde era opcional y en esta nueva **Versión 2.0** se incluye puesto que ahora es una parte obligatoria del nuevo boletín de Práctica Final.

Internet Protocol security (Ipsec) es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el protocolo IP autenticando y/o cifrando cada paquete IP en un flujo de datos (IPsec nos permite cifrar, autenticar y dar integridad a toda la comunicación entre hosts.)



3.5.1 Configuración

Para la correcta instalación debemos de asegurarnos que en ambos equipos (Cliente y Servidor) disponemos de *conexión a internet*. Vamos a utilizar la herramienta **ipsec-tools**, la cual instalaremos en ambos equipos:

```
sudo apt-get install ipsec-tools
```

Se aplicarán algoritmos y claves precompartidas.

- **No se aplicará IKE.**

Se tendrá:

- Seguridad extremo a extremo.
- Autenticación mediante AH.
- Cifrado mediante ESP.

3.5.1.1 Cabeceras AH

Para comenzar con la configuración de la aplicación tendremos que ir al fichero “**/etc/ipsec-tools.conf**” y editarlo. Este consta de dos partes diferentes:

1) Parte donde se especifican las claves y el algoritmo para los HMACs necesarios para el protocolo AH en la dirección Host1 a Host2 y viceversa (Claves128 bits). **Eliminaremos los comentarios de:**

```
#flush;
#spdflush;
```

Tendremos que añadir al fichero las siguientes líneas (tanto en Cliente como en Servidor):

```
# Configuración de AH en Host1 y Host2 (cada comando add en una única línea)
add XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY ah 0x200 -A hmac-md5
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX ah 0x300 -A hmac-md5
0x96358c90783bbfa3d7b196ceabe0536b;
```

- Donde `xxx.xxx.xxx.xxx` corresponde con la IP del Host1 y `yyy.yyy.yyy.yyy` corresponde con la IP del Host2.
- `ah`: cabecera IPSec a utilizar.
- `-A`: parámetros para Autenticación.
- `hmac-md5`: función HMAC para autenticación.
- `0xXXXXX`: clave compartida para autenticación (128 bits).

2) Se establecen las políticas de seguridad a aplicar en función de las características del tráfico. En nuestro caso, el tráfico que vaya destinado o provenga del otro host. Tendremos que añadir lo siguiente:

```
# Configuración de las SPD en host1, para AH
spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P out ipsec ah/transport//require;
spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P in ipsec ah/transport//require;
```

```
# Configuración de las SPD en host2, para AH
spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P in ipsec ah/transport//require;
spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P ioutipsec ah/transport//require;
```

- Donde `xxx.xxx.xxx.xxx` corresponde con la IP del Host1 y `yyy.yyy.yyy.yyy` corresponde con la IP del Host2.
- `any`: se aplicará esta política para cualquier protocolo de la capa de transporte
- `-P`: política a aplicar.
- `out`: se aplica al tráfico de salida.
- `in`: se aplica al tráfico de entrada.
- `ipsec`: se aplicará una transformación ipsec, en este caso, se aplicará la cabecera.
- `AH`, en modo transporte y es obligatorio (require).

La configuración en ambos equipos es la misma, únicamente cambia el sentido de las políticas de seguridad (in/out). De este modo, la comunicación en el canal troncal quedará protegida.

```
GNU nano 2.2.6                               Archivo: /etc/ipsec-tools.conf

#!/usr/sbin/setkey -f
#
# NOTE: Do not use this file if you use racoon with racoon-tool
# utility. racoon-tool will setup SAs and SPDs automatically using
# /etc/racoon/racoon-tool.conf configuration.
#
## Flush the SAD and SPD
#
flush;
spdflush;

## Configuración de AH en Host1 y Host2 (cada comando add en una única línea)
add 192.168.178.144 192.168.178.105 ah 0x200 -A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 192.168.178.105 192.168.178.144 ah 0x300 -A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

## Configuración de las SPD en host2, para AH
spdadd 192.168.178.144 192.168.178.105 any -P in ipsec ah/transport//require;
spdadd 192.168.178.105 192.168.178.144 any -P out ipsec ah/transport//require;

## Some sample SPDs for use racoon
#
# spdadd 10.10.100.1 10.10.100.2 any -P out ipsec
#     esp/transport//require;
#
# spdadd 10.10.100.2 10.10.100.1 any -P in ipsec
#     esp/transport//require;
```

Ilustración 82: fichero `/etc/ipsec-tools.conf`

Una vez realizados los dos pasos anteriores para la **Cabecera AH**, tendremos que activar el servicio setkey de ambos equipos:

```
sudo service setkey restart
```

Testearemos haciendo ping de Cliente → Servidor y Servidor → Cliente.

```
cliente@cliente-VirtualBox:~$ ping 192.168.178.144
PING 192.168.178.144 (192.168.178.144) 56(84) bytes of data.
64 bytes from 192.168.178.144: icmp_req=1 ttl=64 time=0.584 ms
64 bytes from 192.168.178.144: icmp_req=2 ttl=64 time=0.420 ms
64 bytes from 192.168.178.144: icmp_req=3 ttl=64 time=0.454 ms
64 bytes from 192.168.178.144: icmp_req=4 ttl=64 time=0.438 ms
64 bytes from 192.168.178.144: icmp_req=5 ttl=64 time=0.479 ms
64 bytes from 192.168.178.144: icmp_req=6 ttl=64 time=0.445 ms
^C
--- 192.168.178.144 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
rtt min/avg/max/mdev = 0.420/0.470/0.584/0.053 ms
```

Ilustración 83: Cliente → Servidor

```
servidor@servidor-VirtualBox:~$ ping 192.168.178.105
PING 192.168.178.105 (192.168.178.105) 56(84) bytes of data.
64 bytes from 192.168.178.105: icmp_req=1 ttl=64 time=0.444 ms
64 bytes from 192.168.178.105: icmp_req=2 ttl=64 time=0.522 ms
64 bytes from 192.168.178.105: icmp_req=3 ttl=64 time=0.432 ms
64 bytes from 192.168.178.105: icmp_req=4 ttl=64 time=0.454 ms
64 bytes from 192.168.178.105: icmp_req=5 ttl=64 time=0.454 ms
64 bytes from 192.168.178.105: icmp_req=6 ttl=64 time=0.474 ms
^C
--- 192.168.178.105 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4999ms
rtt min/avg/max/mdev = 0.432/0.463/0.522/0.034 ms
```

Ilustración 84: Servidor → Cliente

Trazas con Wireshark

42	4.237259	169.254.1.1	169.254.255.255	NBNS	110 Registration NB FRITZ-NAS<00>
43	4.237695	169.254.1.1	169.254.255.255	NBNS	110 Registration NB WORKGROUP<00>
44	4.238139	169.254.1.1	169.254.255.255	NBNS	110 Registration NB WORKGROUP<1e>
45	4.364668	192.168.178.47	224.0.0.251	MDNS	103 Standard query PTR _CC32E753._sub._googlecast._tcp.local, "QM" question PTR
46	8.745392	192.168.178.47	255.255.255.255	UDP	82 Source port: 57621 Destination port: 57621
47	19.097161	192.168.178.144	192.168.178.105	ICMP	122 Echo (ping) request id=0xd3b, seq=1/256, ttl=64
48	19.097625	192.168.178.105	192.168.178.144	ICMP	122 Echo (ping) reply id=0xd3b, seq=1/256, ttl=64
49	20.098546	192.168.178.144	192.168.178.105	ICMP	122 Echo (ping) request id=0xd3b, seq=2/252, ttl=64
50	20.098626	192.168.178.105	192.168.178.144	ICMP	122 Echo (ping) reply id=0xd3b, seq=2/252, ttl=64

Frame 47: 122 bytes on wire (976 bits), 122 bytes captured (976 bits)
 ▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)
 ▶ Internet Protocol Version 4, Src: 192.168.178.144 (192.168.178.144), Dst: 192.168.178.105 (192.168.178.105)
 ▾ Authentication Header
 Next Header: ICMP (0x01)
 Length: 24
 AH SPI: 0x000000200
 AH Sequence: 36
 AH ICV: aa252b9c094f1ba93dc9da70
 ▾ Internet Control Message Protocol
 Type: 8 (Echo (ping) request)
 Code: 0
 Checksum: 0x731f [correct]
 Identifier (BE): 3387 (0x0d3b)
 Identifier (LE): 15117 (0x3hd3b)

Ilustración 85: cabecera AH en Wireshark

Podemos ver las cabeceras AH y el contenido de los paquetes IP en claro.

3.5.1.2 Cabeceras ESP

Para activar el cifrado ESP para cifrado y autenticación tenemos que modificar el fichero “**ipsec-tools.conf**” en ambos extremos nuevamente.

```
# Configuracion de ESP en host1
add    XXX.XXX.XXX.XXX      YYY.YYY.YYY.YYY      esp      0x201      -E      3des-cbc
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831 -A hmac-md5 0xc0291ff014dcc$ 
add    YYY.YYY.YYY.YYY      XXX.XXX.XXX.XXX      esp      0x301      -E      3des-cbc
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df -A hmac-md5 0x96358c90783bb$ 

# Configuracion de las SPD en host1
spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P out ipsec esp/transport//require;
spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P in ipsec esp/transport//require;
```

```
# Configuracion de ESP en host2
add    XXX.XXX.XXX.XXX      YYY.YYY.YYY.YYY      esp      0x201      -E      3des-cbc
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831 -A hmac-md5 0xc0291ff014dcc$ 
add    YYY.YYY.YYY.YYY      XXX.XXX.XXX.XXX      esp      0x301      -E      3des-cbc
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df -A hmac-md5 0x96358c90783bb$ 

# Configuracion de las SPD en host2
spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P in ipsec esp/transport//require;
spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P out ipsec esp/transport//require;
```

NOTA: es importante dejar entonces comentadas las lineas pertenecientes a las SPD para AH.

```
# Configuración de las SPD en host1, para AH
#spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P out ipsec ah/transport//require;
#spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P in ipsec ah/transport//require;
```

```
# Configuración de las SPD en host2, para AH
#spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P in ipsec ah/transport//require;
#spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P ioutipsec ah/transport//require;
```

Una vez realizadas estas modificaciones anteriores para **Cifrado ESP**, tendremos que reiniciar el servicio setkey de ambos equipos:

```
sudo service setkey restart
```

Trazas con Wireshark

27 11.070247	192.168.178.144	192.168.178.105	ESP	134 ESP (SPI=0x00000201)
28 11.070977	192.168.178.105	192.168.178.144	ESP	134 ESP (SPI=0x00000301)
29 11.822564	30:59:b7:bc:d5:e1	Broadcast	ARP	60 Who has 192.168.178.113? Tell 0.0.0.0
30 11.822819	192.168.178.113	224.0.0.22	IGMP	70 V3 Membership Report / Join group 239.255.255.250 for any
▶ Frame 27: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits)				
▶ Ethernet II, Src: CadmusCo_9e:7c:33 (08:00:27:9e:7c:33), Dst: CadmusCo_92:d7:62 (08:00:27:92:d7:62)				
▼ Internet Protocol Version 4, Src: 192.168.178.144 (192.168.178.144), Dst: 192.168.178.105 (192.168.178.105)				
Version: 4				
Header length: 20 bytes				
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))				
Total Length: 120				
Identification: 0x2713 (10003)				
Flags: 0x02 (Don't Fragment)				
Fragment offset: 0				
Time to live: 64				
Protocol: ESP (50)				
Header checksum: 0x2cf6 [correct]				
Source: 192.168.178.144 (192.168.178.144)				
Destination: 192.168.178.105 (192.168.178.105)				
▼ Encapsulating Security Payload				
ESP SPI: 0x00000201				
ESP Sequence: 11				

Ilustración 86: Ping para ESP

3.5.2 Implementando IKE

Internet Key Exchange (IKE) es un protocolo usado para establecer una Asociación de Seguridad (**SA**) en el protocolo Ipsec.. IKE emplea un intercambio secreto de claves de tipo **Diffie-Hellman** para establecer el secreto compartido de la sesión. Se suelen usar sistemas de clave pública o clave pre-compartida.

Para su implementación usaremos la herramienta **Racoon** de forma conjunta a Ipsec-tools.

```
sudo apt-get install racoon
```

Configuración básica en /etc/racoon/racoon.conf (modo de configuración **directo**).

Configurararemos IKE para que la autenticación de ambos extremos se realice mediante **claves precompartidas**.

- También pueden usarse certificados X.509 o Kerberos.

Paso 0: eliminar las líneas “add” de los ficheros /etc/ipsec-tools.conf y dejar las “spdadd” en ambos hosts (dejamos comentadas).

Paso 1: configurar /etc/racoon/racoon.conf para usar claves precompartidas en ambos equipos.

- Las **claves** se almacenaran en **/etc/racoon/psk.txt**
 - Hemos usado zzz.zzz.zzz.zzz clavecompartida.
 - + zzz.zzz.zzz.zzz es la IP del otro equipo.

Paso 2: definir los métodos y algoritmos de cifrado y autenticación.

- Se definen los métodos y algoritmos de cifrado y autenticación con el otro extremo para la negociación IKE y para la SA IPSec en **/etc/racoon/racoon.conf**

En Servidor (Host1):

```
remote YYY.YYY.YYY.YYY (IP Cliente) {
    exchange_mode aggressive;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group modp1024;
    }
}

sainfo anonymous {
    pfs_group modp768;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}
```

En Cliente (Host2):

```
remote XXX.XXX.XXX.XXX (IP Servidor) {
    exchange_mode aggressive;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group modp1024;
    }
}

sainfo anonymous {
    pfs_group modp768;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}
```

Paso 3: definir las políticas de seguridad en ipsec-tools.conf como en el caso anterior.

Server:

```
flush;
spdflush;
# Configuracion de las SPD en host1
spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P out ipsec esp/transport//require;
spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P in ipsec esp/transport//require;
```

Client:

```
flush;
spdflush;
# Configuracion de las SPD en host1
spdadd XXX.XXX.XXX.XXX YYY.YYY.YYY.YYY any -P in ipsec esp/transport//require;
spdadd YYY.YYY.YYY.YYY XXX.XXX.XXX.XXX any -P out ipsec esp/transport//require;
```

Paso 4: reiniciar setkey y racoon en ambos equipos.

```
sudo service setkey restart
sudo service racoon restart
```

Trazas Wireshark

Realizamos un Ping de un equipo al otro para testear.

1	0.000000	192.168.178.144	192.168.178.105	ISAKMP	306	Aggressive
2	0.009311	192.168.178.105	192.168.178.144	ISAKMP	326	Aggressive
3	0.015278	192.168.178.144	192.168.178.105	ISAKMP	90	Aggressive
4	0.015583	192.168.178.144	192.168.178.105	ISAKMP	126	Informational

▀ User Datagram Protocol, Src Port: isakmp (500), Dst Port: isakmp (500)
Source port: isakmp (500)
Destination port: isakmp (500)
Length: 272
► Checksum: 0xe76c [validation disabled]
▀ Internet Security Association and Key Management Protocol
Initiator cookie: d99b951b8c729b34
Responder cookie: 0000000000000000
Next payload: Security Association (1)
Version: 1.0
Exchange type: Aggressive (4)
► Flags: 0x00
Message ID: 0x00000000
Length: 264
► Type Payload: Security Association (1)
► Type Payload: Key Exchange (4)
► Type Payload: Nonce (10)
► Type Payload: Identification (5)
► Type Payload: Vendor ID (13) : RFC 3706 DPD (Dead Peer Detection)

Ilustración 87: testeando IKE

4. COMUNICAR LAS MÁQUINAS VIRTUALES

Este se trata de un apartado adicional en la práctica que se considera muy importante para el desarrollo de la misma, ya que sin él sería imposible de realizar la tarea correctamente.

Para comunicar las máquinas virtuales tenemos que realizar una pequeña y sencilla configuración en las mismas en la sección **Configuración >> Red** de cada una de ellas.

Servidor ST

Tenemos que configurar el adaptador de red como **Adaptador Puente**. En nuestro caso, como utilizamos red inalámbrica seleccionamos **wlan0**.

Cliente ST

Tenemos que configurar el adaptador de red como **Adaptador Puente**. En nuestro caso, como utilizamos red inalámbrica seleccionamos **wlan0**.

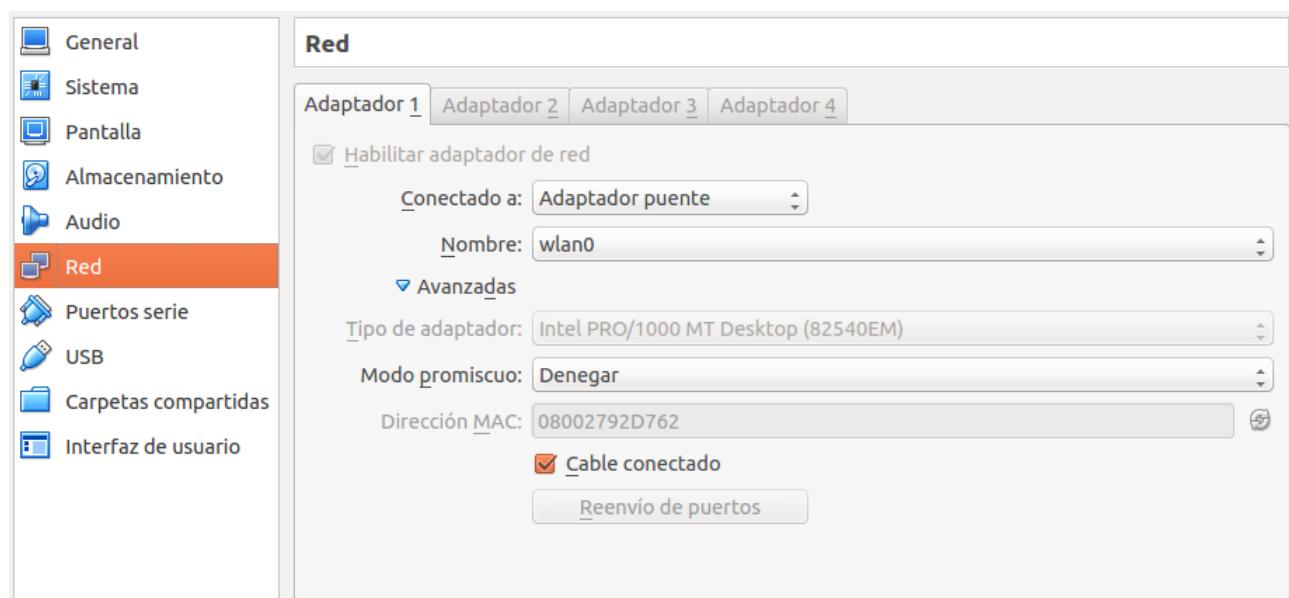


Ilustración 88: configuración red VMs

- Una vez hecho esto ya podemos hacer ping entre las máquinas para comprobar la correcta comunicación entre ellas.

5. PROBLEMAS ENCONTRADOS

En la realización de la práctica han ocurrido varios problemas. Uno de los problemas fue que, como todos los boletines de las sesiones de prácticas (en los cuales nos basamos para la realización de la misma) están orientados a Ubuntu 12.04, como se tenía en un principio instalado **Ubuntu 14.04** en las máquinas virtuales, a la hora de configurar el Servidor DNS en la sesión presencial se tuvo problemas, se optó por cortar de raíz el problema y rehacer el trabajo realizado montando de nuevo las máquinas bajo **Ubuntu 12.04**, como bien recomienda los boletines.

Otro problema que se ha tenido ha sido que mientras la práctica estaba en fase de desarrollo, por una **actualización de Windows**, se estropeó el Grub de arranque de Linux y se borro el sistema (se disponía de una partición de Windows y otra de Linux), por lo que se tuvo que volver a comenzar una vez más la práctica.

Con SSL se tuvo un problema que por culpa del navegador Firefox no podíamos visualizar correctamente webs con https si el CA no era una entidad conocida, por lo que se probó con Chrome y funcionaba todo correctamente.

Finalmente, como problema final que se ha tenido para mencionar fue a la hora de desplegar el servicio **DNS**. Simplemente ocurrió que a no se introdujo por completo la IP del Servidor en uno de los ficheros de configuración de Bind9 y no funcionaba de forma adecuada.

Salvo estos problemas menores no se han encontrado ningún otro destacable.

6. APARTADO EXTRA PARA VERSION 2 (Webserver en C)

Puesto que en la nueva versión del Servidor implementado C requiere la **gestión básica de Cookies** y ha de generar un **fichero de LOGs**, se ha tenido que modificar la versión implementada para la convocatoria de Febrero. Puesto que las modificaciones han sido mínimas se ha considerado más oportuno crear un apartado adicional en lugar de realizar completamente de nuevo el apartado correspondiente.

En la Versión 1 del Servidor realizábamos impresiones de los ERRORES y de lo que iba ocurriendo. Ahora tenemos que ir recargando el fichero de LOGs para que nos vaya imprimiendo por terminal que ocurre.

Ejemplo impresion Version 1:

```
printf("Error al hacer fork %d \n",errno);
```

Pero, gracias a la base que ha sido proporcionada por los profesores, se emplea una función debug que se encarga de insertar todo ello en un fichero .log.

Ejemplo impresion Versión 2:

```
debug(ERROR,"system call","fork",0);
```

6.1 Código función debug

```
/* Función de la Version 2 para el fichero log que se generara */
/*********************************************************/
void debug(int log_message_type, char *message, char *additional_info, int socket_fd)
{
    int fd ;
    char logbuffer[BUFSIZE*2];

    switch (log_message_type) {
        case ERROR: (void)sprintf(logbuffer,"ERROR: %s:%s Errno=%d exiting
pid=%d",message, additional_info, errno,getpid());
                    break;
        case PROHIBIDO:
            // Enviar como respuesta 403 Forbidden
            (void)write(socket_fd, "HTTP/1.1 403 Forbidden\nContent-
Length: 185\nConnection: close\nContent-Type:
text/html\n\n<html><head>\n<title>403
Forbidden</title>\n</head><body>\n<h1>Forbidden</h1>\nThe requested URL, file
type or operation is not allowed on this simple static file
webserver.\n</body></html>\n",271);
            (void)sprintf(logbuffer,"FORBIDDEN: %s:%s",message,
additional_info);
                    break;
        case NOENCONTRADO:
            // Enviar como respuesta 404 Not Found
            (void)write(socket_fd, "HTTP/1.1 404 Not Found\nContent-
Length: 136\nConnection: close\nContent-Type:
text/html\n\n<html><head>\n<title>404 Not Found</title>\n</head><body>\n<h1>Not
Found</h1>\n</body>\n",271);
            (void)sprintf(logbuffer,"NOT FOUND: %s:%s",message,
additional_info);
                    break;
    }
}
```

```

Found</h1>\nThe requested URL was not found on this
server.\n</body></html>\n",224);
        (void)sprintf(logbuffer,"NOT FOUND: %s:%s",message,
additional_info);
        break;
    case LOG: (void)sprintf(logbuffer," INFO: %s:%s:%d",message,
additional_info, socket_fd); break;
}

if((fd = open("web_sstt.log", O_CREAT| O_WRONLY | O_APPEND,0644)) >= 0) {
    (void)write(fd,logbuffer,strlen(logbuffer));
    (void)write(fd,"\n",1);
    (void)close(fd);
}
if(log_message_type == ERROR || log_message_type == NOENCONTRADO ||
log_message_type == PROHIBIDO) exit(3);
}

```

6.2 Código para Cookie

Para la Cookie no nos hemos enrevesado mucho. Simplemente lo que hacemos es cuando llega una petición correcta introducimos la Cookie a “piñón” y desde ahí el propio navegador en posteriores peticiones ya la incluye.

```

(void)sprintf(buffer,"HTTP/1.0 200 OK\r\nSet-Cookie: name=Luis-Cookie;
expires=Sat, 03 May 2025 17:44:22 GMT\r\nContent-Type: %s\r\n\r\n", fstr);

```

```

Fichero abierto correctamente
Se ha procesado todo correctamente, hacer nuevas peticiones
Recibido: GET /favicon.ico HTTP/1.1
Host: localhost:8189
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36
Accept: /*
Referer: http://localhost:8189/ejemplo.htm
Accept-Encoding: gzip, deflate, sdch
Accept-Language: es,de-DE;q=0.8,de;q=0.6,en;q=0.4
Cookie: session_id=10b8cc362cd7cc8c887edfc4f3a820cf8f8173f; name=Luis-Cookie

```

Ilustración 89: Cookie insertada

7. CONCLUSIONES Y VALORACIÓN PERSONAL

7.1 Conclusiones

Como conclusión de la práctica quiero destacar que con una buena guía e instrucciones apropiadas no es muy complicado montar un servidor propio. De esta forma también aprendemos de una forma práctica como funciona todo aquello que solemos usar a diario en nuestros ordenadores, como es todo el entorno de Servicios Telemáticos.

7.2 Valoración Personal

La realización de esta práctica se ha considerado muy útil a la vez que entretenida, puesto que gracias a ello se ha sido capaz de montar un completo Servidor que monta una gran serie de Servicios Telemáticos. Gracias a ella, no solo somos capaces de montar un Servidor, si no un completo entorno de desarrollo web por si se decide uno a dedicarse a esta rama de la Informática.

Me gustaría destacar que la asignatura cuenta con una completísima compenetración en cuanto al desarrollo de la parte teórica como la práctica, siendo su enseñanza completamente paralela y retroalimentándose ambas entre sí, haciendo que sea mucho más sencilla su comprensión. De esta forma hace que la elaboración del trabajo práctico sea sencillo.

Luis Miguel García Macías.

A handwritten signature in blue ink, appearing to read "Luis Miguel García Macías".