Documentación

Servicios Telemáticos

José Miguel Sánchez Almagro

Grupo 2.1

20 de mayo de 2019

Introducción

Para repasar los contenidos teóricos de la asignatura existe esta práctica. Enseña al alumno cómo funcionan los principales servicios web, y éste tendrá que enfrentarse a un proceso de instalación y configuración. Además, se incluye una parte de programación en la que se tendrá que desarrollar un servicio web de juguete, imitando el funcionamiento de soluciones como Apache.

La práctica se divide en apartados, y cada uno se corresponde con un servicio distinto:

Web-SSTT HTTP Server

Esta es la parte más extensa y compleja de la práctica. Como se ha mencionado anteriormente, el objetivo final será tener en funcionamiento un proceso que reciba peticiones HTTP (desde el navegador, por ejemplo) y las gestione. Solo se trabajará con peticiones GET, y finalmente el servicio debe ser capaz de proporcionar una página cuando se pida. Por otro lado, se debe disponer de una gestión adecuada de errores y un funcionamiento eficiente.

Servicio DNS

Los siguientes servicios de la práctica trabajarán sobre un dominio llamado sstt9721.org en mi caso. Para que el cliente pueda interactuar con dicho dominio, este debe estar disponible desde un servidor DNS. En esta área, se instalará la herramienta más extendida en los servidores DNS y se configurará para que funcione en nuestro ámbito.

Servicio SMTP/POP

El primer servicio que funcionará bajo el dominio mencionado anteriormente será un servidor de correo. Se registrarán dos usuarios que podrán intercambiar mensajes entre si. El envío de mensajes se producirá con el protocolo SMTP y la recuperación de los mensajes con POP3.

Servicio HTTP/HTTPS

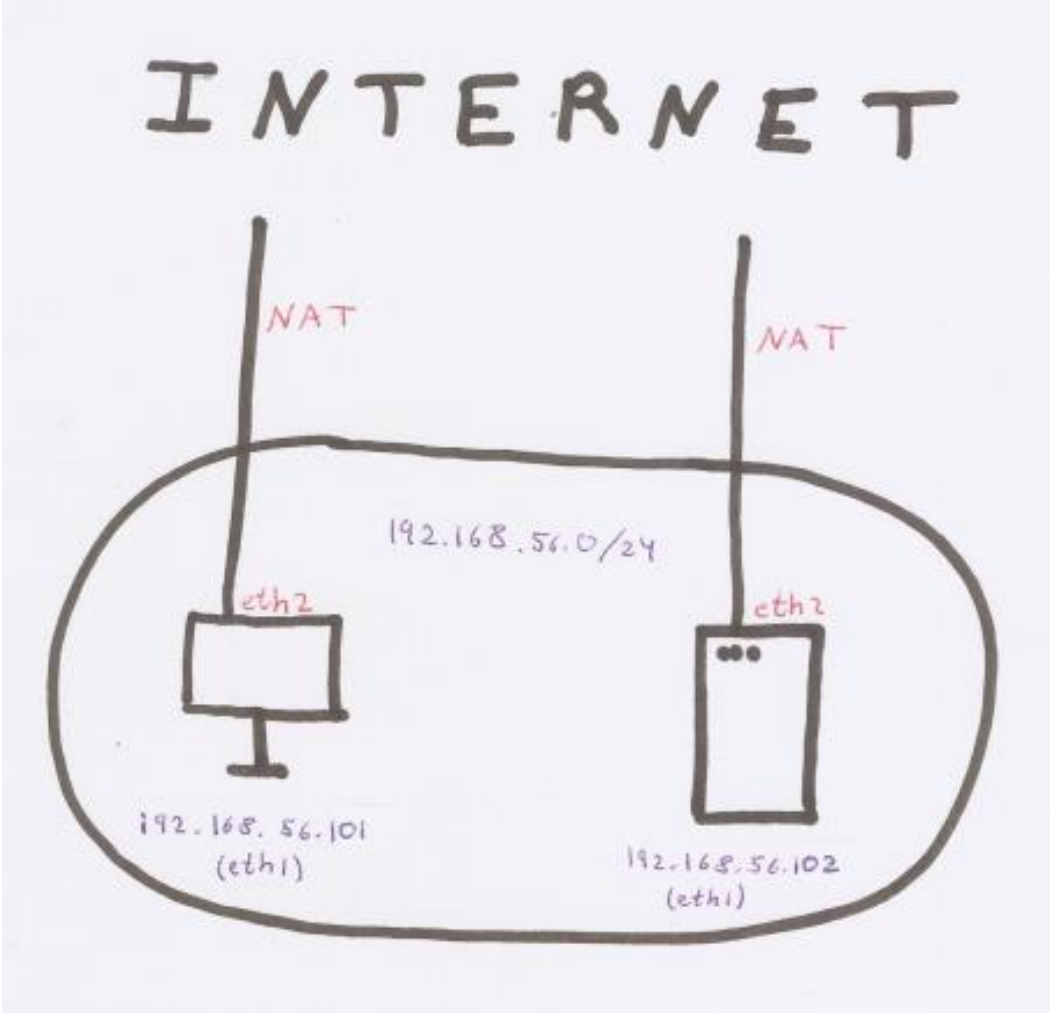
En el primer punto veíamos que se ha de programar un servicio que simule a Apache. En este punto instalaremos Apache y navegaremos sobre sus opciones más superficiales. Una vez el servicio esté funcionando debe ser capaz de proporcionar una página web, tanto con el protocolo HTTP como con HTTPS. Cuando se use HTTPS el servidor enviará un certificado al cliente, y el cliente enviará otro al servidor. Ambos certificados estarán validados por una Autoridad de Certificación creada por nosotros mismos.

IPsec

El objetivo de Ipsec es proteger las comunicaciones que se produzcan entre dos hosts. Puede protegerse la información desde que sale de un host hasta que llega al otro (transporte), o puede protegerse desde que sale de la red de un host hasta que llega a la red del otro (túnel). En este caso nos encargaremos de proteger la comunicación mediante túnel, y solo protegeremos la integridad de la comunicación, es decir, evitaremos que algún host se haga pasar por otro. No se cifrará la comunicación, por lo que se tratará de un protocolo AH.

Descripción del escenario desarrollado y versiones de software

El escenario estará formado por un cliente y un servidor conectados a la misma red local (192.168.56.0/24). Ambos estarán conectados a Internet mediante una red NAT. En el siguiente dibujo se puede observar lo descrito:



El cliente tiene instalado el navegador web Firefox y el programa Wireshark en una versión actualizada.

El servidor tiene las herramientas que se necesitan para poner en marcha los servicios descritos anteriormente:

- **Web-SSTT HTTP Server**. No se necesita nada, simplemente tener instaladas las librerías por defecto de C.
- **Servicio DNS**. Para poner en marcha el servicio DNS se ha usado *BIND (Berkeley Internet Name Domain software)*, que es el software para servidores DNS más usado en todo el mundo. Se ha usado en la versión 9.10.3.
- **Servicio SMTP/POP**. Para usar SMPT se ha instalado *exim4* en su versión 4.86_2. Para el uso de POP se usa *dovecot-pop3d* en su versión 2.2.22.
- **Servicio HTTP/HTTPS**. Para montar el servidor HTTP/HTTPS se utiliza Apache en su versión 2.4.18. También se ha necesitado instalar *php7.0* y *libapache2-mod-php7.0*. Para la seguridad necesaria con HTTPS se utiliza Openssl 1.0.2g-1ubuntu4.15.
- **Ipsec**. Hay muchas herramientas para poner en marcha un servicio de IPsec. En la práctica se ha optado por utilizar Strongswan en la versión U5.3.5/K4.4.0-59-generic.

Todos los paquetes y programas mencionados anteriormente usan a su vez otros paquetes como dependencias que no han sido mencionados.

Descripción de las configuraciones, destacando las opciones de configuración más relevantes

Todo lo mencionado en este apartado pertenece al lado del servidor.

Servicio DNS

Cuando se produzca una consulta DNS que se encuentre fuera del ámbito del dominio sstt9721.org y que el servidor no pueda resolver buscando en Internet, redirigirá dicha consulta a los servidores DNS de la Universidad de Murcia. Para realizar esto se ha modificado el archivo */etc/bind/named.conf.options* añadiendo la opción *forwarders* con las IPs de los servidores DNS de la universidad.

Para que el DNS gestione el dominio mencionado anteriormente se ha creado una zona en el archivo */etc/bind/named.conf.local*, que solo permite las solicitudes de los equipos situados en la red 192.168.56.0/24. Esta zona se encontrará descrita en el archivo */etc/bind/db.sstt9721.org.zone*. En este archivo se encuentran todos los registros (NS, MX y A) que necesitan los servicios de la práctica para funcionar correctamente.

Finalmente, para que el cliente utilice el servicio DNS del servidor, en el fichero */etc/resolv.conf* de la máquina del cliente se ha comentado lo que había anteriormente y se ha añadido la siguiente línea al final: "nameserver 192.168.56.102". Esto hace que cada vez que el cliente quiera realizar una consulta DNS la haga con el servidor que hemos configurado.

Servicio SMTP/POP

Para utilizar SMTP se han modificado algunos parámetros de la configuración del programa. Se ha indicado que el nombre del sistema de correo es sstt9721.org, que se puede recibir correo solo de cuentas de nuestro sistema, que se puede reenviar el correo a usuarios de nuestro sistema o del correo de la Universidad de Murcia, pero que estén en la red local del servidor, y por último, que el correo se almacenará en el directorio Maildir de la carpeta personal de cada usuario.

En el servicio que POP se han activado dos opciones que permiten la autenticación con contraseñas en texto plano en el fichero */etc/dovecot/conf.d/10.auth.conf*. Esto es impensable en sistemas finales, pero en este caso lo necesitamos para poder ver los mensajes que se intercambian con el software Wireshark. Además, al igual que ocurrió con el servicio SMTP, hemos de indicarle a POP que el correo se almacenará en el directorio Maildir. Esto se ha hecho cambiando el valor de la variable "mail_location" en el archivo */etc/dovecot/conf.d/10.mail.conf*.

Servicio HTTP/HTTPS

Se han creado dos Virtual Host, y ambos están en funcionamiento: "servicio-practica-final" y "servicio-practica-final-ssl". Podría haberse creado un único fichero que contuviera a ambos servicios (HTTP y HTTPS), pero así se separaban ambos conceptos y era más fácil de realizar y comprender.

En los archivos de configuración de ambos Virtual Host, */etc/apache2/sites-enabled/servicio-practica-final.conf* y */etc/apache2/sites-enabled/servicio-practica-final-ssl.conf*, se han cambiado los siguientes campos con los mismos valores: ServerAdmin, ServerName, DocumentRoot, y Directory. En el archivo del servicio HTTP se ha indicado que escucha en el puerto 80 y en el del servicio HTTPS, en el puerto 443. Además, en el fichero ...-*ssl*, se han introducido campos que indican donde se encuentran los ficheros con los certificados y claves necesarias, además de otros campos que obligan al cliente a identificarse.

Se ha necesitado por tanto generar certificados que autentiquen al servidor y al cliente. Estos certificados han sido firmados por una autoridad de certificación que nosotros mismos hemos creado. Todos estos certificados se encuentran en el directorio */home/alumno/demoCA*. Para que el cliente pueda utilizar el servicio HTTPS ha tenido que instalar en su navegador la certificación de la autoridad de certificación y el certificado del cliente.

Servicio IPsec

El servicio IPsec ha de configurarse en las dos máquinas que van a comunicarse entre sí, en este caso cliente y servidor. En el archivo */etc/ipsec.conf* se encuentra la configuración del servicio.

Dicho archivo (en cualquiera de los hosts) contiene dos bloques:

El que comienza con 'conn %default' es el bloque que contiene los parámetros para cualquier conexión. En este bloque, hay que prestar especial atención a:

- El valor "ikev2" en la variable "keyexchange", que obliga a IPsec a utilizar IKE en su versión 2 para la generación de claves.
- El parámetro "authby", que tiene el valor "pubkey", para indicar que la autenticación se produce con las claves de los certificados.
- La variable "esp" con "null-sha!". Esto hace que el servicio funcione con autenticación sha pero sin cifrado, es decir, como si usara el protocolo AH.

El siguiente bloque comienza con las palabras 'conn host-host'. Este bloque es específico de la conexión que vamos a utilizar. Aquí indicamos las direcciones IP de ambos hosts, el

certificado del host actual, la información de los hosts almacenada en los certificados, el tipo de conexión que se va a establecer (túnel), y cuando se inicia IKE.

Sin embargo, el archivo no es el mismo en ambas máquinas. Si estamos configurando el archivo en el servidor, los parámetros que comiencen con la palabra 'left' deben tener la información del servidor, y los que comiencen con 'right', la información del cliente. Ocurre lo mismo invirtiendo los sentidos en el cliente.

Además, en los directorios */etc/ipsec.d/cacert*, */etc/ipsec.d/certs* y */etc/ipsec.d/private*, se almacenan el certificado de la autoridad de certificación, el certificado del host, y la clave privada del host, respectivamente. Esto ocurre tanto en el cliente como en el servidor.

Por último, en el fichero */etc/ipsec.secrets* se indica que se usa RSA, y se incluye el nombre del archivo en el que se encuentra la clave privada del host.

Descripción de la implementación del servicio Web-SSTT HTTP

Funciones auxiliares

Se han desarrollado funciones auxiliares que mejoran la legibilidad, estructura y mantenimiento del código. Son las siguientes:

write_estado()

Dado un código de error y un socket, enviará una línea de estado a un cliente a través de dicho socket indicando un error.

write_error()

Enviará al cliente (mediante el socket recibido como parámetro) una página web básica que contiene un mensaje recibido como parámetro que indica un determinado error. Además, enviará la cabecera que corresponda en cada caso.

tratar_error()

Será llamada cuando se produzca un error en el servidor. Se encargará de llamar a las funciones *write_estado()* y *write_error()* indicándoles el tipo de error producido y el socket por el que deben enviar los datos. Dicho error y socket serán pasados como parámetros.

readwrite()

Dado un socket por el que se enviarán los datos al cliente que ha realizado la petición, y otro socket por el que se lee el archivo pedido por el cliente, esta función lee y almacena en un buffer datos del dicho archivo en tandas de 8KB. En cuanto se han leído 8KB de datos se envían al servidor, y se vuelven a leer datos del archivo abierto. Este proceso se produce en bucle hasta que no hay más datos que leer ni enviar.

Funciones principales

El resto del código va a ser analizado siguiendo el camino que se produce desde que llega una petición al servidor hasta que es respondida.

main()

Una vez se ha producido la conexión TCP con el cliente, y antes de procesar la petición, se gestiona la persistencia de dicha conexión. Se establece un tiempo de conexión de diez segundos y se utiliza la función *select()* en un bucle while, que esperará hasta que se reciba una petición desde el socket que se encuentra en el conjunto pasado como parámetro. Si no se recibe nada en diez segundos, se procede a cerrar la conexión. En el caso contrario,

se llama a la función *process_web_request()* para analizar la petición realizada por el cliente.

process_web_request()

En primer lugar, se lee del socket la petición del cliente y se almacena en un buffer.

A continuación, se obtiene la hora actual, se transforma a la hora GMT, y se deja almacenada en otra variable. Será enviada en la cabecera de respuesta al cliente.

Ahora se procede al tratamiento de lo que se ha leído del socket. Se sustituyen todos los caracteres '\r' y '\n' por el carácter '~' para que no den problemas en las funciones que se van a usar a lo largo de todo el código. Esto es realizado con Expresiones Regulares.

En este punto comienza en análisis de la petición. Durante todo este proceso se analiza lo recibido para detectar posibles errores en la petición. Si cualquier error es encontrado se devolverá haciendo uso de la función *tratar_error()* explicada anteriormente. Todo el análisis se realizará en su mayoría haciendo uso de las funciones *strtok()* y *strcmp()*. Entre otras cosas se comprueba:

- Si el método utilizado por el cliente es soportado por el servidor (solo se soporta el método GET).
- Si la versión HTTP usada es correcta.
- Que no se intenta acceder a directorios con acceso no permitido.
- Que los campos de la cabecera tienen una estructura y sintaxis correcta (esto se realiza otra vez haciendo uso de Expresiones Regulares).
- Si el cliente tenía almacenada alguna cookie.
- Que el archivo solicitado existe y su extensión es soportada por el servidor.

Si todo es correcto, el siguiente paso es construir la cookie que se va a enviar al cliente en función de si el cliente tenía almacenada alguna previamente y con qué valor. Estas cookies expirarán a los dos minutos de su creación utilizando el campo 'Expires' de la línea Cookie de las cabeceras HTTP.

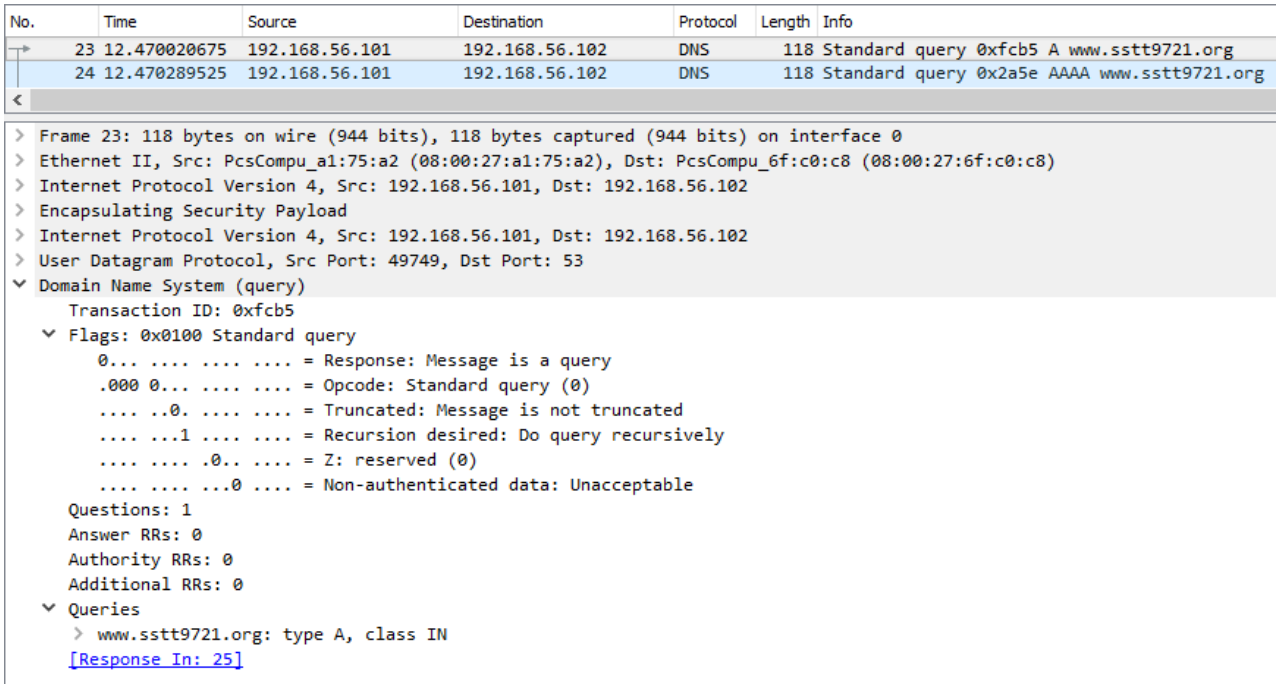
En último lugar: se envía la línea de estado indicando que todo se ha producido correctamente; se abre el archivo solicitado por el cliente, se obtiene su tamaño, y gracias a esto se termina de construir la cabecera y se envía por el socket; y finalmente se envía el archivo utilizando la función definida anteriormente *readwrite()*.

Por último, se cierra el descriptor de fichero que apuntaba al archivo y se termina de procesar la petición, devolviendo el control a la función *main()*, que mantendrá la conexión abierta a la espera de nuevas peticiones.

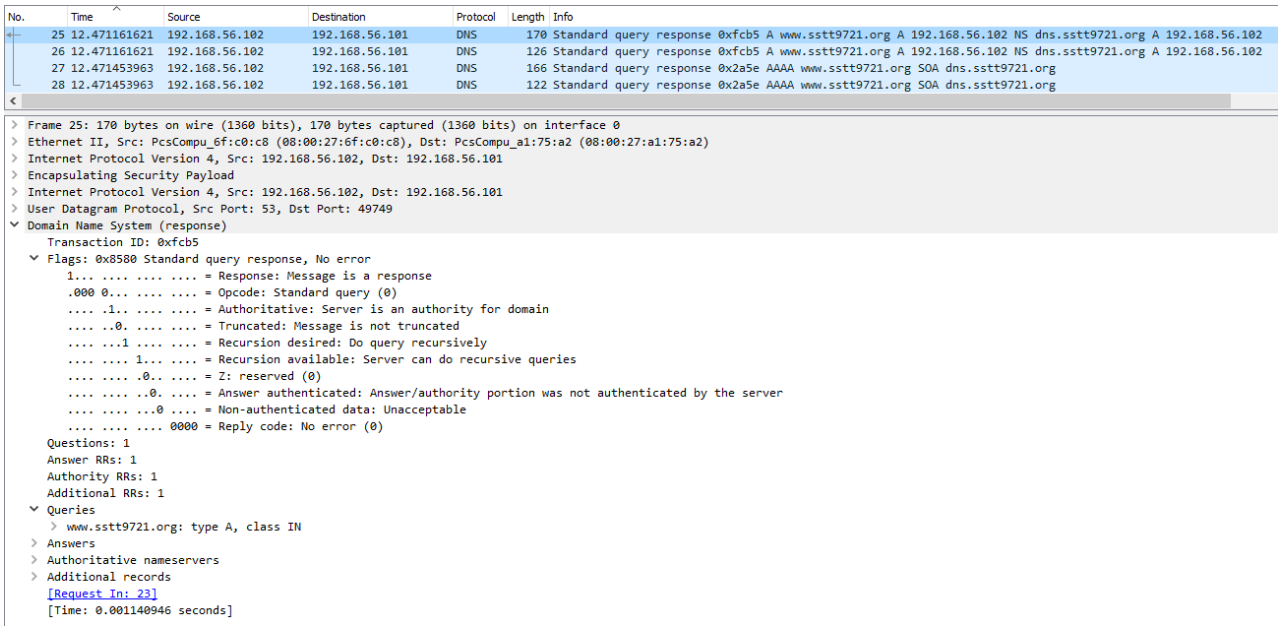
Trazas representativas de los protocolos empleados y explicación de las mismas

Intercambio DNS y el acceso al web cuando se accede a http://www.ssttdni.org

Escribimos http://www.sstt9721.org en la barra de direcciones del navegador, y lo primero que realiza es una consulta DNS para resolver la dirección dada. Quiere saber la dirección IP asociada a la URL. Para ello realiza consultas de tipo A y AAAA. A continuación, podemos ver una imagen de dicho paquete:



El servidor DNS, con la dirección IP de nuestro servidor (192.168.56.102), responde a dichas consulta:

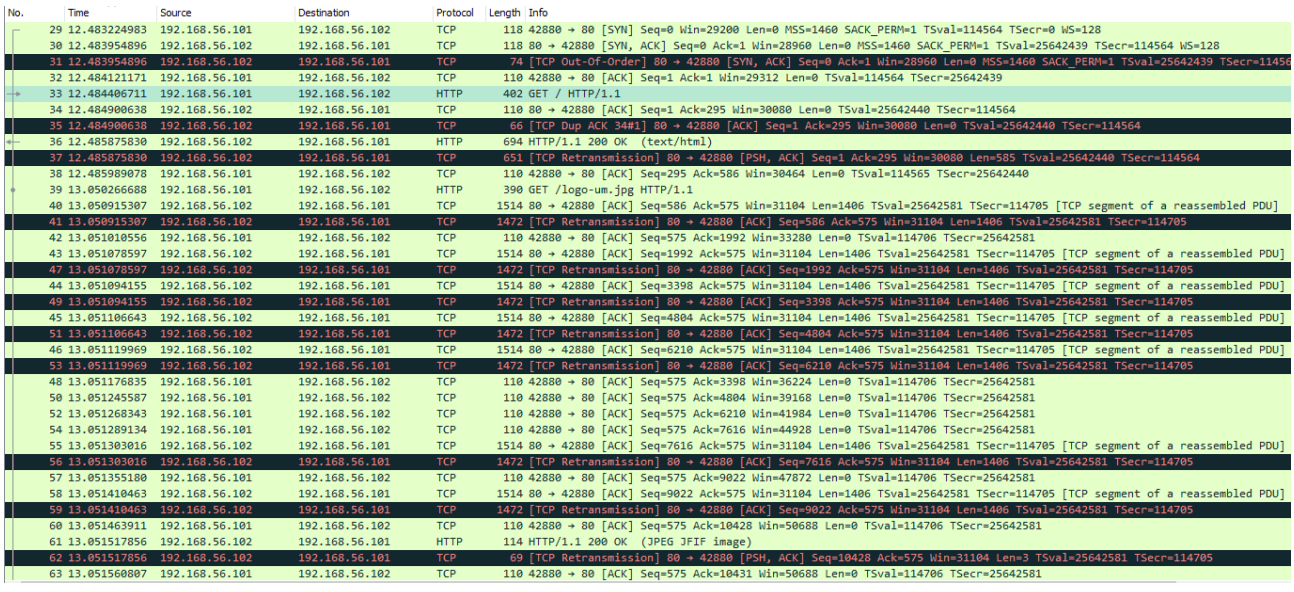


No.	Time	Source	Destination	Protocol	Length	Info
25	12.471161621	192.168.56.102	192.168.56.101	DNS	170	Standard query response 0xfcb5 A www.sstt9721.org A 192.168.56.102 NS dns.sstt9721.org A 192.168.56.102
26	12.471161621	192.168.56.102	192.168.56.101	DNS	126	Standard query response 0xfcb5 A www.sstt9721.org A 192.168.56.102 NS dns.sstt9721.org A 192.168.56.102
27	12.471453963	192.168.56.102	192.168.56.101	DNS	166	Standard query response 0x2a5e AAAA www.sstt9721.org SOA dns.sstt9721.org
28	12.471453963	192.168.56.102	192.168.56.101	DNS	122	Standard query response 0x2a5e AAAA www.sstt9721.org SOA dns.sstt9721.org

```
> Frame 25: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits) on interface 0
> Ethernet II, Src: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8), Dst: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2)
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> User Datagram Protocol, Src Port: 53, Dst Port: 49749
v Domain Name System (response)
    Transaction ID: 0xfcb5
  v Flags: 0x8580 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .1.. .... .... = Authoritative: Server is an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
  v Queries
    > www.sstt9721.org: type A, class IN
  > Answers
  > Authoritative nameservers
  > Additional records
    [Request In: 23]
    [Time: 0.001140946 seconds]
```

En el caso de la consulta A, devuelve la dirección IP correspondiente a la URL por la que se había preguntado, además de proporcionar la URL del servidor DNS asociado al dominio sstt9721 y su dirección IP. En la consulta AAAA solamente devuelve la URL del DNS del dominio mencionado anteriormente.

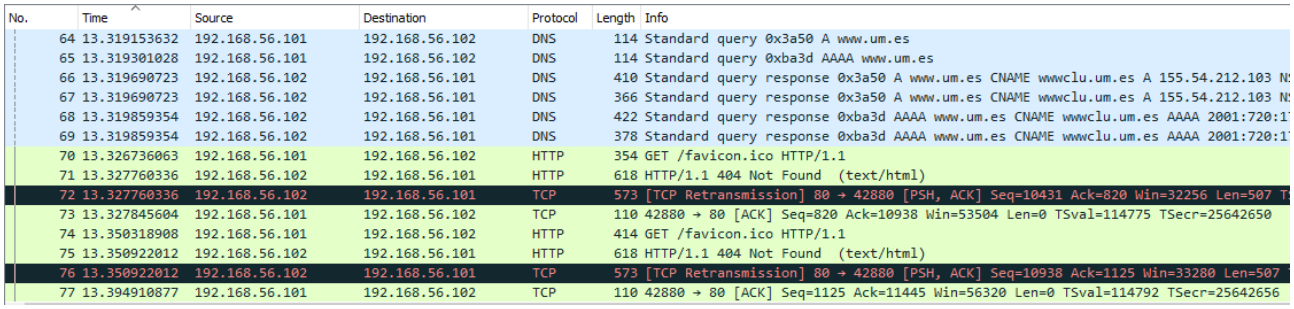
Una vez que la consulta DNS se ha realizado correctamente, se intercambian tres mensajes TCP para establecer una conexión, y una vez que esta se encuentra activa, el cliente realiza una petición GET, donde solicita la página web:



No.	Time	Source	Destination	Protocol	Length	Info
29	12.483224983	192.168.56.101	192.168.56.102	TCP	118	42880 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=114564 TSecr=0 WS=128
30	12.483954896	192.168.56.102	192.168.56.101	TCP	118	80 → 42880 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25642439 TSecr=114564 WS=128
31	12.483954896	192.168.56.102	192.168.56.101	TCP	74	[TCP Out-Of-Order] 80 → 42880 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25642439 TSecr=11456
32	12.484121171	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=114564 TSecr=25642439
33	12.484406711	192.168.56.101	192.168.56.102	HTTP	402	GET / HTTP/1.1
34	12.484900638	192.168.56.102	192.168.56.101	TCP	110	80 → 42880 [ACK] Seq=1 Ack=295 Win=30080 Len=0 TSval=25642440 TSecr=114564
35	12.484900638	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 34#1] 80 → 42880 [ACK] Seq=1 Ack=295 Win=30080 Len=0 TSval=25642440 TSecr=114564
36	12.485875830	192.168.56.102	192.168.56.101	HTTP	694	HTTP/1.1 200 OK (text/html)
37	12.485875830	192.168.56.102	192.168.56.101	TCP	651	[TCP Retransmission] 80 → 42880 [PSH, ACK] Seq=1 Ack=295 Win=30080 Len=585 TSval=25642440 TSecr=114564
38	12.485989078	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=295 Ack=586 Win=30464 Len=0 TSval=114565 TSecr=25642440
39	13.050266688	192.168.56.101	192.168.56.102	HTTP	390	GET /logo-um.jpg HTTP/1.1
40	13.050915307	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=586 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
41	13.050915307	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=586 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
42	13.051010556	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=1992 Win=33280 Len=0 TSval=114706 TSecr=25642581
43	13.051078597	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=1992 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
47	13.051078597	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=1992 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
44	13.051094155	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=3398 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
49	13.051094155	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=3398 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
45	13.051106643	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=4804 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
51	13.051106643	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=4804 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
46	13.051119969	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=6210 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
53	13.051119969	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=6210 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
48	13.051176835	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=3398 Win=36224 Len=0 TSval=114706 TSecr=25642581
50	13.051245587	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=4804 Win=39168 Len=0 TSval=114706 TSecr=25642581
52	13.051268343	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=6210 Win=41984 Len=0 TSval=114706 TSecr=25642581
54	13.051289134	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=7616 Win=44928 Len=0 TSval=114706 TSecr=25642581
55	13.051303016	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=7616 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
56	13.051303016	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=7616 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
57	13.051355180	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=9022 Win=47872 Len=0 TSval=114706 TSecr=25642581
58	13.051410463	192.168.56.102	192.168.56.101	TCP	1514	80 → 42880 [ACK] Seq=9022 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705 [TCP segment of a reassembled PDU]
59	13.051410463	192.168.56.102	192.168.56.101	TCP	1472	[TCP Retransmission] 80 → 42880 [ACK] Seq=9022 Ack=575 Win=31104 Len=1406 TSval=25642581 TSecr=114705
60	13.051463911	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=10428 Win=50688 Len=0 TSval=114706 TSecr=25642581
61	13.051517856	192.168.56.102	192.168.56.101	HTTP	114	HTTP/1.1 200 OK (JPEG JFIF image)
62	13.051517856	192.168.56.102	192.168.56.101	TCP	69	[TCP Retransmission] 80 → 42880 [PSH, ACK] Seq=10428 Ack=575 Win=31104 Len=3 TSval=25642581 TSecr=114705
63	13.051560807	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=575 Ack=10431 Win=50688 Len=0 TSval=114706 TSecr=25642581

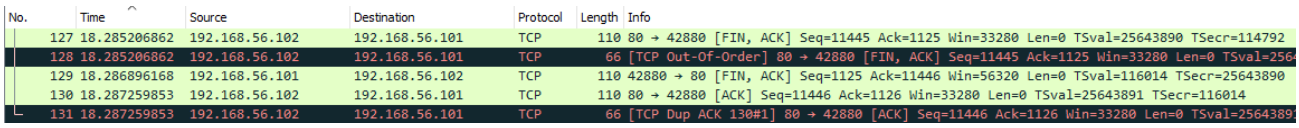
Después, el cliente solicita la imagen que contiene la web, y el cliente se la envía.

En esta última imagen vemos que el cliente vuelve a realizar una petición DNS, esta vez para la dirección www.um.es. Esto es debido a que la página web contiene un enlace a dicha página. Después se solicita el icono favicon.ico, pero el servidor no lo tiene, por lo que le dice al cliente que no se ha encontrado el archivo.



No.	Time	Source	Destination	Protocol	Length	Info
64	13.319153632	192.168.56.101	192.168.56.102	DNS	114	Standard query 0x3a50 A www.um.es
65	13.319301028	192.168.56.101	192.168.56.102	DNS	114	Standard query 0xba3d AAAA www.um.es
66	13.319690723	192.168.56.102	192.168.56.101	DNS	410	Standard query response 0x3a50 A www.um.es CNAME wwwclu.um.es A 155.54.212.103 N
67	13.319690723	192.168.56.102	192.168.56.101	DNS	366	Standard query response 0x3a50 A www.um.es CNAME wwwclu.um.es A 155.54.212.103 N
68	13.319859354	192.168.56.102	192.168.56.101	DNS	422	Standard query response 0xba3d AAAA www.um.es CNAME wwwclu.um.es AAAA 2001:720:1
69	13.319859354	192.168.56.102	192.168.56.101	DNS	378	Standard query response 0xba3d AAAA www.um.es CNAME wwwclu.um.es AAAA 2001:720:1
70	13.326736063	192.168.56.101	192.168.56.102	HTTP	354	GET /favicon.ico HTTP/1.1
71	13.327760336	192.168.56.102	192.168.56.101	HTTP	618	HTTP/1.1 404 Not Found (text/html)
72	13.327760336	192.168.56.102	192.168.56.101	TCP	573	[TCP Retransmission] 80 → 42880 [PSH, ACK] Seq=10431 Ack=820 Win=32256 Len=507 T
73	13.327845604	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=820 Ack=10938 Win=53504 Len=0 TSval=114775 TSecr=25642650
74	13.350318908	192.168.56.101	192.168.56.102	HTTP	414	GET /favicon.ico HTTP/1.1
75	13.350922012	192.168.56.102	192.168.56.101	HTTP	618	HTTP/1.1 404 Not Found (text/html)
76	13.350922012	192.168.56.102	192.168.56.101	TCP	573	[TCP Retransmission] 80 → 42880 [PSH, ACK] Seq=10938 Ack=1125 Win=33280 Len=507
77	13.394910877	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [ACK] Seq=1125 Ack=11445 Win=56320 Len=0 TSval=114792 TSecr=25642656

Finalmente, el servidor cierra la conexión TCP con el cliente, como podemos ver en estos mensajes:

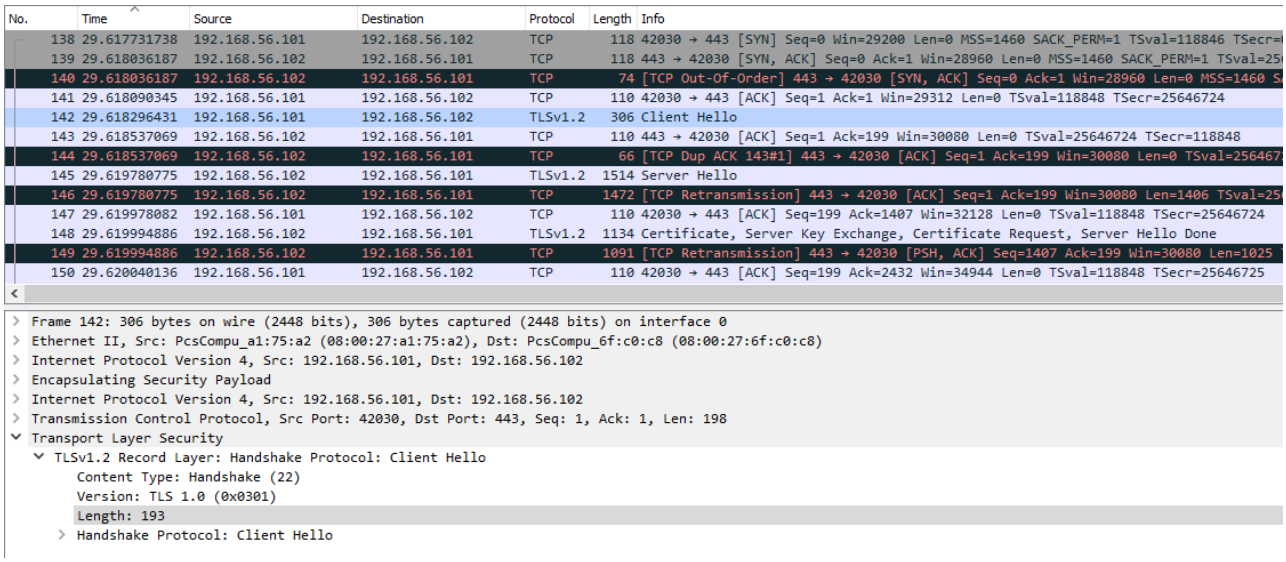


No.	Time	Source	Destination	Protocol	Length	Info
127	18.285206862	192.168.56.102	192.168.56.101	TCP	110	80 → 42880 [FIN, ACK] Seq=11445 Ack=1125 Win=33280 Len=0 TSval=25643890 TSecr=114792
128	18.285206862	192.168.56.102	192.168.56.101	TCP	66	[TCP Out-Of-Order] 80 → 42880 [FIN, ACK] Seq=11445 Ack=1125 Win=33280 Len=0 TSval=256
129	18.286896168	192.168.56.101	192.168.56.102	TCP	110	42880 → 80 [FIN, ACK] Seq=1125 Ack=11446 Win=56320 Len=0 TSval=116014 TSecr=25643890
130	18.287259853	192.168.56.102	192.168.56.101	TCP	110	80 → 42880 [ACK] Seq=11446 Ack=1126 Win=33280 Len=0 TSval=25643891 TSecr=116014
131	18.287259853	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 130#1] 80 → 42880 [ACK] Seq=11446 Ack=1126 Win=33280 Len=0 TSval=2564389

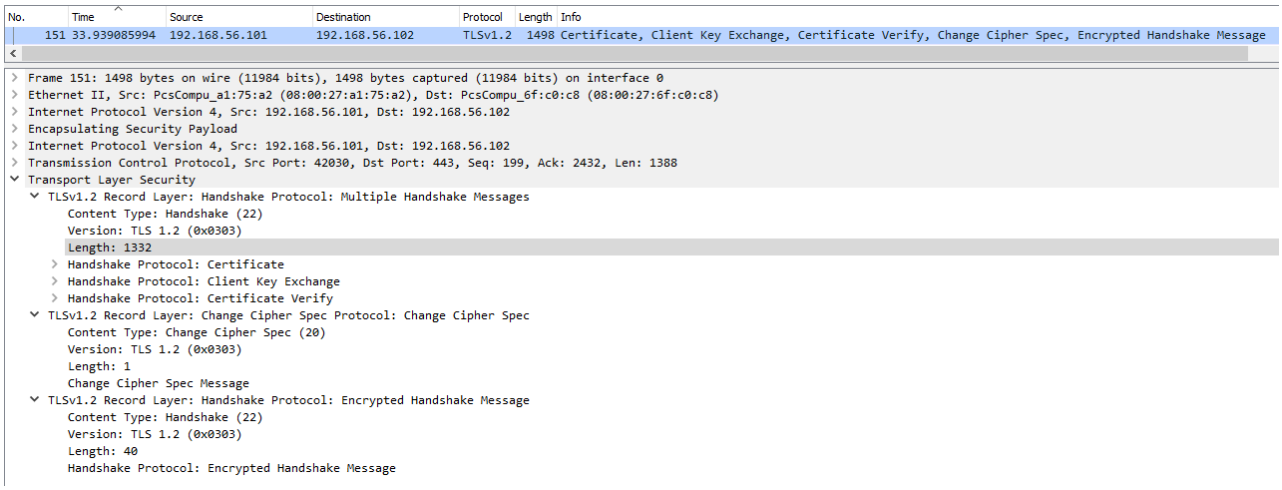
Intercambio DNS y el acceso al web seguro https://www.ssttdni.org

Al principio debería producirse el mismo intercambio DNS que hemos visto en el caso anterior, pero gracias a la caché del navegador el cliente ya conoce la IP de la dirección https://www.sstt9721.org, por lo que no necesita hacer una solicitud al servidor DNS. Lo que si se produce es la comunicación para establecer la conexión TCP.

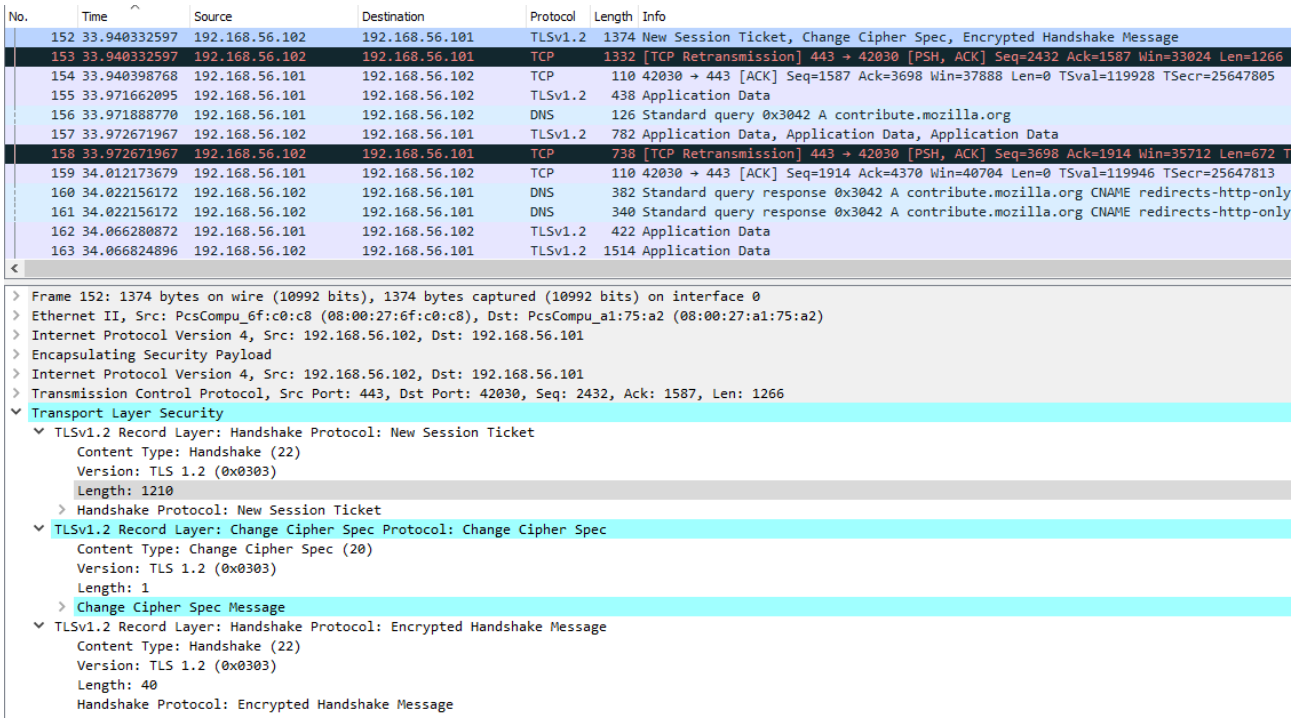
En el caso anterior, el navegador realizaba una petición HTTP GET al servidor. En este caso, utiliza el protocolo TLS y envía un mensaje "Client Hello" al servidor. Este le contesta con un "Server Hello", y le envía su certificado.



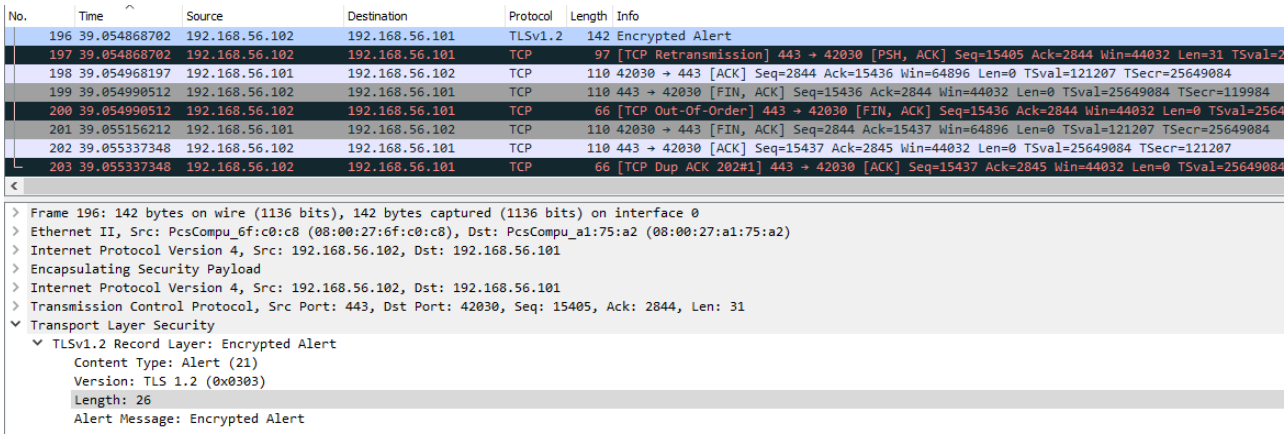
El cliente procede entonces a verificar el certificado, y la Autoridad de Certificación que lo ha firmado, para saber si puede "fiarse" de él. Como insertamos en el navegador del cliente el certificado de la Autoridad de Certificación, el cliente toma por válido el certificado del servidor y envía el suyo al mismo, además del resto de mensajes que se comparten al establecerse una conexión TLS.



El servidor comprueba el certificado del cliente, y como está firmado por una CA en la que confía, autoriza la conexión, enviando un *ticket* de conexión al cliente. A partir de ese momento, comienzan a comunicarse de manera similar a como lo hacía el protocolo HTTP del apartado anterior.



Finalmente, el servidor envía una alerta al cliente y cierra la conexión TCP:



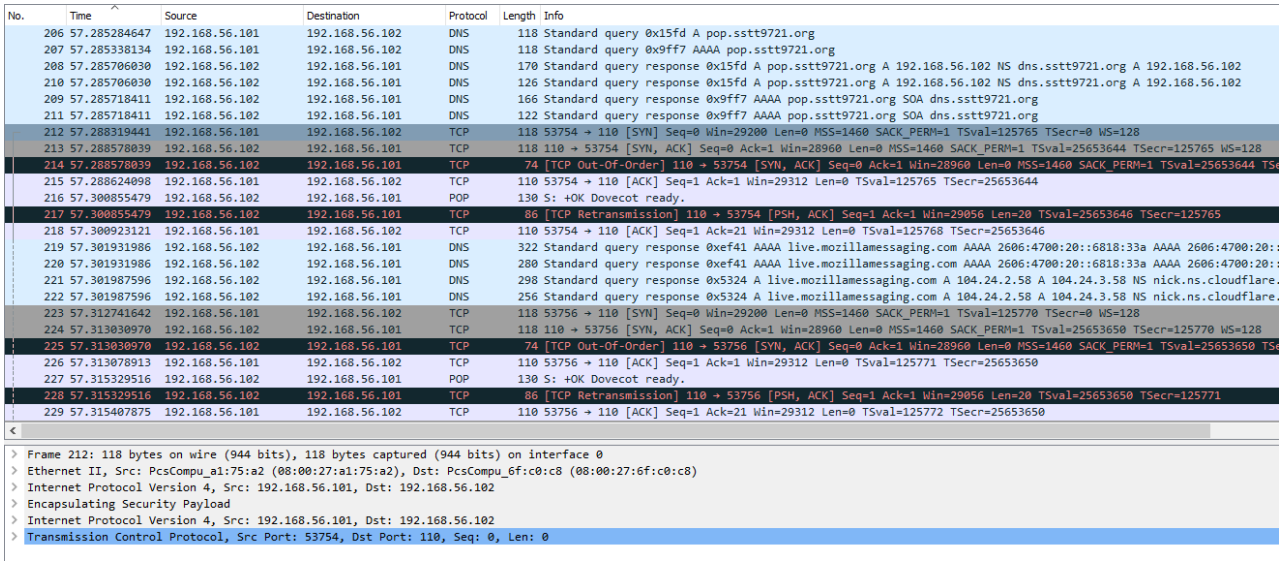
No.	Time	Source	Destination	Protocol	Length	Info
196	39.054868702	192.168.56.102	192.168.56.101	TLSv1.2	142	Encrypted Alert
197	39.054868702	192.168.56.102	192.168.56.101	TCP	97	[TCP Retransmission] 443 → 42030 [PSH, ACK] Seq=15405 Ack=2844 Win=44032 Len=31 TSval=2
198	39.054968197	192.168.56.101	192.168.56.102	TCP	110	42030 → 443 [ACK] Seq=2844 Ack=15436 Win=64896 Len=0 TSval=121207 TSecr=25649084
199	39.054990512	192.168.56.102	192.168.56.101	TCP	110	443 → 42030 [FIN, ACK] Seq=15436 Ack=2844 Win=44032 Len=0 TSval=25649084 TSecr=119984
200	39.054990512	192.168.56.102	192.168.56.101	TCP	66	[TCP Out-Of-Order] 443 → 42030 [FIN, ACK] Seq=15436 Ack=2844 Win=44032 Len=0 TSval=2564
201	39.055156212	192.168.56.101	192.168.56.102	TCP	110	42030 → 443 [FIN, ACK] Seq=2844 Ack=15437 Win=64896 Len=0 TSval=121207 TSecr=25649084
202	39.055337348	192.168.56.102	192.168.56.101	TCP	110	443 → 42030 [ACK] Seq=15437 Ack=2845 Win=44032 Len=0 TSval=25649084 TSecr=121207
203	39.055337348	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 202#1] 443 → 42030 [ACK] Seq=15437 Ack=2845 Win=44032 Len=0 TSval=2564908

```
> Frame 196: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0
> Ethernet II, Src: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8), Dst: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2)
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> Transmission Control Protocol, Src Port: 443, Dst Port: 42030, Seq: 15405, Ack: 2844, Len: 31
v Transport Layer Security
  v TLSv1.2 Record Layer: Encrypted Alert
      Content Type: Alert (21)
      Version: TLS 1.2 (0x0303)
      Length: 26
      Alert Message: Encrypted Alert
```

Intercambios DNS, SMTP y POP

Cuando se abre el cliente de correo Thunderbird, éste nos pide que introduzcamos la contraseña de los usuarios que tienen cuentas vinculadas a dicho cliente. Procede entonces a realizar una autenticación con el cliente de correo POP3.

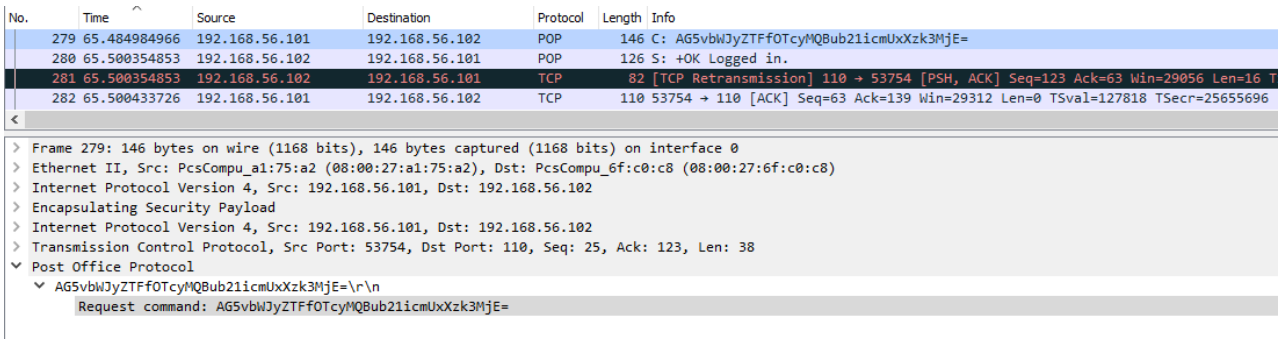
Para ello, en primer lugar realiza una consulta DNS preguntando por el servidor pop.sstt9721.org. A continuación, establece una conexión TCP para cada usuario, y el servidor POP3 le contesta con un mensaje OK, indicando que está listo:



No.	Time	Source	Destination	Protocol	Length	Info
206	57.285284647	192.168.56.101	192.168.56.102	DNS	118	Standard query 0x15fd A pop.sstt9721.org
207	57.285338134	192.168.56.101	192.168.56.102	DNS	118	Standard query 0x9ff7 AAAA pop.sstt9721.org
208	57.285706030	192.168.56.102	192.168.56.101	DNS	170	Standard query response 0x15fd A pop.sstt9721.org A 192.168.56.102 NS dns.sstt9721.org A 192.168.56.102
210	57.285706030	192.168.56.102	192.168.56.101	DNS	126	Standard query response 0x15fd A pop.sstt9721.org A 192.168.56.102 NS dns.sstt9721.org A 192.168.56.102
209	57.285718411	192.168.56.102	192.168.56.101	DNS	166	Standard query response 0x9ff7 AAAA pop.sstt9721.org SOA dns.sstt9721.org
211	57.285718411	192.168.56.102	192.168.56.101	DNS	122	Standard query response 0x9ff7 AAAA pop.sstt9721.org SOA dns.sstt9721.org
212	57.288319441	192.168.56.101	192.168.56.102	TCP	118	53754 → 110 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=125765 TSecr=0 WS=128
213	57.288578039	192.168.56.102	192.168.56.101	TCP	118	110 → 53754 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25653644 TSecr=125765 WS=128
214	57.288578039	192.168.56.102	192.168.56.101	TCP	74	[TCP Out-Of-Order] 110 → 53754 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25653644 TSe
215	57.288624098	192.168.56.101	192.168.56.102	TCP	110	53754 → 110 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=125765 TSecr=25653644
216	57.300855479	192.168.56.102	192.168.56.101	POP	130	S: +OK Dovecot ready.
217	57.300855479	192.168.56.102	192.168.56.101	TCP	86	[TCP Retransmission] 110 → 53754 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=20 TSval=25653646 TSecr=125765
218	57.300923121	192.168.56.101	192.168.56.102	TCP	110	53754 → 110 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=125768 TSecr=25653646
219	57.301931986	192.168.56.102	192.168.56.101	DNS	322	Standard query response 0xef41 AAAA live.mozillamessaging.com AAAA 2606:4700:20::6818:33a AAAA 2606:4700:20::
220	57.301931986	192.168.56.102	192.168.56.101	DNS	280	Standard query response 0xef41 AAAA live.mozillamessaging.com AAAA 2606:4700:20::6818:33a AAAA 2606:4700:20::
221	57.301987596	192.168.56.102	192.168.56.101	DNS	298	Standard query response 0x5324 A live.mozillamessaging.com A 104.24.2.58 A 104.24.3.58 NS nick.ns.cloudflare.
222	57.301987596	192.168.56.102	192.168.56.101	DNS	256	Standard query response 0x5324 A live.mozillamessaging.com A 104.24.2.58 A 104.24.3.58 NS nick.ns.cloudflare.
223	57.312741642	192.168.56.101	192.168.56.102	TCP	118	53756 → 110 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=125770 TSecr=0 WS=128
224	57.313030970	192.168.56.102	192.168.56.101	TCP	118	110 → 53756 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25653650 TSecr=125770 WS=128
225	57.313030970	192.168.56.102	192.168.56.101	TCP	74	[TCP Out-Of-Order] 110 → 53756 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25653650 TSe
226	57.313078913	192.168.56.101	192.168.56.102	TCP	110	53756 → 110 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=125771 TSecr=25653650
227	57.315329516	192.168.56.102	192.168.56.101	POP	130	S: +OK Dovecot ready.
228	57.315329516	192.168.56.102	192.168.56.101	TCP	86	[TCP Retransmission] 110 → 53756 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=20 TSval=25653650 TSecr=125771
229	57.315407875	192.168.56.101	192.168.56.102	TCP	110	53756 → 110 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=125772 TSecr=25653650

```
> Frame 212: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
> Ethernet II, Src: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2), Dst: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8)
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Transmission Control Protocol, Src Port: 53754, Dst Port: 110, Seq: 0, Len: 0
```

A continuación, establece otra conexión TCP para que los usuarios se autentiquen. Se produce entonces un intercambio de mensajes con ese fin, en el que encontramos el envío de claves en mensajes como el siguiente:



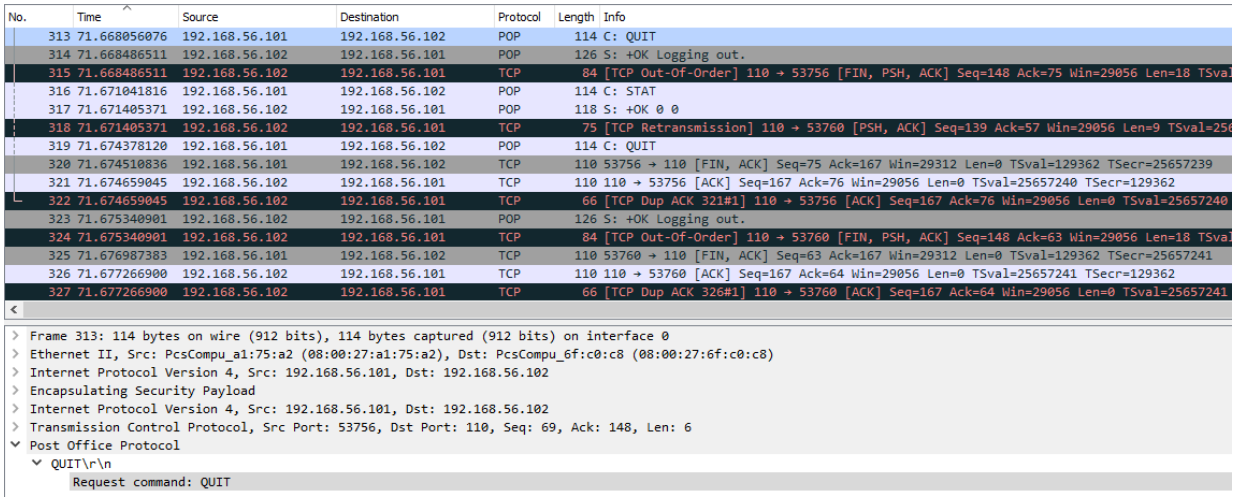
No.	Time	Source	Destination	Protocol	Length	Info
279	65.484984966	192.168.56.101	192.168.56.102	POP	146	C: AG5vbWJyZTFfOTcyMQBub21icmUxXzk3MjE=
280	65.500354853	192.168.56.102	192.168.56.101	POP	126	S: +OK Logged in.
281	65.500354853	192.168.56.102	192.168.56.101	TCP	82	[TCP Retransmission] 110 → 53754 [PSH, ACK] Seq=123 Ack=63 Win=29056 Len=16 T
282	65.500433726	192.168.56.101	192.168.56.102	TCP	110	53754 → 110 [ACK] Seq=63 Ack=139 Win=29312 Len=0 TSval=127818 TSecr=25655696

```
> Frame 279: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 0
> Ethernet II, Src: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2), Dst: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8)
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Transmission Control Protocol, Src Port: 53754, Dst Port: 110, Seq: 25, Ack: 123, Len: 38
v Post Office Protocol
  v AG5vbWJyZTFfOTcyMQBub21icmUxXzk3MjE=\r\n
      Request command: AG5vbWJyZTFfOTcyMQBub21icmUxXzk3MjE=
```

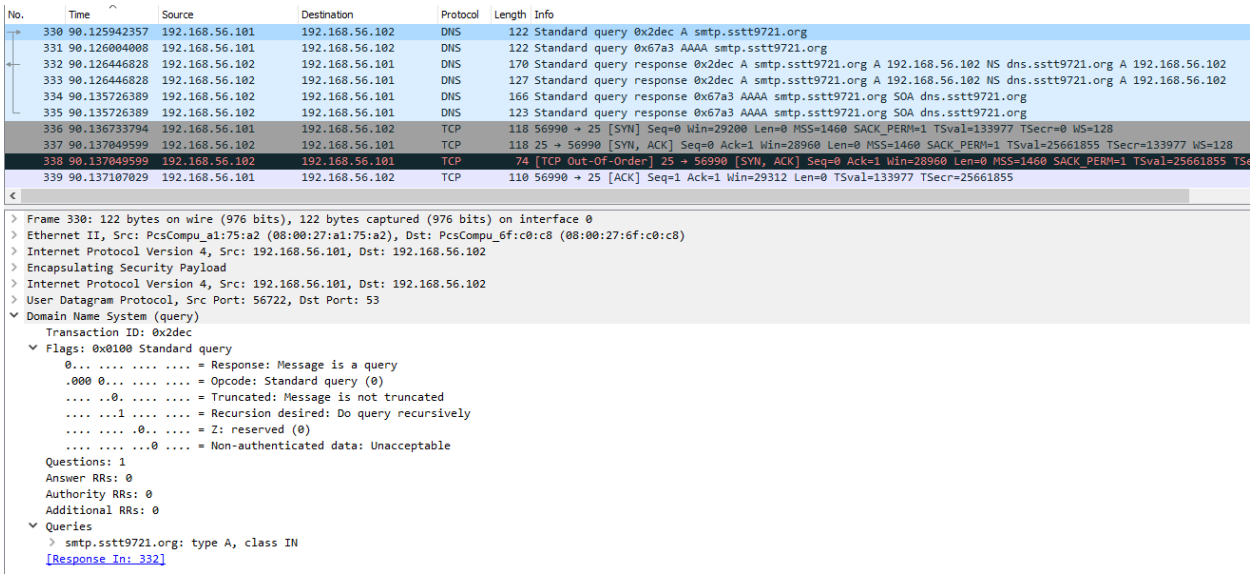
Como podemos observar, las claves se envían en claro, tal y como habíamos configurado previamente. Aunque ahí vemos un galimatías de caracteres, vemos que la cadena termina con el carácter '=', y esto es un indicador de que lo que estamos viendo es texto codificado en base64. Para poder ver el texto, procedemos a descodificarlo (hay muchas

herramientas en Internet que lo realizan), y entonces vemos la clave que mencionábamos anteriormente, con su correspondiente usuario:

Finalmente, se autentican ambos usuarios y se cierran las sesiones POP3 y las conexiones TCP:



El usuario ‘nombre1_9721’ va ahora a enviar un correo al usuario ‘nombre2_9721’. En este caso la consulta DNS se realiza sobre el servidor de correo smpt.sstt9721.org. La respuesta del servidor la vemos a continuación:

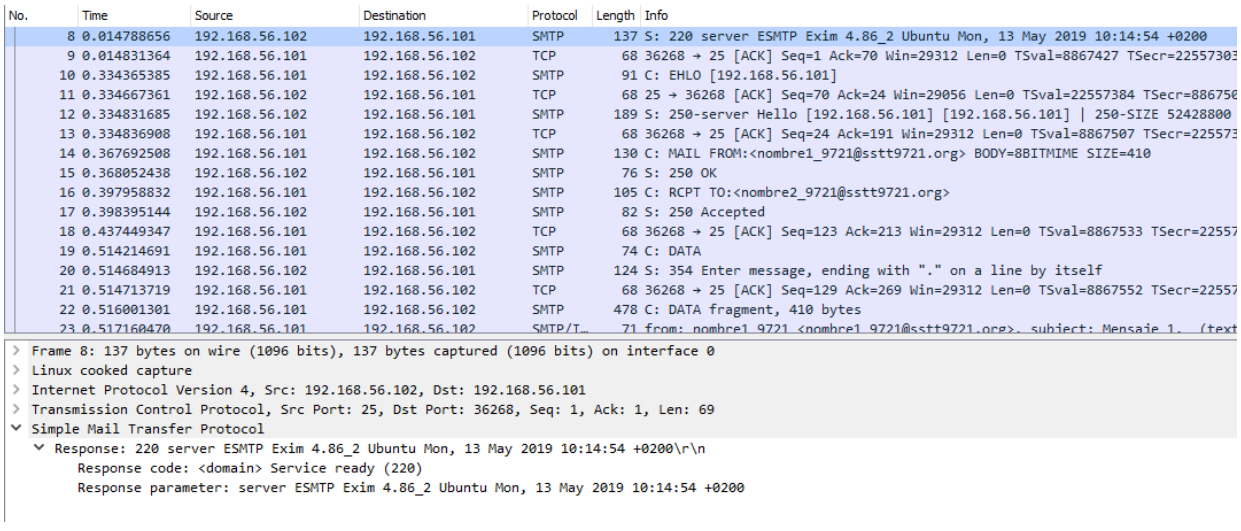


No.	Time	Source	Destination	Protocol	Length	Info
330	90.125942357	192.168.56.101	192.168.56.102	DNS	122	Standard query 0x2dec A smtp.sstt9721.org
331	90.126004008	192.168.56.101	192.168.56.102	DNS	122	Standard query 0x67a3 AAAA smtp.sstt9721.org
332	90.126446828	192.168.56.102	192.168.56.101	DNS	170	Standard query response 0x2dec A smtp.sstt9721.org A 192.168.56.102 NS dns.sstt9721.org A 192.168.56.102
333	90.126446828	192.168.56.102	192.168.56.101	DNS	127	Standard query response 0x2dec A smtp.sstt9721.org A 192.168.56.102 NS dns.sstt9721.org A 192.168.56.102
334	90.135726389	192.168.56.102	192.168.56.101	DNS	166	Standard query response 0x67a3 AAAA smtp.sstt9721.org SOA dns.sstt9721.org
335	90.135726389	192.168.56.102	192.168.56.101	DNS	123	Standard query response 0x67a3 AAAA smtp.sstt9721.org SOA dns.sstt9721.org
336	90.136733794	192.168.56.101	192.168.56.102	TCP	118	56990 → 25 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=133977 TSecr=0 WS=128
337	90.137049599	192.168.56.102	192.168.56.101	TCP	118	25 → 56990 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25661855 TSecr=133977 WS=128
338	90.137049599	192.168.56.102	192.168.56.101	TCP	74	[TCP Out-Of-Order] 25 → 56990 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25661855 TS
339	90.137107029	192.168.56.101	192.168.56.102	TCP	110	56990 → 25 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=133977 TSecr=25661855

```
> Frame 330: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
> Ethernet II, Src: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2), Dst: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8)
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> User Datagram Protocol, Src Port: 56722, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x2dec
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > smtp.sstt9721.org: type A, class IN
    [Response In: 332]
```

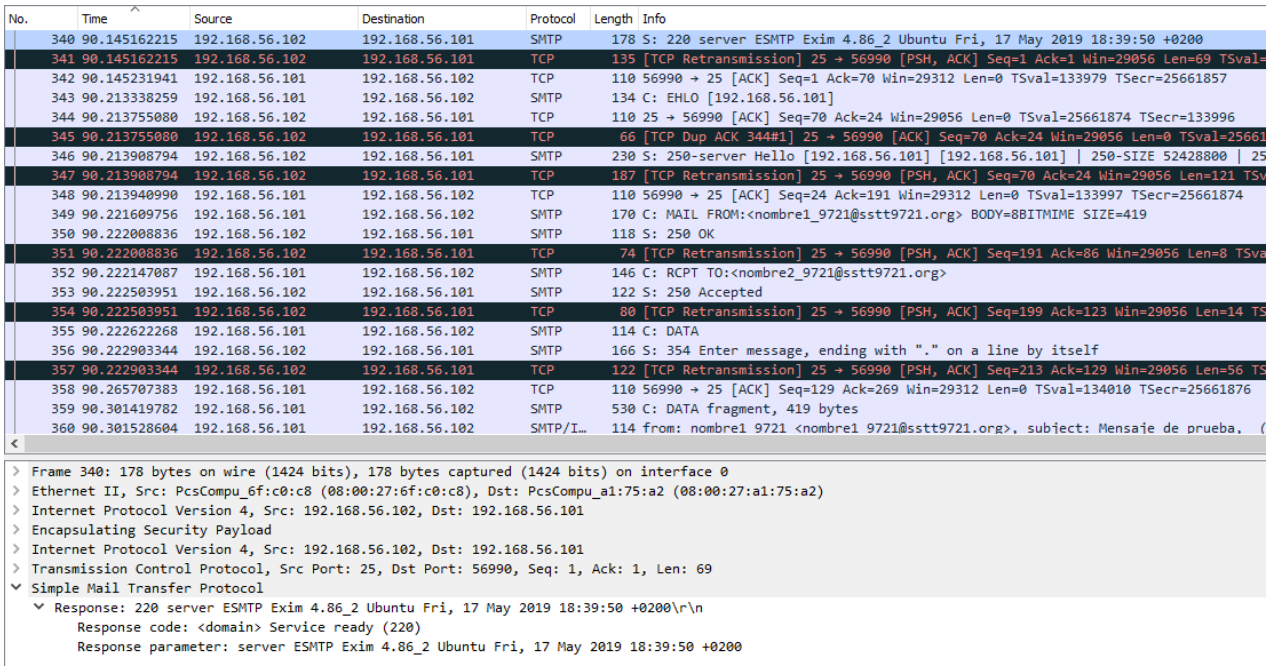
Al igual que ocurría anteriormente, el servidor nos proporciona la IP del servicio de correo y la dirección del servidor DNS del dominio sstt9721.org. Inmediatamente después, se establece la conexión TCP en el puerto 25, correspondiente al servicio SMPT.

En este punto, el servidor sabe que el cliente ha iniciado una conexión SMTP, por lo que responde enviando información.



No.	Time	Source	Destination	Protocol	Length	Info
8	0.014788656	192.168.56.102	192.168.56.101	SMTP	137	S: 220 server ESMTP Exim 4.86_2 Ubuntu Mon, 13 May 2019 10:14:54 +0200
9	0.014831364	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=1 Ack=70 Win=29312 Len=0 TSval=8867427 TSecr=22557303
10	0.334365385	192.168.56.101	192.168.56.102	SMTP	91	C: EHLO [192.168.56.101]
11	0.334667361	192.168.56.102	192.168.56.101	TCP	68	25 → 36268 [ACK] Seq=70 Ack=24 Win=29056 Len=0 TSval=22557384 TSecr=886750
12	0.334831685	192.168.56.102	192.168.56.101	SMTP	189	S: 250-server Hello [192.168.56.101] [192.168.56.101] | 250-SIZE 52428800
13	0.334836908	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=24 Ack=191 Win=29312 Len=0 TSval=8867507 TSecr=225573
14	0.367692508	192.168.56.101	192.168.56.102	SMTP	130	C: MAIL FROM:<nombre1_9721@sstt9721.org> BODY=8BITMIME SIZE=410
15	0.368052438	192.168.56.102	192.168.56.101	SMTP	76	S: 250 OK
16	0.397958832	192.168.56.101	192.168.56.102	SMTP	105	C: RCPT TO:<nombre2_9721@sstt9721.org>
17	0.398395144	192.168.56.102	192.168.56.101	SMTP	82	S: 250 Accepted
18	0.437449347	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=123 Ack=213 Win=29312 Len=0 TSval=8867533 TSecr=22557
19	0.514214691	192.168.56.101	192.168.56.102	SMTP	74	C: DATA
20	0.514684913	192.168.56.102	192.168.56.101	SMTP	124	S: 354 Enter message, ending with "." on a line by itself
21	0.514713719	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=129 Ack=269 Win=29312 Len=0 TSval=8867552 TSecr=22557
22	0.516001301	192.168.56.101	192.168.56.102	SMTP	478	C: DATA fragment, 410 bytes
23	0.517160470	192.168.56.101	192.168.56.102	SMTP/I…	71	from: nombre1 9721 <nombre1_9721@sstt9721.org>, subject: Mensaje 1, (text

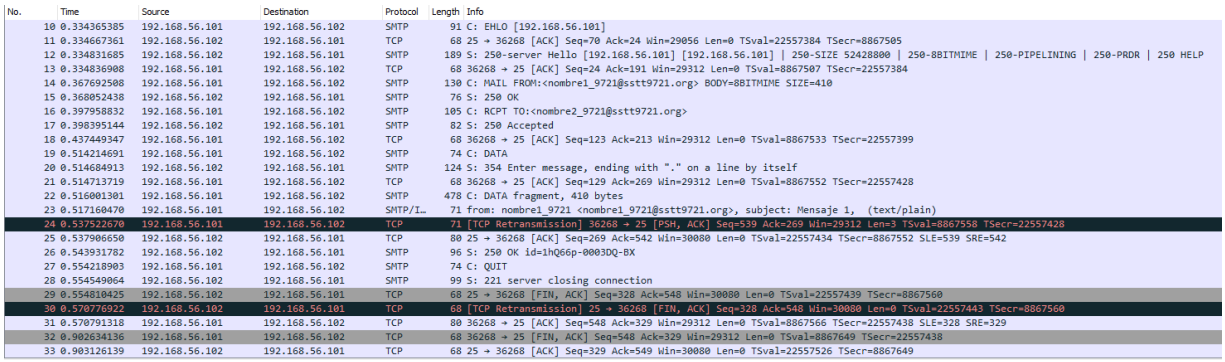
```
> Frame 8: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits) on interface 0
> Linux cooked capture
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> Transmission Control Protocol, Src Port: 25, Dst Port: 36268, Seq: 1, Ack: 1, Len: 69
v Simple Mail Transfer Protocol
  v Response: 220 server ESMTP Exim 4.86_2 Ubuntu Mon, 13 May 2019 10:14:54 +0200\r\n
      Response code: <domain> Service ready (220)
      Response parameter: server ESMTP Exim 4.86_2 Ubuntu Mon, 13 May 2019 10:14:54 +0200
```



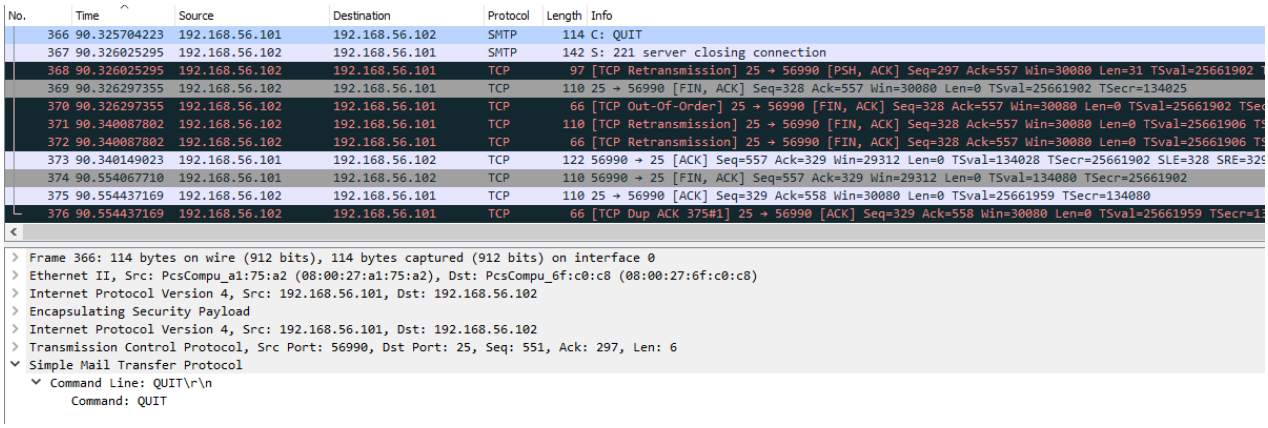
No.	Time	Source	Destination	Protocol	Length	Info
340	90.145162215	192.168.56.102	192.168.56.101	SMTP	178	S: 220 server ESMTP Exim 4.86_2 Ubuntu Fri, 17 May 2019 18:39:50 +0200
341	90.145162215	192.168.56.102	192.168.56.101	TCP	135	[TCP Retransmission] 25 → 56990 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=69 TSval=
342	90.145231941	192.168.56.101	192.168.56.102	TCP	110	56990 → 25 [ACK] Seq=1 Ack=70 Win=29312 Len=0 TSval=133979 TSecr=25661857
343	90.213338259	192.168.56.101	192.168.56.102	SMTP	134	C: EHLO [192.168.56.101]
344	90.213755080	192.168.56.102	192.168.56.101	TCP	110	25 → 56990 [ACK] Seq=70 Ack=24 Win=29056 Len=0 TSval=25661874 TSecr=133996
345	90.213755080	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 344#1] 25 → 56990 [ACK] Seq=70 Ack=24 Win=29056 Len=0 TSval=25661
346	90.213908794	192.168.56.102	192.168.56.101	SMTP	230	S: 250-server Hello [192.168.56.101] [192.168.56.101] | 250-SIZE 52428800 | 25
347	90.213908794	192.168.56.102	192.168.56.101	TCP	187	[TCP Retransmission] 25 → 56990 [PSH, ACK] Seq=70 Ack=24 Win=29056 Len=121 TSv
348	90.213940990	192.168.56.101	192.168.56.102	TCP	110	56990 → 25 [ACK] Seq=24 Ack=191 Win=29312 Len=0 TSval=133997 TSecr=25661874
349	90.221609756	192.168.56.101	192.168.56.102	SMTP	170	C: MAIL FROM:<nombre1_9721@sstt9721.org> BODY=8BITMIME SIZE=419
350	90.222008836	192.168.56.102	192.168.56.101	SMTP	118	S: 250 OK
351	90.222008836	192.168.56.102	192.168.56.101	TCP	74	[TCP Retransmission] 25 → 56990 [PSH, ACK] Seq=191 Ack=86 Win=29056 Len=8 TSva
352	90.222147087	192.168.56.101	192.168.56.102	SMTP	146	C: RCPT TO:<nombre2_9721@sstt9721.org>
353	90.222503951	192.168.56.102	192.168.56.101	SMTP	122	S: 250 Accepted
354	90.222503951	192.168.56.102	192.168.56.101	TCP	80	[TCP Retransmission] 25 → 56990 [PSH, ACK] Seq=199 Ack=123 Win=29056 Len=14 TS
355	90.222622268	192.168.56.101	192.168.56.102	SMTP	114	C: DATA
356	90.222903344	192.168.56.102	192.168.56.101	SMTP	166	S: 354 Enter message, ending with "." on a line by itself
357	90.222903344	192.168.56.102	192.168.56.101	TCP	122	[TCP Retransmission] 25 → 56990 [PSH, ACK] Seq=213 Ack=129 Win=29056 Len=56 TS
358	90.265707383	192.168.56.101	192.168.56.102	TCP	110	56990 → 25 [ACK] Seq=129 Ack=269 Win=29312 Len=0 TSval=134010 TSecr=25661876
359	90.301419782	192.168.56.101	192.168.56.102	SMTP	530	C: DATA fragment, 419 bytes
360	90.301528604	192.168.56.101	192.168.56.102	SMTP/I…	114	from: nombre1 9721 <nombre1 9721@sstt9721.org>, subject: Mensaje de prueba, (

```
> Frame 340: 178 bytes on wire (1424 bits), 178 bytes captured (1424 bits) on interface 0
> Ethernet II, Src: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8), Dst: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2)
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.101
> Transmission Control Protocol, Src Port: 25, Dst Port: 56990, Seq: 1, Ack: 1, Len: 69
v Simple Mail Transfer Protocol
  v Response: 220 server ESMTP Exim 4.86_2 Ubuntu Fri, 17 May 2019 18:39:50 +0200\r\n
      Response code: <domain> Service ready (220)
      Response parameter: server ESMTP Exim 4.86_2 Ubuntu Fri, 17 May 2019 18:39:50 +0200
```

A partir de ahora, vamos a relatar la sucesión de mensajes intercambiada entre cliente y servidor, en la que podremos apreciar lo que va ocurriendo. Primero, el cliente (Thunderbird) abre una sesión con el servidor utilizando el mensaje *EHLO*, y el servidor la acepta, con el mensaje *250-server Hello*. El cliente comienza a enviar información al servidor sobre un correo que quiere enviar. Le envía el remitente con un mensaje *MAIL FROM*, a continuación el destino, con *RCPT TO*, y después acaba enviando el correo completo. Finalmente, el cliente acaba cerrando la conexión con un mensaje *QUIT*. Aquí podemos ver la sucesión de mensajes:



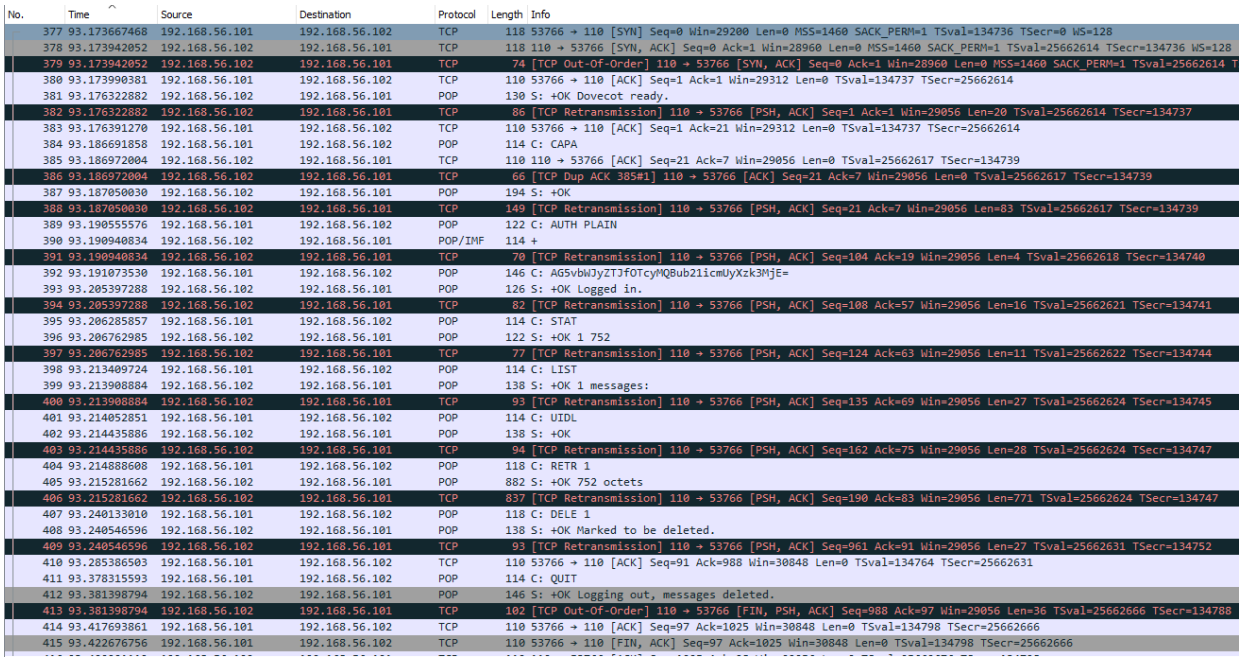
No.	Time	Source	Destination	Protocol	Length	Info
10	0.334365385	192.168.56.101	192.168.56.102	SMTP	91	C: EHLO [192.168.56.101]
11	0.334667361	192.168.56.102	192.168.56.101	TCP	68	25 → 36268 [ACK] Seq=70 Ack=24 Win=29056 Len=0 TSval=22557384 TSecr=8867505
12	0.334831685	192.168.56.102	192.168.56.101	SMTP	189	S: 250-server Hello [192.168.56.101] [192.168.56.101] | 250-SIZE 52428800 | 250-8BITMIME | 250-PIPELINING | 250-PRDR | 250 HELP
13	0.334836908	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=24 Ack=191 Win=29312 Len=0 TSval=8867507 TSecr=22557384
14	0.367692508	192.168.56.101	192.168.56.102	SMTP	130	C: MAIL FROM:<nombre1_9721@sstt9721.org> BODY=8BITMIME SIZE=410
15	0.368052438	192.168.56.102	192.168.56.101	SMTP	76	S: 250 OK
16	0.397958832	192.168.56.101	192.168.56.102	SMTP	105	C: RCPT TO:<nombre2_9721@sstt9721.org>
17	0.398395144	192.168.56.102	192.168.56.101	SMTP	82	S: 250 Accepted
18	0.437449347	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=123 Ack=213 Win=29312 Len=0 TSval=8867533 TSecr=22557399
19	0.514214691	192.168.56.101	192.168.56.102	SMTP	74	C: DATA
20	0.514684913	192.168.56.102	192.168.56.101	SMTP	124	S: 354 Enter message, ending with "." on a line by itself
21	0.514713719	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [ACK] Seq=129 Ack=269 Win=29312 Len=0 TSval=8867552 TSecr=22557428
22	0.516001301	192.168.56.101	192.168.56.102	SMTP	478	C: DATA fragment, 410 bytes
23	0.517160470	192.168.56.101	192.168.56.102	SMTP/I…	71	from: nombre1_9721 <nombre1_9721@sstt9721.org>, subject: Mensaje 1, (text/plain)
24	0.537522670	192.168.56.101	192.168.56.102	TCP	71	[TCP Retransmission] 36268 → 25 [PSH, ACK] Seq=539 Ack=269 Win=29312 Len=3 TSval=8867558 TSecr=22557428
25	0.537906650	192.168.56.102	192.168.56.101	TCP	80	25 → 36268 [ACK] Seq=269 Ack=542 Win=30080 Len=0 TSval=22557434 TSecr=8867552 SLE=539 SRE=542
26	0.543931782	192.168.56.102	192.168.56.101	SMTP	96	S: 250 OK id=1hQ66p-0003DQ-BX
27	0.554218903	192.168.56.101	192.168.56.102	SMTP	74	C: QUIT
28	0.554549064	192.168.56.102	192.168.56.101	SMTP	99	S: 221 server closing connection
29	0.554810425	192.168.56.102	192.168.56.101	TCP	68	25 → 36268 [FIN, ACK] Seq=328 Ack=548 Win=30080 Len=0 TSval=22557439 TSecr=8867560
30	0.570776922	192.168.56.102	192.168.56.101	TCP	68	[TCP Retransmission] 25 → 36268 [FIN, ACK] Seq=328 Ack=548 Win=30080 Len=0 TSval=22557443 TSecr=8867560
31	0.570791318	192.168.56.101	192.168.56.102	TCP	80	36268 → 25 [ACK] Seq=548 Ack=329 Win=29312 Len=0 TSval=8867566 TSecr=22557438 SLE=328 SRE=329
32	0.902634136	192.168.56.101	192.168.56.102	TCP	68	36268 → 25 [FIN, ACK] Seq=548 Ack=329 Win=29312 Len=0 TSval=8867649 TSecr=22557438
33	0.903126139	192.168.56.102	192.168.56.101	TCP	68	25 → 36268 [ACK] Seq=329 Ack=549 Win=30080 Len=0 TSval=22557526 TSecr=8867649



No.	Time	Source	Destination	Protocol	Length	Info
366	90.325704223	192.168.56.101	192.168.56.102	SMTP	114	C: QUIT
367	90.326025295	192.168.56.102	192.168.56.101	SMTP	142	S: 221 server closing connection
368	90.326025295	192.168.56.102	192.168.56.101	TCP	97	[TCP Retransmission] 25 → 56990 [PSH, ACK] Seq=297 Ack=557 Win=30080 Len=31 TSval=25661902 T
369	90.326297355	192.168.56.102	192.168.56.101	TCP	110	25 → 56990 [FIN, ACK] Seq=328 Ack=557 Win=30080 Len=0 TSval=25661902 TSecr=134025
370	90.326297355	192.168.56.102	192.168.56.101	TCP	66	[TCP Out-Of-Order] 25 → 56990 [FIN, ACK] Seq=328 Ack=557 Win=30080 Len=0 TSval=25661902 TSe
371	90.340087802	192.168.56.102	192.168.56.101	TCP	110	[TCP Retransmission] 25 → 56990 [FIN, ACK] Seq=328 Ack=557 Win=30080 Len=0 TSval=25661906 TS
372	90.340087802	192.168.56.102	192.168.56.101	TCP	66	[TCP Retransmission] 25 → 56990 [FIN, ACK] Seq=328 Ack=557 Win=30080 Len=0 TSval=25661906 TS
373	90.340149023	192.168.56.101	192.168.56.102	TCP	122	56990 → 25 [ACK] Seq=557 Ack=329 Win=29312 Len=0 TSval=134028 TSecr=25661902 SLE=328 SRE=329
374	90.554067710	192.168.56.101	192.168.56.102	TCP	110	56990 → 25 [FIN, ACK] Seq=557 Ack=329 Win=29312 Len=0 TSval=134080 TSecr=25661902
375	90.554437169	192.168.56.102	192.168.56.101	TCP	110	25 → 56990 [ACK] Seq=329 Ack=558 Win=30080 Len=0 TSval=25661959 TSecr=134080
376	90.554437169	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 375#1] 25 → 56990 [ACK] Seq=329 Ack=558 Win=30080 Len=0 TSval=25661959 TSecr=13

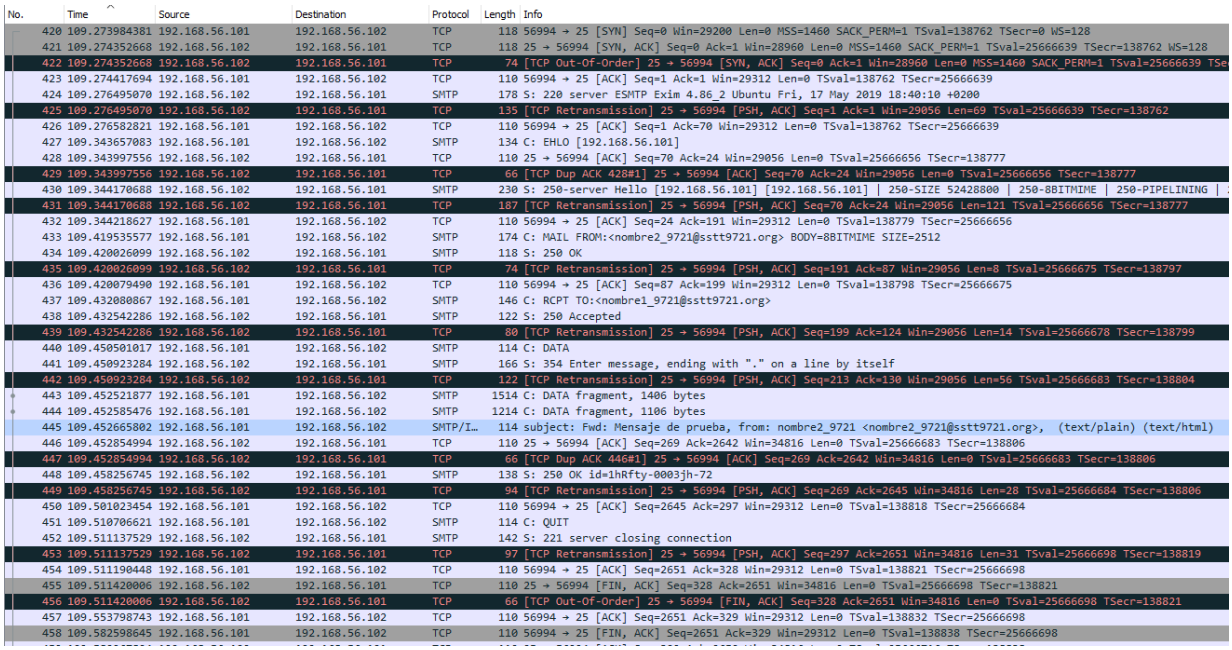
```
> Frame 366: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
> Ethernet II, Src: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2), Dst: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8)
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Encapsulating Security Payload
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> Transmission Control Protocol, Src Port: 56990, Dst Port: 25, Seq: 551, Ack: 297, Len: 6
v Simple Mail Transfer Protocol
  v Command Line: QUIT\r\n
      Command: QUIT
```

Ahora el usuario nombre2_9721@sstt9721.org quiere recuperar los correos que haya en el servidor. En este caso solo los nuevos, porque tenemos activada la opción de borrar los mensajes después de ser leídos. Para hacerlo, ya no debe hacer una consulta DNS para saber dónde se encuentra el servidor POP3, ya que esa consulta ha sido realizada anteriormente y se encuentra almacenada en caché. El cliente establece entonces una conexión con el servidor, y el usuario se autentica. Éste solicita un listado de los correos sin leer con la orden *LIST*, y el servidor le responde que tiene un mensaje nuevo. El usuario solicita dicho mensaje (*RETR 1*), y el servidor se lo envía. Como tenemos activada la opción de borrado de correos, Thunderbird solicita al servidor el borrado del mensaje 1 con *DELE 1*, y el servidor le responde cuando lo ha eliminado. Finalmente cierra la conexión con la orden *QUIT*.



No.	Time	Source	Destination	Protocol	Length	Info
377	93.173667468	192.168.56.101	192.168.56.102	TCP	118	53766 → 110 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=134736 TSecr=0 WS=128
378	93.173942052	192.168.56.102	192.168.56.101	TCP	118	110 → 53766 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25662614 TSecr=134736 WS=128
379	93.173942052	192.168.56.102	192.168.56.101	TCP	74	[TCP Out-Of-Order] 110 → 53766 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25662614 T
380	93.173990381	192.168.56.101	192.168.56.102	TCP	110	53766 → 110 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=134737 TSecr=25662614
381	93.176322882	192.168.56.102	192.168.56.101	POP	130	S: +OK Dovecot ready.
382	93.176322882	192.168.56.102	192.168.56.101	TCP	86	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=20 TSval=25662614 TSecr=134737
383	93.176391270	192.168.56.101	192.168.56.102	TCP	110	53766 → 110 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=134737 TSecr=25662614
384	93.186691858	192.168.56.101	192.168.56.102	POP	114	C: CAPA
385	93.186972004	192.168.56.102	192.168.56.101	TCP	110	110 → 53766 [ACK] Seq=21 Ack=7 Win=29056 Len=0 TSval=25662617 TSecr=134739
386	93.186972004	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 385#1] 110 → 53766 [ACK] Seq=21 Ack=7 Win=29056 Len=0 TSval=25662617 TSecr=134739
387	93.187050030	192.168.56.102	192.168.56.101	POP	194	S: +OK
388	93.187050030	192.168.56.102	192.168.56.101	TCP	149	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=21 Ack=7 Win=29056 Len=83 TSval=25662617 TSecr=134739
389	93.190555576	192.168.56.101	192.168.56.102	POP	122	C: AUTH PLAIN
390	93.190940834	192.168.56.102	192.168.56.101	POP/IMF	114	+
391	93.190940834	192.168.56.102	192.168.56.101	TCP	70	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=104 Ack=19 Win=29056 Len=4 TSval=25662618 TSecr=134740
392	93.191073530	192.168.56.101	192.168.56.102	POP	146	C: AG5vbWJyZTJfOTcyMQBub21icmUyXzk3MjE=
393	93.205397288	192.168.56.102	192.168.56.101	POP	126	S: +OK Logged in.
394	93.205397288	192.168.56.102	192.168.56.101	TCP	82	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=108 Ack=57 Win=29056 Len=16 TSval=25662621 TSecr=134741
395	93.206285857	192.168.56.101	192.168.56.102	POP	114	C: STAT
396	93.206762985	192.168.56.102	192.168.56.101	POP	122	S: +OK 1 752
397	93.206762985	192.168.56.102	192.168.56.101	TCP	77	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=124 Ack=63 Win=29056 Len=11 TSval=25662622 TSecr=134744
398	93.213409724	192.168.56.101	192.168.56.102	POP	114	C: LIST
399	93.213908884	192.168.56.102	192.168.56.101	POP	138	S: +OK 1 messages:
400	93.213908884	192.168.56.102	192.168.56.101	TCP	93	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=135 Ack=69 Win=29056 Len=27 TSval=25662624 TSecr=134745
401	93.214052851	192.168.56.101	192.168.56.102	POP	114	C: UIDL
402	93.214435886	192.168.56.102	192.168.56.101	POP	138	S: +OK
403	93.214435886	192.168.56.102	192.168.56.101	TCP	94	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=162 Ack=75 Win=29056 Len=28 TSval=25662624 TSecr=134747
404	93.214888608	192.168.56.101	192.168.56.102	POP	118	C: RETR 1
405	93.215281662	192.168.56.102	192.168.56.101	POP	882	S: +OK 752 octets
406	93.215281662	192.168.56.102	192.168.56.101	TCP	837	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=190 Ack=83 Win=29056 Len=771 TSval=25662624 TSecr=134747
407	93.240133010	192.168.56.101	192.168.56.102	POP	118	C: DELE 1
408	93.240546596	192.168.56.102	192.168.56.101	POP	138	S: +OK Marked to be deleted.
409	93.240546596	192.168.56.102	192.168.56.101	TCP	93	[TCP Retransmission] 110 → 53766 [PSH, ACK] Seq=961 Ack=91 Win=29056 Len=27 TSval=25662631 TSecr=134752
410	93.285386503	192.168.56.101	192.168.56.102	TCP	110	53766 → 110 [ACK] Seq=91 Ack=988 Win=30848 Len=0 TSval=134764 TSecr=25662631
411	93.378315593	192.168.56.101	192.168.56.102	POP	114	C: QUIT
412	93.381398794	192.168.56.102	192.168.56.101	POP	146	S: +OK Logging out, messages deleted.
413	93.381398794	192.168.56.102	192.168.56.101	TCP	102	[TCP Out-Of-Order] 110 → 53766 [FIN, PSH, ACK] Seq=988 Ack=97 Win=29056 Len=36 TSval=25662666 TSecr=134788
414	93.417693861	192.168.56.101	192.168.56.102	TCP	110	53766 → 110 [ACK] Seq=97 Ack=1025 Win=30848 Len=0 TSval=134798 TSecr=25662666
415	93.422676756	192.168.56.101	192.168.56.102	TCP	110	53766 → 110 [FIN, ACK] Seq=97 Ack=1025 Win=30848 Len=0 TSval=134798 TSecr=25662666

Si seguimos avanzando en la traza, veremos que el último mensaje que envía el usuario nombre1_9721@sstt9721.org es un reenvío del mensaje que el usuario nombre2_9721@sstt9721.org le ha enviado:

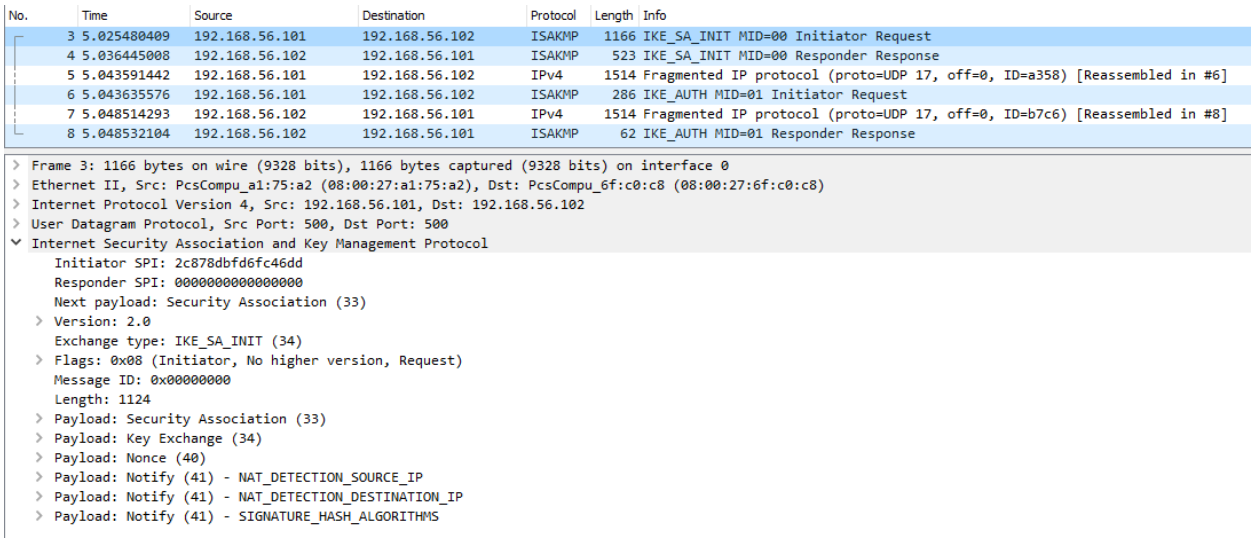


No.	Time	Source	Destination	Protocol	Length	Info
420	109.273984381	192.168.56.101	192.168.56.102	TCP	118	56994 → 25 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=138762 TSecr=0 WS=128
421	109.274352668	192.168.56.102	192.168.56.101	TCP	118	25 → 56994 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25666639 TSecr=138762 WS=128
422	109.274352668	192.168.56.102	192.168.56.101	TCP	74	[TCP Out-Of-Order] 25 → 56994 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=25666639 TSe
423	109.274417694	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=138762 TSecr=25666639
424	109.276495070	192.168.56.102	192.168.56.101	SMTP	178	S: 220 server ESMTP Exim 4.86_2 Ubuntu Fri, 17 May 2019 18:40:10 +0200
425	109.276495070	192.168.56.102	192.168.56.101	TCP	135	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=69 TSval=25666639 TSecr=138762
426	109.276582821	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=1 Ack=70 Win=29312 Len=0 TSval=138762 TSecr=25666639
427	109.343657083	192.168.56.101	192.168.56.102	SMTP	134	C: EHLO [192.168.56.101]
428	109.343997556	192.168.56.102	192.168.56.101	TCP	110	25 → 56994 [ACK] Seq=70 Ack=24 Win=29056 Len=0 TSval=25666656 TSecr=138777
429	109.343997556	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 428#1] 25 → 56994 [ACK] Seq=70 Ack=24 Win=29056 Len=0 TSval=25666656 TSecr=138777
430	109.344170688	192.168.56.102	192.168.56.101	SMTP	230	S: 250-server Hello [192.168.56.101] [192.168.56.101] | 250-SIZE 52428800 | 250-8BITMIME | 250-PIPELINING |
431	109.344170688	192.168.56.102	192.168.56.101	TCP	187	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=70 Ack=24 Win=29056 Len=121 TSval=25666656 TSecr=138777
432	109.344218627	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=24 Ack=191 Win=29312 Len=0 TSval=138779 TSecr=25666656
433	109.419535577	192.168.56.101	192.168.56.102	SMTP	174	C: MAIL FROM:<nombre2_9721@sstt9721.org> BODY=8BITMIME SIZE=2512
434	109.420026099	192.168.56.102	192.168.56.101	SMTP	118	S: 250 OK
435	109.420026099	192.168.56.102	192.168.56.101	TCP	74	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=191 Ack=87 Win=29056 Len=8 TSval=25666675 TSecr=138797
436	109.420079490	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=87 Ack=199 Win=29312 Len=0 TSval=138798 TSecr=25666675
437	109.432080867	192.168.56.101	192.168.56.102	SMTP	146	C: RCPT TO:<nombre1_9721@sstt9721.org>
438	109.432542286	192.168.56.102	192.168.56.101	SMTP	122	S: 250 Accepted
439	109.432542286	192.168.56.102	192.168.56.101	TCP	80	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=199 Ack=124 Win=29056 Len=14 TSval=25666678 TSecr=138799
440	109.450501017	192.168.56.101	192.168.56.102	SMTP	114	C: DATA
441	109.450923284	192.168.56.102	192.168.56.101	SMTP	166	S: 354 Enter message, ending with "." on a line by itself
442	109.450923284	192.168.56.102	192.168.56.101	TCP	122	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=213 Ack=130 Win=29056 Len=56 TSval=25666683 TSecr=138804
443	109.452521877	192.168.56.101	192.168.56.102	SMTP	1514	C: DATA fragment, 1406 bytes
444	109.452585476	192.168.56.101	192.168.56.102	SMTP	1214	C: DATA fragment, 1106 bytes
445	109.452665802	192.168.56.101	192.168.56.102	SMTP/I…	114	subject: Fwd: Mensaje de prueba, from: nombre2_9721 <nombre2_9721@sstt9721.org>, (text/plain) (text/html)
446	109.452854994	192.168.56.102	192.168.56.101	TCP	110	25 → 56994 [ACK] Seq=269 Ack=2642 Win=34816 Len=0 TSval=25666683 TSecr=138806
447	109.452854994	192.168.56.102	192.168.56.101	TCP	66	[TCP Dup ACK 446#1] 25 → 56994 [ACK] Seq=269 Ack=2642 Win=34816 Len=0 TSval=25666683 TSecr=138806
448	109.458256745	192.168.56.102	192.168.56.101	SMTP	138	S: 250 OK id=1hRfty-0003jh-72
449	109.458256745	192.168.56.102	192.168.56.101	TCP	94	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=269 Ack=2645 Win=34816 Len=28 TSval=25666684 TSecr=138806
450	109.501023454	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=2645 Ack=297 Win=29312 Len=0 TSval=138818 TSecr=25666684
451	109.510706621	192.168.56.101	192.168.56.102	SMTP	114	C: QUIT
452	109.511137529	192.168.56.102	192.168.56.101	SMTP	142	S: 221 server closing connection
453	109.511137529	192.168.56.102	192.168.56.101	TCP	97	[TCP Retransmission] 25 → 56994 [PSH, ACK] Seq=297 Ack=2651 Win=34816 Len=31 TSval=25666698 TSecr=138819
454	109.511190448	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=2651 Ack=328 Win=29312 Len=0 TSval=138821 TSecr=25666698
455	109.511420006	192.168.56.102	192.168.56.101	TCP	110	25 → 56994 [FIN, ACK] Seq=328 Ack=2651 Win=34816 Len=0 TSval=25666698 TSecr=138821
456	109.511420006	192.168.56.102	192.168.56.101	TCP	66	[TCP Out-Of-Order] 25 → 56994 [FIN, ACK] Seq=328 Ack=2651 Win=34816 Len=0 TSval=25666698 TSecr=138821
457	109.553798743	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [ACK] Seq=2651 Ack=329 Win=29312 Len=0 TSval=138832 TSecr=25666698
458	109.582598645	192.168.56.101	192.168.56.102	TCP	110	56994 → 25 [FIN, ACK] Seq=2651 Ack=329 Win=29312 Len=0 TSval=138838 TSecr=25666698

Uso de IKE e IPsec.

En primer lugar, ocurre la comunicación que pone en marcha el servicio IPsec entre cliente y servidor. Se ejecuta IKE para intercambiar claves y certificados entre ambas partes, y si todo se produce correctamente, se creará un estado en el SAD de cada host.

En esta primera captura podemos ver los mensajes IKE que se intercambian entre cliente y servidor:



Hay dos fases diferenciadas:

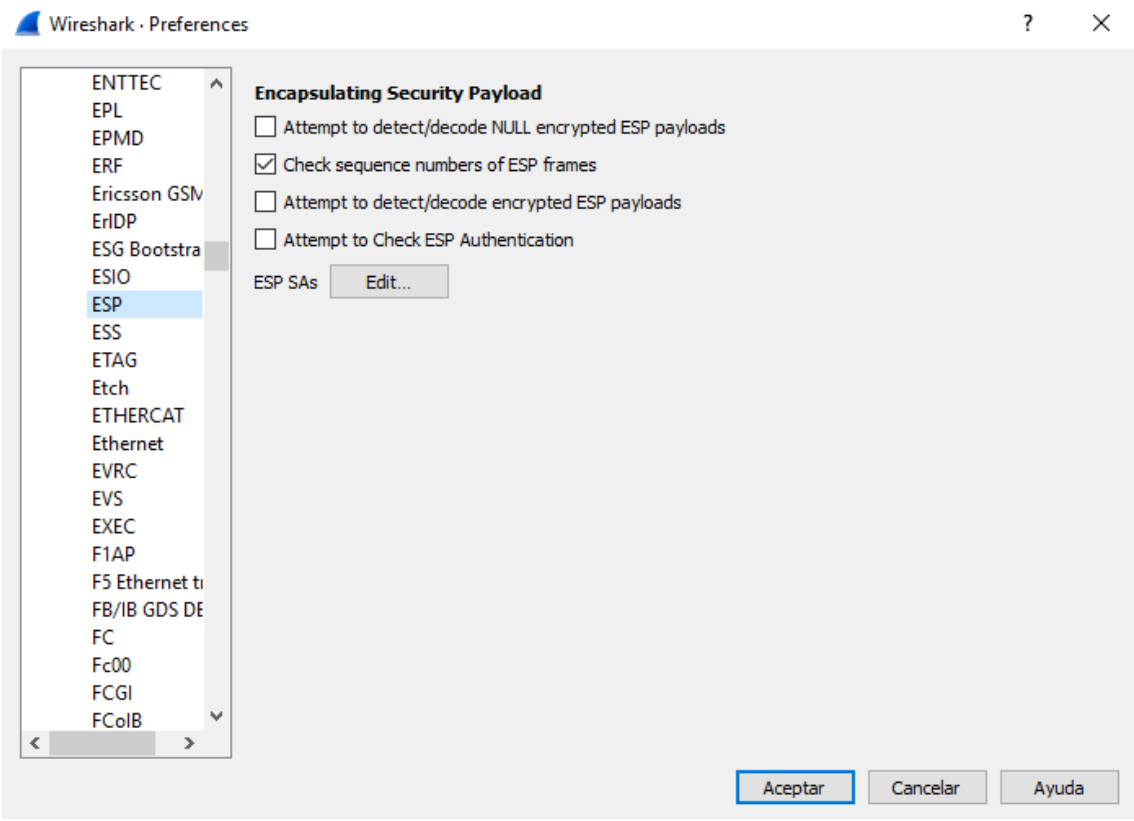
- La primera fase tiene lugar con los mensajes IKE_SA_INIT. Aquí se produce el intercambio Diffie-Hellman, el intercambio de números aleatorios, y también el intercambio de suites criptográficas. El "Initiator" es el que primero comienza el intercambio: en este caso, el cliente.
- La segunda fase ocurre con los mensajes IKE_AUTH. Aquí ya se ha producido correctamente el intercambio anterior, y se intercambian los certificados, las políticas, las identidades, y se vuelve a compartir la suite criptográfica. Si hay algún error en la autenticación todo el intercambio se detendrá en esta parte.

Una vez que se autentican las partes y se crea en estado en la SAD, se pueden comenzar a enviar mensajes protegidos. Como vemos en esta captura, cliente y servidor intercambian varios mensajes entre sí protegidos con ESP, pero no podemos ver el contenido de ellos a pesar de que no existe cifrado:

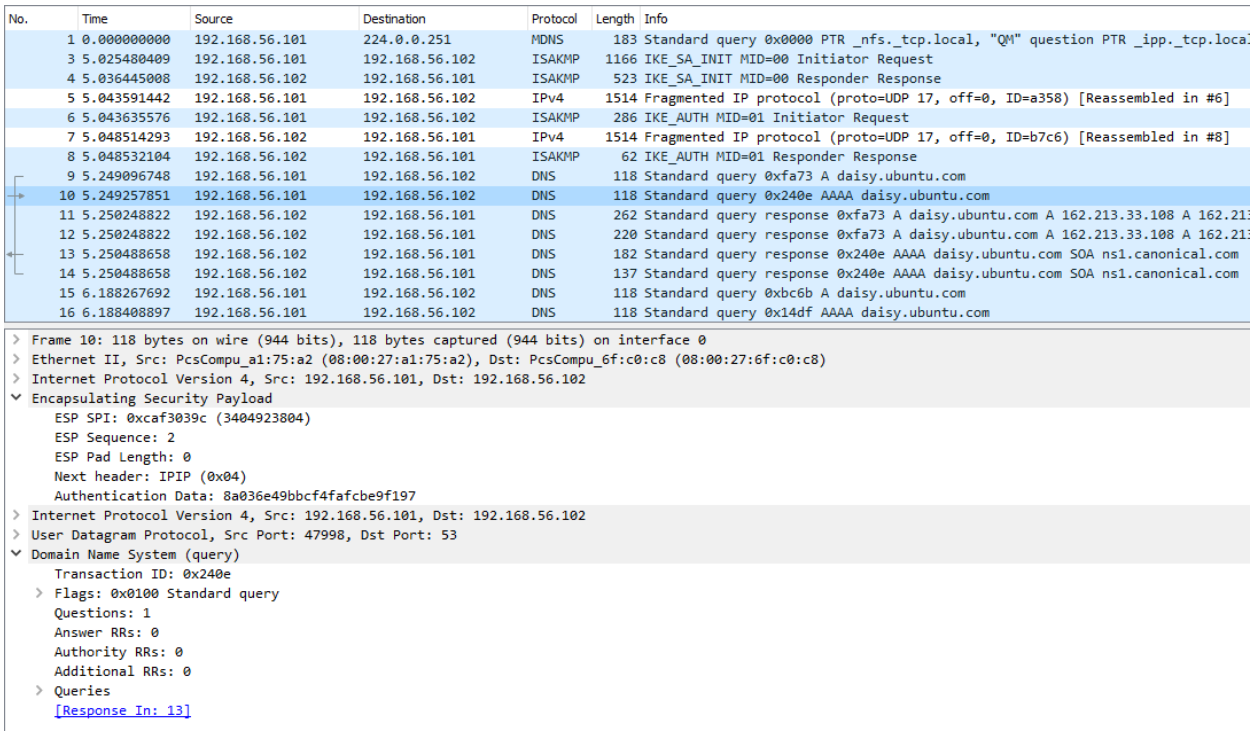
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.101	224.0.0.251	MDNS	183	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local
3	5.025480409	192.168.56.101	192.168.56.102	ISAKMP	1166	IKE_SA_INIT MID=00 Initiator Request
4	5.036445008	192.168.56.102	192.168.56.101	ISAKMP	523	IKE_SA_INIT MID=00 Responder Response
5	5.043591442	192.168.56.101	192.168.56.102	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=a358) [Reassembled in #6]
6	5.043635576	192.168.56.101	192.168.56.102	ISAKMP	286	IKE_AUTH MID=01 Initiator Request
7	5.048514293	192.168.56.102	192.168.56.101	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=b7c6) [Reassembled in #8]
8	5.048532104	192.168.56.102	192.168.56.101	ISAKMP	62	IKE_AUTH MID=01 Responder Response
9	5.249096748	192.168.56.101	192.168.56.102	ESP	118	ESP (SPI=0xcaf3039c)
10	5.249257851	192.168.56.101	192.168.56.102	ESP	118	ESP (SPI=0xcaf3039c)
11	5.250248822	192.168.56.102	192.168.56.101	ESP	262	ESP (SPI=0xc43b4f9b)
12	5.250248822	192.168.56.102	192.168.56.101	DNS	220	Standard query response 0xfa73 A daisy.ubuntu.com A 162.213.33.108 A 162.213
13	5.250488658	192.168.56.102	192.168.56.101	ESP	182	ESP (SPI=0xc43b4f9b)
14	5.250488658	192.168.56.102	192.168.56.101	DNS	137	Standard query response 0x240e AAAA daisy.ubuntu.com SOA ns1.canonical.com
15	6.188267692	192.168.56.101	192.168.56.102	ESP	118	ESP (SPI=0xcaf3039c)
16	6.188408897	192.168.56.101	192.168.56.102	ESP	118	ESP (SPI=0xcaf3039c)

```
> Frame 10: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
> Ethernet II, Src: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2), Dst: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8)
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
v Encapsulating Security Payload
    ESP SPI: 0xcaf3039c (3404923804)
    ESP Sequence: 2
```

Esto es porque hay que activar una opción en el protocolo ESP en Wireshark. Seguiremos los siguientes pasos: **Edit** -> **Preferences** -> Desplegamos la lista de **Protocols** -> Buscamos el protocolo **ESP**. Estas son las opciones de configuración que permite Wireshark sobre el protocolo ESP:



La primera opción es la que tenemos que marcar, ya que hará que Wireshark intente descodificar aquellos mensajes ESP que no estén encriptados. La marcamos y pulsamos aceptar. Ahora tenemos el siguiente resultado:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.101	224.0.0.251	MDNS	183	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local
3	5.025480409	192.168.56.101	192.168.56.102	ISAKMP	1166	IKE_SA_INIT MID=00 Initiator Request
4	5.036445008	192.168.56.102	192.168.56.101	ISAKMP	523	IKE_SA_INIT MID=00 Responder Response
5	5.043591442	192.168.56.101	192.168.56.102	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=a358) [Reassembled in #6]
6	5.043635576	192.168.56.101	192.168.56.102	ISAKMP	286	IKE_AUTH MID=01 Initiator Request
7	5.048514293	192.168.56.102	192.168.56.101	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=b7c6) [Reassembled in #8]
8	5.048532104	192.168.56.102	192.168.56.101	ISAKMP	62	IKE_AUTH MID=01 Responder Response
9	5.249096748	192.168.56.101	192.168.56.102	DNS	118	Standard query 0xfa73 A daisy.ubuntu.com
10	5.249257851	192.168.56.101	192.168.56.102	DNS	118	Standard query 0x240e AAAA daisy.ubuntu.com
11	5.250248822	192.168.56.102	192.168.56.101	DNS	262	Standard query response 0xfa73 A daisy.ubuntu.com A 162.213.33.108 A 162.21
12	5.250248822	192.168.56.102	192.168.56.101	DNS	220	Standard query response 0xfa73 A daisy.ubuntu.com A 162.213.33.108 A 162.21
13	5.250488658	192.168.56.102	192.168.56.101	DNS	182	Standard query response 0x240e AAAA daisy.ubuntu.com SOA ns1.canonical.com
14	5.250488658	192.168.56.102	192.168.56.101	DNS	137	Standard query response 0x240e AAAA daisy.ubuntu.com SOA ns1.canonical.com
15	6.188267692	192.168.56.101	192.168.56.102	DNS	118	Standard query 0xbc6b A daisy.ubuntu.com
16	6.188408897	192.168.56.101	192.168.56.102	DNS	118	Standard query 0x14df AAAA daisy.ubuntu.com

```
> Frame 10: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
> Ethernet II, Src: PcsCompu_a1:75:a2 (08:00:27:a1:75:a2), Dst: PcsCompu_6f:c0:c8 (08:00:27:6f:c0:c8)
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
v Encapsulating Security Payload
    ESP SPI: 0xcaf3039c (3404923804)
    ESP Sequence: 2
    ESP Pad Length: 0
    Next header: IPIP (0x04)
    Authentication Data: 8a036e49bbcf4fafcbe9f197
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.102
> User Datagram Protocol, Src Port: 47998, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x240e
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 13]
```

Ya podemos ver el contenido de los mensajes protegidos con ESP. Todos los mensajes mostrados en los apartados anteriores iban protegidos con IPsec, por eso veíamos algunos mensajes duplicados. Para poder verlos desde el principio, se marcó la opción explicada anteriormente, y al llegar a este apartado se desactivó, ya que por defecto viene desactivada y era mejor explicarla así.

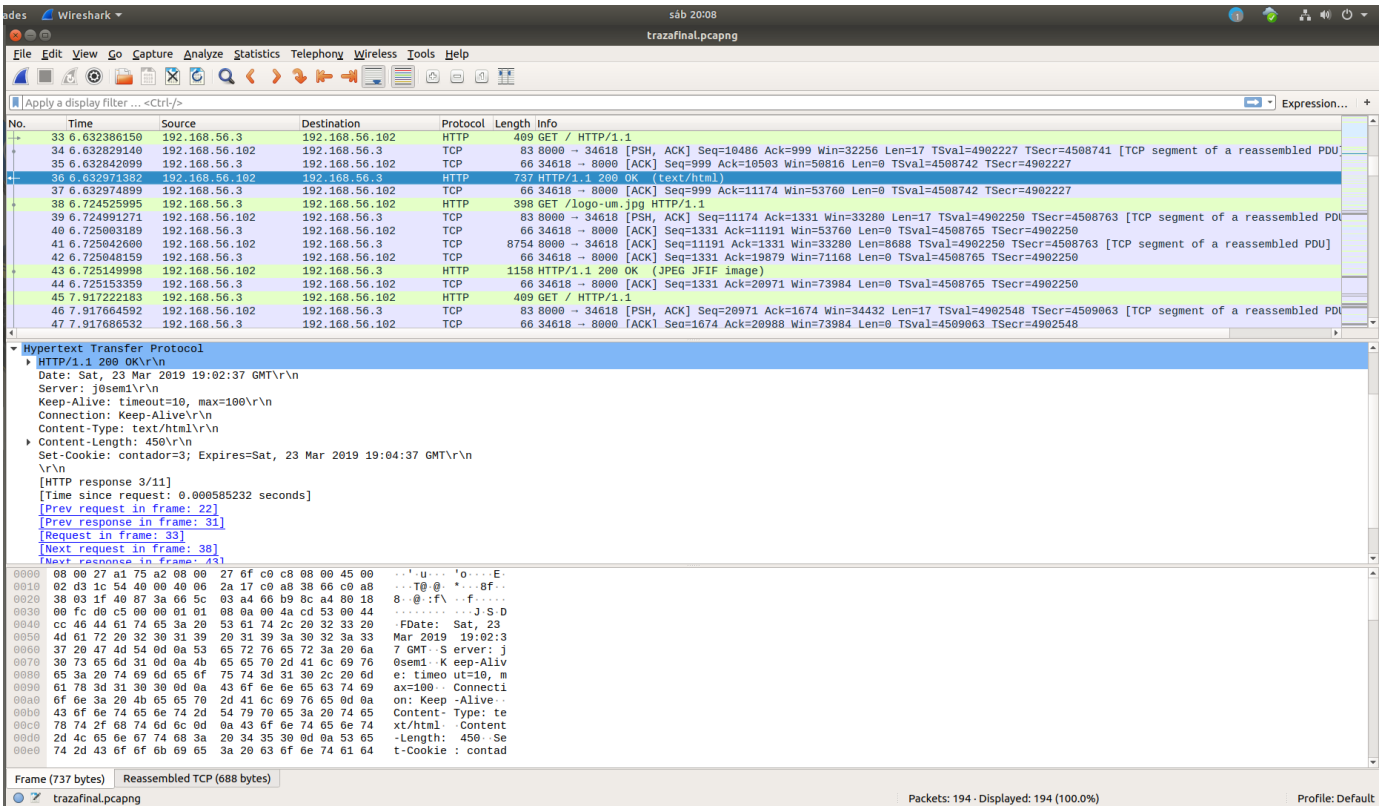
Realmente se ha usado el protocolo ESP, pero podría haberse usado el protocolo AH, ya que no se han cifrado los mensajes, solo se ha protegido la integridad de los mismos, y eso es lo que realiza AH, aunque ESP también puede realizarlo, como hemos visto.

Protocolo HTTP implementado

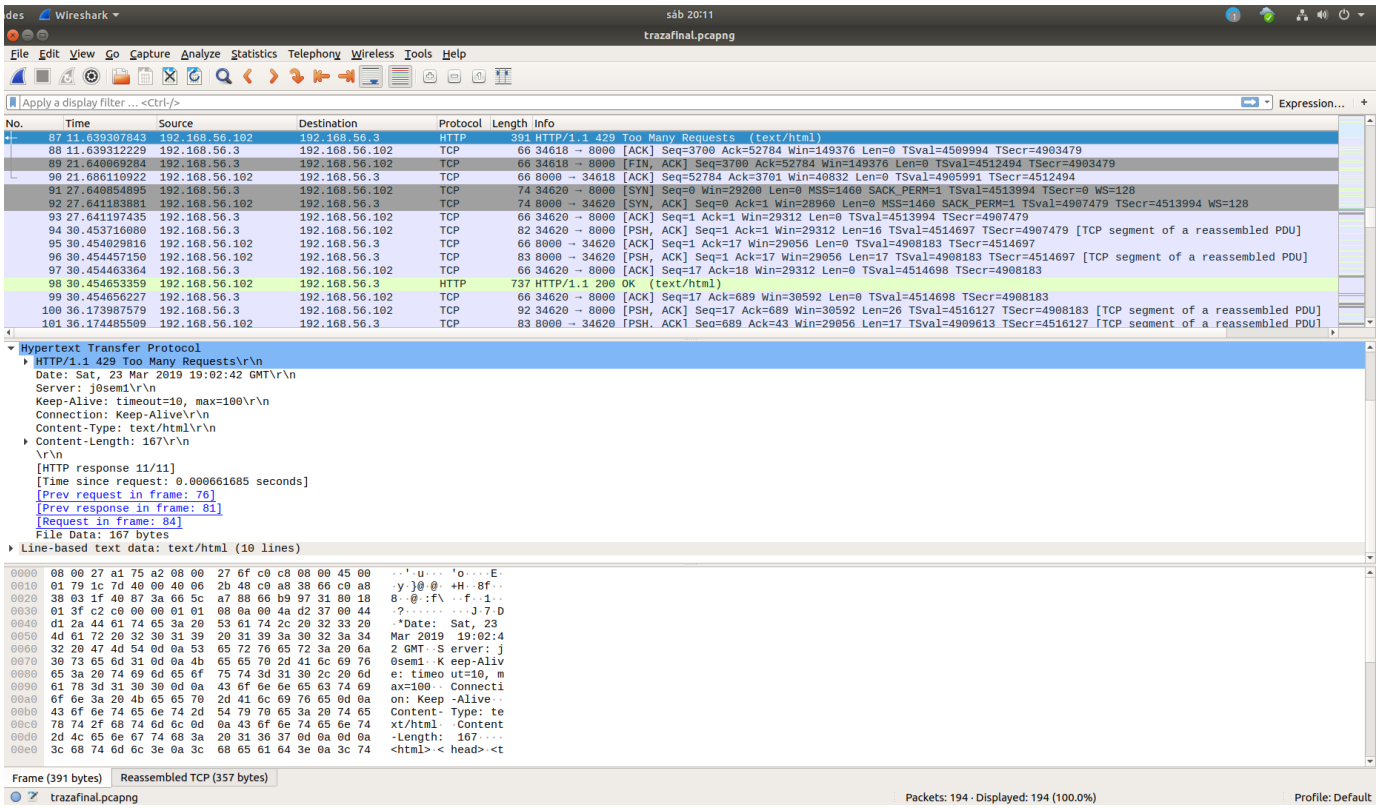
En la traza se encuentran las siguientes situaciones:

- La primera es una solicitud estándar desde un navegador (Firefox).
- A continuación, se siguen realizando solicitudes desde un navegador. Se puede ver en cada una como va aumentando el valor de la cookie, y finalmente la respuesta del servidor cuando le llega una cookie con valor 10.
- La siguiente es una solicitud GET básica realizada con telnet.
- Después encontramos solicitudes erróneas realizadas con telnet.
- Se puede apreciar la persistencia, observando cuándo se abre y cuándo se cierran las conexiones TCP.

Analicemos la segunda solicitud realizada desde el navegador:



Como se puede observar, el formato de la cabecera enviada por el servidor es correcta, y el contador de la cookie es 3, que ha aumentado su valor respecto a las anteriores cabeceras.



Con esta imagen vemos la respuesta del servidor cuando se la cookie ha alcanzado el valor 10. En los errores no se envía cookie ya que no es necesario. En esta captura también se pueden observar los mensajes de cierre de la conexión TCP al expirar la persistencia, y la respuesta enviada por el servidor a la primera petición GET realizada desde telnet.

Problemas encontrados en el proceso del desarrollo del escenario

Implementación del servicio Web-SSTT HTTP

El primer problema era provocado por la función *strtok()*, relacionado con los '\r' y '\n' que tienen las peticiones. Para solventarlo se utilizaron las expresiones regulares, y se sustituyeron por el carácter '*'. Sin embargo, este carácter trajo problemas más tarde, ya que cuando se pretendía comprobar si el formato de la cabecera recibida era correcto, *strtok()* encontraba otros '*' que no eran saltos de línea, por lo que se a partir de ese momento se sustituyeron por el carácter '~', que no supone ningún problema.

El formato de la cookie también ocasionaba malos funcionamientos por múltiples motivos: el formato de la fecha introducida con 'Expires' tenía que ser exactamente el que reconoce HTTP; cuando se envía información al cliente debe ser de esta manera <nombre>=<valor>, y eso provoca que el <valor> solo admita unos valores específicos. Eso eliminó la posibilidad de eliminar la fecha de expiración al cliente en el valor.

Otro problema fue la falta de restricción de Firefox, ya que Chrome es más restrictivo, y si algo es incorrecto lo sabes, pero con Firefox el formato de la cabecera puede no ser correcto y obviarse. Esto hizo que se arrastrara un problema hasta el final con el número de bytes que se enviaban al cliente.

Y el otro problema vino a la hora de poder detectar qué código de error se debía enviar al peticionario cuando algo no se produjera correctamente. Esto aumentó la complejidad del código para soportar más posibles escenarios que no se habían tenido en cuenta.

Número de horas aproximadas empleadas en cada apartado (2.1-2.5) y en la documentación

Apartado 2.1. Como es sabido por todos, esta se es la parte más extensa de la práctica. En mi caso, se han invertido 55 horas aproximadamente para poder realizarla. Es tan solo una aproximación, pueden haber sido más o menos.

Apartado 2.2. El servidor DNS era algo de una dificultad intermedia, y se realizó sin complicaciones, aproximadamente en 6 horas.

Apartado 2.3. Esta de las partes más sencillas de la práctica, sin embargo, debido a errores en la realización de la misma, se tardó unas 6 horas por tener que rehacerlo todo dos veces.

Apartado 2.4. Esta parte es más sencilla aún que la anterior, ya que con instalar el software Apache y configurar algunas cosas se tenía el escenario montado. En la parte de los certificados hay más cosas que entender que realizar. Se calculan en total unas 4 horas.

Apartado 2.5. Se calculan unas 7 horas, ya que hubo un problema en la autenticación con certificados y se perdió mucho tiempo en intentar solucionar dicho problema.

Documentación. 10 horas aproximadamente, teniendo en cuenta la parte de programación Web y el resto. También se le dedicó demasiado tiempo a tener que rehacer toda la parte de las trazas, porque en un primer momento se hicieron todas las trazas por separado, y luego hubo que hacer una que englobara a todas, por lo que se tuvieron que cambiar todas las imágenes de los mensajes y algunas explicaciones.

Conclusiones y valoración personal del trabajo realizado

Valoración personal de la práctica

Personalmente me parece una práctica muy interesante, ya que la carga de trabajo es adecuada, y el alumno aprende sobre la puesta en marcha de las infraestructuras que podemos encontrar en internet. Además, obliga al entendimiento de los conceptos teóricos expuestos en la asignatura. La parte de programación resulta más sencilla para aquellos alumnos que han superado la práctica de la asignatura 'Ampliación de los Sistemas Operativos'.

Valoración del trabajo realizado

En general he quedado contento con mi trabajo realizado en la práctica, especialmente en la parte de programación. Soy consciente de que la parte menos trabajada de la práctica se trata de la documentación que se entregó en la entrega de la parte de programación. Sin embargo, esta documentación es más completa y con una mejor estructura.

Conclusión

He disfrutado realizando la práctica, me parece que los servicios que se ponen en marcha y las reflexiones que pueden obtenerse de funcionamiento y seguridad ayudan a entender un poco mejor cómo funciona la infraestructura de Internet. Además, me han ayudado a entender el contenido teórico de la asignatura, o a pensar en cosas que solo con la parte teórica no pueden llegar a verse. La parte de seguridad de la asignatura es más extensa de lo que pensaba y supone una buena antesala a la intensificación de 4º.

Aún desconozco la intensificación que realizaré, y hace unos meses no tenía planeado realizar la de Tecnología de la Información, sin embargo, ahora parece ser la que más prefiero realizar. Eso habla muy bien de la asignatura y la práctica.