

Servicios Telemáticos

Documentación Práctica Final



Eduardo Blazque Verdejo
eduardo.blazquezv@um.es / 48846313V
Alfredo Marquina Meseguer
alfredo.m.m@um.es / 49445345Z

Profesor: Esteban Belmonte Rodríguez
Grupo/Subgrupo: 3/1

May 12, 2024

Contents

1	Introducción	4
1.1	Propósito	4
2	Web-SSTT HTTP Server	5
2.1	Descripción del servicio	5
2.2	Funcionalidad básica	5
2.3	Gestión de errores	6
2.4	Apache HTTP	6
2.5	Configuración de Apache HTTP	7
2.6	Análisis	9
2.6.1	Ejecución	9
2.6.2	Trazas	10
3	Servicio DNS	13
3.1	Descripción del servicio	13
3.2	Descripción del escenario	13
3.3	Configuración	13
3.3.1	BIND	14
3.4	Análisis	16
3.4.1	Trazas	16
4	Servicio SMTP/POP	19
4.1	Descripción del servicio	19
4.2	Descripción del escenario	19
4.3	Configuración	19
4.3.1	Exim4	19
4.3.2	Dovecot	20
4.3.3	Configuración de cliente SMTP/POP	20
4.4	Análisis	22
4.4.1	Ejecución	22
4.4.2	Trazas	22
5	Servicios HTTP/HTTPS	25
5.1	Descripción del servicio	25
5.2	Descripción del escenario	25
5.3	Configuración	25
5.3.1	Servidor	25
5.3.2	Cliente	29
5.4	Análisis	30
5.4.1	Ejecución	30
5.4.2	Trazas	32
6	IPsec	35
6.1	Descripción del servicio	35
6.2	Descripción del escenario	35
6.3	Configuración	35
6.3.1	Ficheros de configuración	35
6.3.2	Uso de certificados X.509	37
6.4	Análisis	37
6.4.1	Trazas	37

7	Conclusiones	40
7.1	Horas Empleadas	40
7.2	Problemas Encontrados	40

List of Figures

1	Escenario Objetivo	4
2	Módulos apache2.conf	7
3	apache2/ports.conf	7
4	apache2/sites-available	8
5	Ejemplo fichero 'VirtualHost'	8
6	Authentication Required	9
7	nebulanexus1345.com	10
8	Traza HTTP	10
9	Contenido HTTP	11
10	Petición GET con credenciales correctas	12
11	Respuesta del servidor 200 OK	12
12	Fichero '/etc/resolv.conf' en el cliente	14
13	Jerarquía /etc/bind	14
14	Fichero /etc/bind/named.conf.options	15
15	Fichero /etc/bind/named.conf.local	15
16	Fichero db de DNS del sitio web	16
17	Traza DNS	17
18	Paquete 1 DNS	17
19	Paquete 2 DNS	18
20	Configuración Servidor en Thunderbird	21
21	Configuración Servidor de Salida (SMTP)	21
22	Escribimos un correo para enviar por SMTP	22
23	Recibimos el correo por POP	22
24	Interacción de Thunderbird con el servidor SMTP	23
25	Interacción de Thunderbird con el servidor POP	23
26	Correo siendo enviado por SMTP	23
27	Correo siendo recibido por POP	24
28	Descifrando el contenido del AUTH con cyberchef	24
29	[CA_default] en openssl.cnf	26
30	[req_distinguished_name] en openssl.cnf	27
31	Directorio /home/alumno/demoCA	27
32	Contenido servercert.pem	28
33	Puerto 443	29
34	SSL en nebulanexus.conf	29
35	Utilizamos el certificado generado por el servidor	31
36	El candado verde y la URL confirman que nos encontramos en HTTPS	31
37	Interacción completa de TLS y TCP	32
38	source: cloudfare.com	32
39	Primer mensaje del handshake	33
40	Segundo paquete del handshake	33
41	Tercer paquete del handshake	34
42	Último paquete del handshake	34
43	Fichero /etc/ipsec.conf	36
44	Fichero /etc/ipsec.secrets	36
45	Handshake de IKE completo	38
46	El paquete ESP contiene un paquete de DNS	38
47	El paquete ESP contiene un paquete de HTTP	38
48	El paquete ESP contiene un paquete de SMTP	39
49	El paquete ESP contiene un paquete de POP	39

1 Introducción

1.1 Propósito

La práctica consiste en la configuración y puesta en marcha de un conjunto de servicios telemáticos vistos en la asignatura para una organización, sobre el escenario de red que se muestra a continuación:

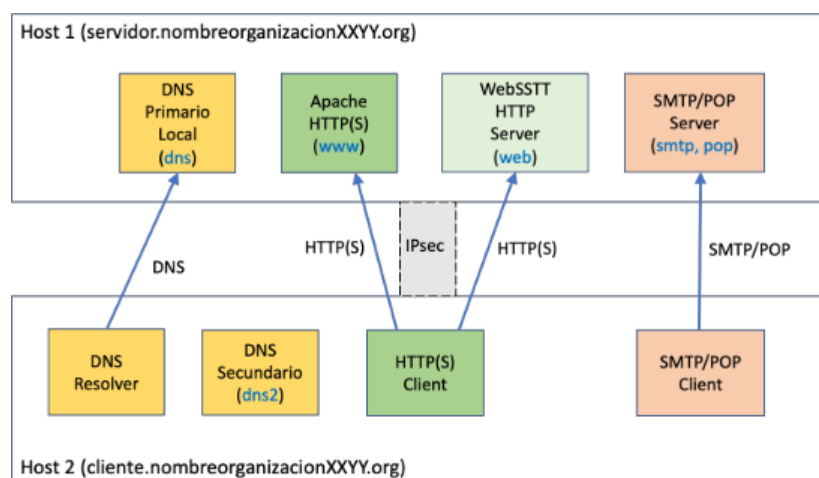


Figure 1: Escenario Objetivo

El proyecto que desarrollamos como parte de nuestra asignatura de servicios telemáticos tiene como objetivo implementar y configurar un conjunto de servicios fundamentales sobre una infraestructura de red diseñada específicamente para una organización ficticia, la cual hemos denominado "nebulanexus1345.com".

Este proyecto busca abordar varios aspectos técnicos y prácticos fundamentales en la implementación de servicios de red. En particular, se enfoca en el despliegue y configuración de servicios HTTP, DNS, SMTP/POP, HTTP/HTTPS y IPsec, cada uno de ellos esencial para el funcionamiento eficiente de las operaciones diarias de una organización.

A través de esta experiencia práctica, pretendemos no solo aplicar los conocimientos teóricos adquiridos durante el curso, sino también enfrentarnos a los desafíos reales que implican la configuración y mantenimiento de estos servicios en un entorno controlado pero realista.

El diseño del proyecto se ha basado en una topología de red específica, incluyendo servidores y clientes configurados bajo el dominio "nebulanexus1345". Este enfoque nos permite explorar la interacción entre diferentes tecnologías y la integración de múltiples servicios en un sistema cohesivo.

En las siguientes secciones de esta memoria, describiremos detalladamente cada uno de los servicios implementados, explicando las decisiones de diseño, configuración y los resultados obtenidos.

2 Web-SSTT HTTP Server

2.1 Descripción del servicio

El servicio HTTP (Hypertext Transfer Protocol) es un protocolo de la capa de aplicación utilizado para transmitir documentos de hipertexto, como HTML. Es la base de cualquier intercambio de datos en la Web y es un protocolo orientado a transacciones y sigue el modelo solicitud-respuesta entre un cliente y un servidor.

Cuando un usuario quiere acceder a una página web, su navegador (el cliente) realiza una solicitud HTTP al servidor donde reside la página. Esta solicitud incluye métodos como GET para solicitar datos del servidor o POST para enviar datos al servidor. Una vez que el servidor recibe y procesa la solicitud, envía una respuesta al cliente. Esta respuesta puede incluir el contenido solicitado, como una página HTML, y metadatos sobre el contenido en forma de encabezados HTTP.

La implementación de este servicio la hacemos basandonos en el fichero "web.sstt", fichero que hemos desarrollado previamente y que es un script básico de servidor HTTP, maneja solicitudes simples y archivos estáticos. En concreto este servidor cumpliría con las siguientes características:

- Gestión básica del método GET en peticiones HTTP request que provienen del cliente y su correspondiente respuesta.
- Gestionar una petición HTTP con el método POST básica de un formulario que el servidor devolverá al cliente.
- Gestión básica de cookies en el servidor Web-SSTT HTTP. Se enviará una cookie con un contador de accesos al servidor de modo que al décimo intento de acceso se denegará el acceso al contenido. Esta cookie expirará a los 2 minutos de su creación.
- Mecanismo de persistencia HTTP. El servicio deberá mantener una conexión abierta durante un tiempo determinado y, en el caso de no recibir peticiones durante ese periodo de tiempo, se terminará la conexión.
- Verificación de que la petición HTTP es válida.
- Verificar que en la petición se ha incluido la cabecera Host.

2.2 Funcionalidad básica

1. Inicialización y Configuración

- El script acepta argumentos de línea de comandos para configurar la dirección IP (-host), el puerto (-port) y la ubicación del directorio raíz desde donde se servirán los archivos (-webroot). También tiene una opción para aumentar el nivel de detalle de los logs (-verbose).

2. Servidor Socket

- Crea un socket TCP y lo configura para escuchar en la dirección IP y puerto especificados.
- El servidor puede reutilizar direcciones IP previamente asignadas (SO_REUSEADDR) para evitar errores de "dirección ya en uso".

3. Gestión de Conexiones

- Utiliza un bucle infinito para aceptar conexiones de clientes.

- Cada conexión aceptada genera un proceso hijo mediante `fork()` para manejar la solicitud del cliente, mientras que el proceso principal sigue escuchando más conexiones.

4. Procesamiento de Solicitudes

- Recibe datos HTTP del cliente, interpreta la solicitud HTTP (analizando los métodos, la URL, y las cabeceras).
- Verifica la versión de HTTP, el método de la solicitud, y la presencia y validez del recurso solicitado.
- Si se solicita la raíz ('/'), automáticamente se redirige a `/index.html`.
- Maneja cookies para contar y limitar el número de accesos.

5. Respuestas

- Construye respuestas HTTP apropiadas según la solicitud, incluyendo manejo de errores.
- Envía el contenido del archivo solicitado si existe, con las cabeceras HTTP adecuadas (incluyendo tipo de contenido y longitud).

2.3 Gestión de errores

1. **Solicitudes mal formadas:** Si la solicitud no cumple con el formato esperado de HTTP, se devuelve un error **'400 Bad Request'**.
2. **Método HTTP No Soportado:** Si se utiliza un método distinto de GET o POST, se devuelve un error **'405 Method Not Allowed'**.
3. **Versión de HTTP No Soportada:** Si la versión de HTTP no es 1.1, se devuelve un error **'505 HTTP Version Not Supported'**.
4. **Archivo No Encontrado:** Si el recurso solicitado no existe, se devuelve un error **'404 Not Found'**.
5. **Número Máximo de Accesos:** Si se excede el número máximo de accesos permitidos (definido por las cookies), se devuelve un error **'403 Forbidden'**.
6. **Timeout:** Si no se recibe ninguna solicitud durante un periodo especificado (`TIMEOUT_CONNECTION`), se cierra la conexión con un error **'408 Request Timeout'**.
7. Cualquier excepción no capturada durante el procesamiento de la solicitud resulta en un error **'500 Internal Server Error'**.

2.4 Apache HTTP

- Características principales
 1. Soporte de Múltiples Lenguajes y Tecnologías: Apache puede servir archivos HTML estáticos y también es capaz de integrarse con lenguajes de programación como PHP, Python, Perl, y otros a través de módulos de extensión. Esto permite a Apache servir como la base para aplicaciones web dinámicas.
 2. Configuración Modular: Apache funciona mediante un sistema de módulos que pueden cargarse o descargarse para agregar o quitar funcionalidades. Esto incluye módulos para autenticación, compresión, SSL, reescritura de URLs, entre muchos otros.

3. Soporte de SSL/TLS: A través del módulo `mod_ssl`, Apache puede configurarse para servir contenido de forma segura usando criptografía SSL/TLS, lo que es esencial para proteger la transmisión de datos sensibles.
 4. Autenticación y Autorización: Apache ofrece extensas capacidades para controlar el acceso a recursos del servidor, incluyendo autenticación básica y digest, así como integración con sistemas externos de gestión de usuarios.
 5. Virtual Hosting: Apache puede configurarse para servir múltiples sitios web desde un solo servidor físico. Esto se conoce como hosting virtual, y puede basarse en diferencias de IP, puerto o nombre de host.
- Seguridad: Apache incluye una variedad de configuraciones de seguridad, incluyendo la capacidad de restringir acceso por dirección IP, configurar listas de control de acceso, y realizar bloqueos basados en características de la solicitud.
 - Extensibilidad y Soporte: Apache es extremadamente extensible mediante sus módulos y tiene una vasta comunidad de usuarios y desarrolladores que constantemente contribuyen con actualizaciones, seguridad y nuevas características.

2.5 Configuración de Apache HTTP

El archivo `'apache2.conf'` es el archivo principal de configuración de Apache HTTP Server en sistemas basados en Debian y Ubuntu. Contiene la configuración global del servidor Apache y establece directrices para el comportamiento general del servidor, así como la configuración predeterminada que se aplicará a todos los sitios virtuales y directorios de un servidor. Este archivo incluye otros ficheros para modularizar y simplificar la configuración. En el desarrollo de esta práctica utilizamos los siguientes:

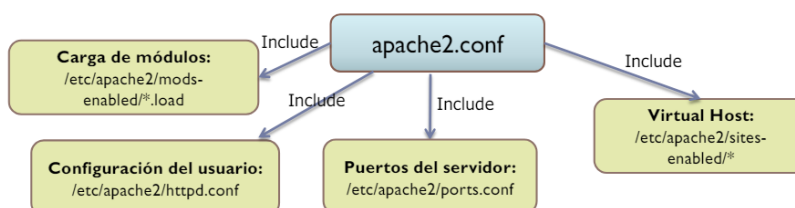


Figure 2: Módulos `apache2.conf`

1. **`/etc/apache2/ports.conf`:** el archivo `ports.conf` en Apache HTTP Server es utilizado para configurar los puertos en los que el servidor debe escuchar las conexiones entrantes.

```
Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

Figure 3: `apache2/ports.conf`

De este fichero podemos observar que en nuestro caso Apache está escuchando las solicitudes HTTP en el puerto 80 (puerto por defecto). También podemos ver cláusulas del tipo `<IfModule>` que están relacionadas con la seguridad SSL/TLS y de las que hablaremos más adelante en este documento.

2. **/etc/apache2/sites-available:** en este directorio es donde se encuentran los archivos de configuración de los sitios web disponibles para Apache. En este contexto cada archivo representa lo que es conocido como 'VirtualHost' y definen como Apache ha de manejar las solicitudes para cada sitio web.

```
alumno@server:/etc/apache2/sites-available$ ls
000-default.conf  default-ssl.conf  example.conf  nebulaNexus.conf  prueba.conf  st1.conf  st2.conf
```

Figure 4: apache2/sites-available

```
alumno@server:/etc/apache2/sites-available$ cat nebulaNexus.conf
<VirtualHost *:443>
    ServerAdmin edu@nebulanexus1345.com
    ServerName nebulanexus1345.com
    DocumentRoot /var/www/mi_web
    <Directory /var/www/mi_web>
        AllowOverride AuthConfig
        AuthType Basic
        AuthName "Acceso restringido a trabajadores"
        AuthBasicProvider file
        AuthUserFile /etc/apache2/passwords
        AuthGroupFile /etc/apache2/groups
        Require group trabajadores
        Order allow,deny
        allow from all
    </Directory>
    SSLEngine on
    SSLCertificateFile /home/alumno/demoCA/servercert.pem
    SSLCertificateKeyFile /home/alumno/demoCA/serverkey.pem
    SSLCACertificateFile /home/alumno/demoCA/cacert.pem
    SSLVerifyClient require
    SSLVerifyDepth 10
</VirtualHost>
```

Figure 5: Ejemplo fichero 'VirtualHost'

En la Figura 5 podemos ver como es un fichero 'VirtualHost', en este caso el fichero pertenece a la organización en la que trabajamos en esta práctica, 'nebulaNexus', y de este fichero podemos concluir:

- El VirtualHost escucha en todas las direcciones IP (*) en el puerto 443.
- Especifica la dirección de correo del administrador del servidor (edu@nebulanexus1345.com)
- Especifica el nombre de dominio 'nebulanexus1345.com'.
- 'DocumentRoot /var/www/mi_web': Define la ruta del directorio raíz del documento para este VirtualHost.
- Dentro de `<Directory...>` se define la configuración específica para el directorio '/var/www/mi_web'.
- 'AllowOverride AuthConfig': Permite que se utilicen directivas de configuración de autenticación (AuthConfig) en archivos de configuración '.htaccess'.
- 'AuthType Basic': Especifica que se utilizará la autenticación básica para proteger el acceso al sitio web.
- 'AuthUserFile /etc/apache2/passwords': Especifica la ubicación del archivo de contraseñas que contiene los usuarios y contraseñas para la autenticación básica.
- 'AuthGroupFile /etc/apache2/groups': Especifica la ubicación del archivo de grupos que contiene los grupos de usuarios para la autenticación básica.
- Solo permite el acceso a usuarios que pertenecen al grupo 'trabajadores'.

Las últimas líneas del fichero hacen referencia a la configuración de HTTPS que se explicará más adelante en este documento.

3. **/etc/apache2/sites-enabled:** se trata de un directorio en donde se encuentran los archivos de configuración de los sitios web activos que Apache está sirviendo actualmente. Cada archivo de configuración en este directorio corresponde a un sitio web específico y contiene las directivas de configuración necesarias para ese sitio. En resumen en este directorio se encuentran los fichero 'VirtualHost' de 'sites-available' que hayan sido activados.
4. **/etc/apache2/mods-enabled:** contiene enlaces simbólicos a los módulos de Apache que están habilitados en la configuración del servidor. Cada enlace simbólico en este directorio apunta a un archivo correspondiente en el directorio '/etc/apache2/mods-available/', que es donde residen los archivos de configuración de los módulos disponibles para Apache.

2.6 Análisis

El protocolo HTTP es utilizado en la mayor parte de casos a través de un navegador web, en el nuestro utilizaremos Firefox.

2.6.1 Ejecución

Para el usuario, acceder a la página web sería tan sencillo como introducir el dominio en el navegador. Esta dirección es <http://nebulanexus1345.com>, hay que tener en cuenta que se tiene que especificar HTTP para no utilizar HTTPS en su lugar, el cual se verá más adelante.

Una vez introducimos el dominio, nos saltará una ventana emergente. Si recordamos, en el fichero de configuración se ha especificado que solo deberían poder acceder a esta página aquellos usuarios que pertenecen al grupo de trabajadores. Hemos creado un par de usuarios, llamados edu (contraseña edu) y alf, que pertenecen a este grupo.

Cuando se entra a la página web, lo primero que nos saldrá será una ventana emergente pidiendo autenticación. Debemos entrar el nombre y contraseña de uno de estos usuarios.

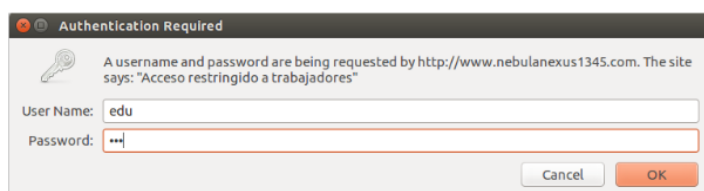


Figure 6: Authentication Required

Una vez autenticados, podemos acceder a la página web. Los ficheros utilizados para la página son los provistos por la práctica.



Figure 7: nebulaNexus1345.com

2.6.2 Trazas

En esta última parte vamos a comprobar cómo ha actuado el servidor durante esta interacción del apartado anterior. La traza generada ha sido la siguiente:

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000000000	192.168.56.103	192.168.56.102	TCP	76	49346 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=855653 TSecr=0 WS=128
6	0.001255934	192.168.56.102	192.168.56.103	TCP	76	80 → 49346 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=418335 TSecr=855653 WS=128
7	0.001284479	192.168.56.103	192.168.56.102	TCP	68	49346 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=855653 TSecr=418335
8	0.001375709	192.168.56.103	192.168.56.102	HTTP	369	GET / HTTP/1.1
9	0.001578171	192.168.56.102	192.168.56.103	TCP	68	80 → 49346 [ACK] Seq=1 Ack=302 Win=38888 Len=0 TSval=418335 TSecr=855653
10	0.001790671	192.168.56.102	192.168.56.103	HTTP	825	HTTP/1.1 401 Unauthorized (text/html)
11	0.001795979	192.168.56.103	192.168.56.102	TCP	68	49346 → 80 [ACK] Seq=302 Ack=758 Win=30720 Len=0 TSval=855654 TSecr=418335
12	2.016210816	192.168.56.103	192.168.56.102	HTTP	404	GET / HTTP/1.1
13	2.017351034	192.168.56.102	192.168.56.103	HTTP	897	HTTP/1.1 200 OK (text/html)
14	2.017387561	192.168.56.103	192.168.56.102	TCP	68	49346 → 80 [ACK] Seq=638 Ack=1387 Win=32256 Len=0 TSval=856382 TSecr=419064
15	2.091775673	192.168.56.103	192.168.56.102	HTTP	427	GET /logo-un.jpg HTTP/1.1
16	2.983365887	192.168.56.102	192.168.56.103	HTTP	9913	HTTP/1.1 200 OK (JPEG image)
17	2.983391625	192.168.56.103	192.168.56.102	TCP	68	49346 → 80 [ACK] Seq=997 Ack=11232 Win=51068 Len=0 TSval=856401 TSecr=419083
18	7.908146901	192.168.56.102	192.168.56.103	TCP	68	80 → 49346 [FIN, ACK] Seq=11232 Ack=997 Win=32256 Len=0 TSval=420312 TSecr=856401
19	7.908434642	192.168.56.103	192.168.56.102	TCP	68	49346 → 80 [FIN, ACK] Seq=997 Ack=11233 Win=51068 Len=0 TSval=857630 TSecr=420312
20	7.908778529	192.168.56.102	192.168.56.103	TCP	68	80 → 49346 [ACK] Seq=11233 Ack=998 Win=32256 Len=0 TSval=420312 TSecr=857630

Figure 8: Taza HTTP

Si recordamos el funcionamiento del protocolo HTTP, este comienza con una interacción TCP para establecer el canal de comunicación. Los dos primeros mensajes son de tipo SYN que inician la conexión TCP que inicia el cliente con el servidor. Sin embargo, los dos siguientes mensajes (7 y 8) no corresponden con el intercambio de mensajes ACK como especifica el protocolo. En su lugar, tenemos el primer mensaje ACK del cliente y la primera petición HTTP del index.html.

En nuestra página web, como requiere de autenticación, la petición contiene un campo llamado Credential (credenciales) que especifica el usuario siendo utilizado con la estructura usuario:contraseña.

Continuando con el resto de la interacción, la podemos dividir en tres partes: la de petición del index.html (que se repite dos veces porque en el primer intento las credenciales no son válidas), la petición de la imagen y el cierre de la conexión TCP. Si nos fijamos en el tipo de los mensajes siendo enviados, el que ha iniciado esa parte de la interacción confirma al otro que su último mensaje ha sido recibido correctamente con un mensaje ACK de TCP. Las dos primeras partes son iniciadas por el cliente, mientras que la última, el cierre de la conexión es iniciado por el servidor.

5-6) Son los mensajes TCP SYN del cliente y servidor respectivamente.

7-8) El primer mensaje de este es el TCP ACK del cliente y el segundo el a petición de la página web con GET. En este caso las credenciales son incorrectas y se mostrará la traza cuando lo son.

9-10) El servidor responde termina de negociar la conexión TCP con otro ACK y responde al cliente con un mensaje de error 401. Como podemos ver el contenido no va vacío sino que se envía una página web que explica el error al cliente.

```

Hypertext Transfer Protocol
  HTTP/1.1 401 Unauthorized\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 401 Unauthorized\r\n]
      Response Version: HTTP/1.1
      Status Code: 401
      [Status Code Description: Unauthorized]
      Response Phrase: Unauthorized
      Date: Sun, 12 May 2024 11:36:53 GMT\r\n
      Server: Apache/2.4.18 (Ubuntu)\r\n
      WWW-Authenticate: Basic realm="Acceso restringido a trabajadores"\r\n
  Content-Length: 470\r\n
    [Content length: 470]
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=iso-8859-1\r\n
    \r\n
    [HTTP response 1/4]
    [Time since request: 0.000612257 seconds]
0000 6f 72 65 73 22 0d 0a 43 6f 6e 74 65 6b 74 2d 4c 0005 0006 0007 0008 0009 0010 0011 0012 0013 0014 0015 0016 0017 0018 0019 0020 0021 0022 0023 0024 0025 0026 0027 0028 0029 0030 0031 0032 0033 0034 0035 0036 0037 0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069 0070 0071 0072 0073 0074 0075 0076 0077 0078 0079 0080 0081 0082 0083 0084 0085 0086 0087 0088 0089 0090 0091 0092 0093 0094 0095 0096 0097 0098 0099 0100 0101 0102 0103 0104 0105 0106 0107 0108 0109 0110 0111 0112 0113 0114 0115 0116 0117 0118 0119 0120 0121 0122 0123 0124 0125 0126 0127 0128 0129 0130 0131 0132 0133 0134 0135 0136 0137 0138 0139 0140 0141 0142 0143 0144 0145 0146 0147 0148 0149 0150 0151 0152 0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0168 0169 0170 0171 0172 0173 0174 0175 0176 0177 0178 0179 0180 0181 0182 0183 0184 0185 0186 0187 0188 0189 0190 0191 0192 0193 0194 0195 0196 0197 0198 0199 0200 0201 0202 0203 0204 0205 0206 0207 0208 0209 0210 0211 0212 0213 0214 0215 0216 0217 0218 0219 0220 0221 0222 0223 0224 0225 0226 0227 0228 0229 0230 0231 0232 0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247 0248 0249 0250 0251 0252 0253 0254 0255 0256 0257 0258 0259 0260 0261 0262 0263 0264 0265 0266 0267 0268 0269 0270 0271 0272 0273 0274 0275 0276 0277 0278 0279 0280 0281 0282 0283 0284 0285 0286 0287 0288 0289 0290 0291 0292 0293 0294 0295 0296 0297 0298 0299 0300 0301 0302 0303 0304 0305 0306 0307 0308 0309 0310 0311 0312 0313 0314 0315 0316 0317 0318 0319 0320 0321 0322 0323 0324 0325 0326 0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337 0338 0339 0340 0341 0342 0343 0344 0345 0346 0347 0348 0349 0350 0351 0352 0353 0354 0355 0356 0357 0358 0359 0360 0361 0362 0363 0364 0365 0366 0367 0368 0369 0370 0371 0372 0373 0374 0375 0376 0377 0378 0379 0380 0381 0382 0383 0384 0385 0386 0387 0388 0389 0390 0391 0392 0393 0394 0395 0396 0397 0398 0399 0400 0401 0402 0403 0404 0405 0406 0407 0408 0409 0410 0411 0412 0413 0414 0415 0416 0417 0418 0419 0420 0421 0422 0423 0424 0425 0426 0427 0428 0429 0430 0431 0432 0433 0434 0435 0436 0437 0438 0439 0440 0441 0442 0443 0444 0445 0446 0447 0448 0449 0450 0451 0452 0453 0454 0455 0456 0457 0458 0459 0460 0461 0462 0463 0464 0465 0466 0467 0468 0469 0470 0471 0472 0473 0474 0475 0476 0477 0478 0479 0480 0481 0482 0483 0484 0485 0486 0487 0488 0489 0490 0491 0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0504 0505 0506 0507 0508 0509 0510 0511 0512 0513 0514 0515 0516 0517 0518 0519 0520 0521 0522 0523 0524 0525 0526 0527 0528 0529 0530 0531 0532 0533 0534 0535 0536 0537 0538 0539 0540 0541 0542 0543 0544 0545 0546 0547 0548 0549 0550 0551 0552 0553 0554 0555 0556 0557 0558 0559 0560 0561 0562 0563 0564 0565 0566 0567 0568 0569 0570 0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585 0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596 0597 0598 0599 0600 0601 0602 0603 0604 0605 0606 0607 0608 0609 0610 0611 0612 0613 0614 0615 0616 0617 0618 0619 0620 0621 0622 0623 0624 0625 0626 0627 0628 0629 0630 0631 0632 0633 0634 0635 0636 0637 0638 0639 0640 0641 0642 0643 0644 0645 0646 0647 0648 0649 0650 0651 0652 0653 0654 0655 0656 0657 0658 0659 0660 0661 0662 0663 0664 0665 0666 0667 0668 0669 0670 0671 0672 0673 0674 0675 0676 0677 0678 0679 0680 0681 0682 0683 0684 0685 0686 0687 0688 0689 0690 0691 0692 0693 0694 0695 0696 0697 0698 0699 0700 0701 0702 0703 0704 0705 0706 0707 0708 0709 0710 0711 0712 0713 0714 0715 0716 0717 0718 0719 0720 0721 0722 0723 0724 0725 0726 0727 0728 0729 0730 0731 0732 0733 0734 0735 0736 0737 0738 0739 0740 0741 0742 0743 0744 0745 0746 0747 0748 0749 0750 0751 0752 0753 0754 0755 0756 0757 0758 0759 0760 0761 0762 0763 0
```

Figure 9: Contenido HTTP

11-12) El cliente vuelve a intentar realizar la misma petición que la última vez, pero esta vez las credenciales sí que son correctas.

13) El servidor vuelve a responder, pero esta vez como las credenciales son correctas, se limita a enviar la página web con el código OK 200.

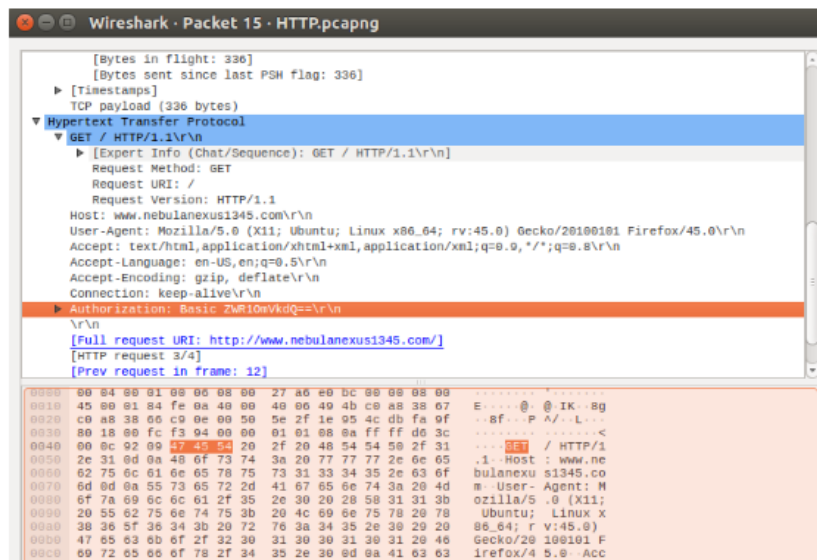


Figure 10: Petición GET con credenciales correctas

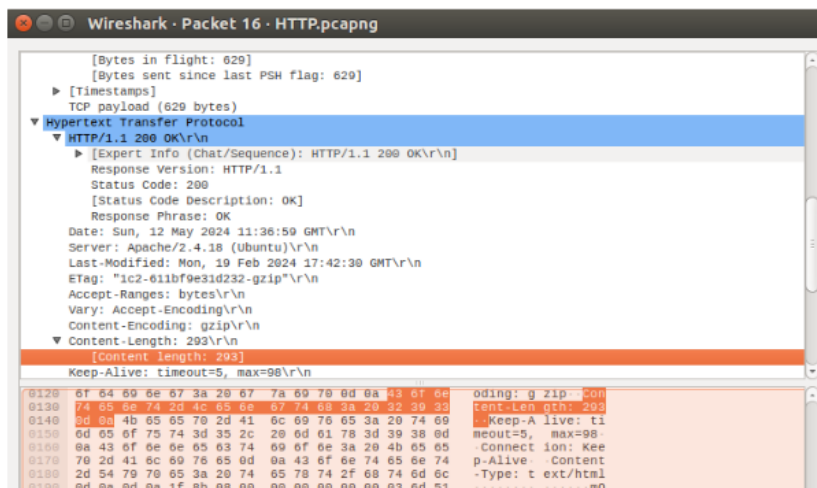


Figure 11: Respuesta del servidor 200 OK

14) Es el mensaje de confirmación del cliente al servidor para que sepa que le ha llegado el elemento enviado, en este caso el fichero index.html.

15-20) La interacción es esencialmente la misma pero pidiendo la imagen.

21) El servidor termina la interacción HTTP con un mensaje TCP FIN ACK.

22) El cliente confirma el final con otro mensaje TCP FIN ACK.

23) El servidor confirma que le ha llegado la confirmación del fin de la conexión.

3 Servicio DNS

3.1 Descripción del servicio

El DNS (Domain Name System) es un sistema distribuido y jerárquico utilizado para traducir nombres de dominio legibles por humanos, como "ejemplo.com", en direcciones IP numéricas, como "192.0.2.1", utilizadas por las máquinas en internet.

- **Jerarquía de servidores:** El DNS opera mediante una estructura jerárquica de servidores, comenzando con los servidores raíz en la cima, seguidos de servidores de dominio de nivel superior (TLD), servidores de nombres autoritativos y servidores de caché locales.
- **Funcionamiento de resolución:** Cuando un usuario solicita acceder a un sitio web, su dispositivo realiza una consulta DNS a su servidor DNS local. Si el servidor local no tiene la información en caché, inicia un proceso de resolución que puede implicar consultas a varios servidores DNS en la jerarquía hasta encontrar la dirección IP correspondiente al nombre de dominio solicitado.
- **Tipos de registros:** El DNS admite varios tipos de registros que almacenan información asociada con un nombre de dominio, como registros A (dirección IPv4), registros AAAA (dirección IPv6), registros MX (servidor de correo), registros CNAME (alias de nombre canónico), registros TXT (texto arbitrario), entre otros.
- **Caché:** Los servidores DNS locales almacenan en caché las respuestas a las consultas DNS para mejorar la eficiencia y reducir la carga en la red. Esto permite que las consultas futuras para los mismos nombres de dominio se resuelvan más rápidamente.
- **Zonas y registros de recursos:** La información en el DNS se organiza en zonas, que representan áreas de autoridad administrativa sobre porciones del espacio de nombres de dominio. Cada zona contiene registros de recursos que describen la información asociada con los nombres de dominio dentro de esa zona, como direcciones IP, servidores de correo, etc.

3.2 Descripción del escenario

3.3 Configuración

Ficheros importantes en el cliente:

- `/etc/hosts`: mapea nombres de host a direcciones IP en una red local. Este archivo se utiliza como una forma básica de resolución de nombres de dominio antes de realizar consultas al servidor DNS.
- `/etc/resolv.conf`: Define el dominio de la máquina, los dominios de búsqueda y los servidores DNS. En esta práctica incluimos la directiva 'nameserver' apuntando al servidor DNS local en la máquina servidor.

```
# 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
# DNS 1 del Servidor
nameserver 192.168.56.102
# DNS 2 del Cliente
nameserver 192.168.56.103
nameserver 192.168.56.102
nameserver 8.8.8.8
```

Figure 12: Fichero '/etc/resolv.conf' en el cliente

3.3.1 BIND

BIND (Berkeley Internet Name Domain) es el servidor de nombres de dominio (DNS) más utilizado en internet. BIND actúa como un servidor DNS que responde a consultas de resolución de nombres de dominio, traduciendo nombres de dominio legibles por humanos en direcciones IP numéricas y viceversa. También puede actuar como un servidor de nombres autoritativo, almacenando y distribuyendo información sobre los nombres de dominio bajo su autoridad. Implementa la estructura jerárquica estándar de servidores DNS, incluidos los servidores raíz, los servidores de dominio de nivel superior (TLD) y los servidores de nombres autoritativos. Esta jerarquía permite una resolución eficiente de nombres de dominio en internet.

En /etc/bind podemos encontrar los siguientes ficheros:

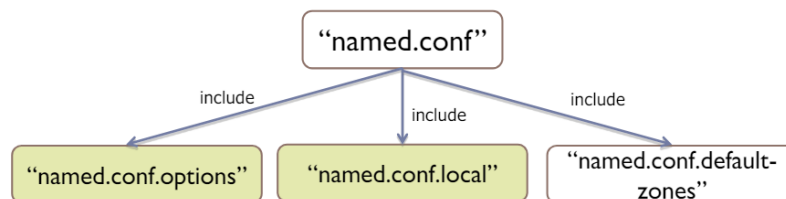


Figure 13: Jerarquía /etc/bind

1. **'named.conf.options'**: Opciones de configuración de servidor globales. Se especifica la ruta del directorio de trabajo y para el desarrollo de esta práctica la directiva `dnssec-validation` queda desactivada, ya que no hacemos uso de DNSSEC (Domain Name System Security Extensions).

```
options {  
    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.  
  
    // 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  
    dnssec-validation no;  
  
    auth-nxdomain no;    # conform to RFC1035  
    listen-on-v6 { any; };  
};
```

Figure 14: Fichero /etc/bind/named.conf.options

2. **'named.conf.local'**: Se definen las características de una zona administrada por este servidor DNS. Para nuestra organización queda así:

```
zone "nebulanexus1345.com" {  
    type master;  
    file "/etc/bind/zones/db.nebulanexus.com.zone";  
    allow-transfer { 192.168.56.103; };  
    also-notify { 192.168.56.103; };  
};
```

Figure 15: Fichero /etc/bind/named.conf.local

- **'zone "nebulanexus1345.com"'**: indica el inicio de la configuración para la zona de dominio nebulanexus1345.com.
 - **'type master'**: indica que el servidor BIND es el servidor maestro para esta zona de dominio.
 - **file "/etc/bind/zones/db.nebulanexus1345.com.zone"**: especifica la ruta del archivo de zona que contiene los datos de la zona de dominio.
 - **allow-transfer 192.168.56.103; :** Esta línea especifica qué servidores están autorizados para transferir (copiar) la zona de dominio desde este servidor BIND.
 - **also-notify 192.168.56.103; :** especifica una lista de servidores que deben recibir notificaciones de zona cuando ocurran cambios en la zona de dominio.
3. **/etc/bind/zones/db.nebulanexus1345.com.zone**: configura los parámetros básicos y el registro SOA para la zona de difusión en BIND. Define la autoridad para la zona y establece los valores de tiempo para la actualización y expiración de la zona, así como el TTL predeterminado para los registros de la zona.


```

; BIND reverse data file for empty rfc1918 zone
;
; DO NOT EDIT THIS FILE - it is used for multiple zones.
; Instead, copy it, edit named.conf, and use that copy.
;
$TTL      604800
@         IN      SOA     nebulanexus1345.com. root.nebulanexus1345.com. (
                                1           ; Serial
                                604800      ; Refresh
                                86400       ; Retry
                                2419200     ; Expire
                                604800 )    ; Negative Cache TTL
;
; Name server
@         IN      NS      dns1.nebulanexus1345.com.
@         IN      NS      dns2.nebulanexus1345.com.
dns1      IN      A       192.168.56.102
dns2      IN      A       192.168.56.103
;
; Nombre Raiz
@         IN      A       192.168.56.102
;
; Servidores Mail
smtp      IN      A       192.168.56.102
pop       IN      A       192.168.56.102
@         IN      MX      10 smtp.nebulanexus1345.com.
;
; Redirecciones
www       IN      CNAME   nebulanexus1345.com.
servidor  IN      CNAME   nebulanexus1345.com.
web       IN      CNAME   nebulanexus1345.com.
;
cliente   IN      A       192.168.56.103
;
;
;
"db.nebulanexus.com.zone" 32L, 949C

```

Figure 16: Fichero db de DNS del sitio web

3.4 Análisis

El protocolo DNS es utilizado por el ordenador cada vez que se quiere acceder a una página web a través de su URL en vez de su IP. Como el tráfico se enruta a través de IPs, cuando se solicita una URL se necesita conocer la IP asociada.

De hecho, antes de que se pudiese ejecutar el protocolo HTTP en el ejemplo del capítulo anterior, Firefox tuvo que hacer una consulta DNS al servidor que hemos configurado. Como nuestro dominio solo va a ser utilizado localmente, no podemos realizar consultas de DNS a servidores externos para saber su IP. En su lugar, hemos configurado un servidor DNS en la propia máquina servidor (192.168.56.102) que hará de DNS principal y uno en la máquina cliente (192.168.56.103) que hará de secundario (o de maestro y esclavo, respectivamente). Estos servidores deben poder acceder a los servidores DNS Raíz, así como tener guardado localmente todas las direcciones pertinentes a nuestro dominio nebulanexus1345.com.

3.4.1 Trazas

Como hemos comentado antes los clientes web, como firefox, realizan peticiones DNS cuando necesitan conectarse a una URL. Es por esto que vamos a analizar la traza de un acceso que se ha realizado a la página web de nebulanexus1345.com en la prueba de TLS.

Firefox realiza dos peticiones por página web, una del registro A de la propia página y otra del registro AAAA. Por esta razón en la foto de abajo tenemos los dos primeros mensajes de la conversación subrayada en salmón abajo son las peticiones de estos dos registros y las dos siguientes son las repuestas. Como son muy similares, solo vamos a analizar las relacionadas con el registro A ya que son más completas.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.103	192.168.56.102	DNS	83	Standard query 0x62cc A www.nebulaexus1345.com
2	0.000000000	192.168.56.103	192.168.56.102	DNS	83	Standard query response 0xb2a8 AAAA www.nebulaexus1345.com
3	0.000000000	192.168.56.103	192.168.56.102	DNS	138	Standard query response 0xb2a8 AAAA www.nebulaexus1345.com

Figure 17: Traza DNS

En el mensaje 1 podemos ver un mensaje DNS del cliente al servidor que solo contiene una entrada de tipo Queries, por lo que el contador Questions está a 1. La URL de esta consulta tiene que ser igual a la escrita en la barra de búsqueda en el propio cliente.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.56.103	192.168.56.102	DNS	83	Standard query 0x62cc A www.nebulaexus1345.com
2	0.000000000	192.168.56.103	192.168.56.102	DNS	83	Standard query 0xb2a8 AAAA www.nebulaexus1345.com
3	0.000000000	192.168.56.103	192.168.56.102	DNS	138	Standard query response 0xb2a8 AAAA www.nebulaexus1345.com

Frame 1: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
 Ethernet II, Src: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc), Dst: PcsCompu_74:0f:ba (08:00:27:74:0f:ba)
 Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.102
 User Datagram Protocol, Src Port: 38569, Dst Port: 53

Cliente de correo Thunderbird

Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0

Queries
 www.nebulaexus1345.com: type A, class IN
 [Response in: 3]

Figure 18: Paquete 1 DNS

A su vez, la respuesta del servidor contiene la pregunta y también la información de respuesta. En la sección de respuestas tenemos los registros que nos llevan a la IP deseada siendo uno de tipo CNAME que especifica la dirección absoluta de la URL y otra de tipo A que contiene la IP de esta URL absoluta.

3	0.000465829	192.168.56.102	192.168.56.103	DNS	183 Standard query response 0x62cc A www.n
4	0.000478278	192.168.56.102	192.168.56.103	DNS	138 Standard query response 0xb2a8 AAAA ww
12	2.590378031	192.168.56.103	192.168.56.102	DNS	83 Standard query 0xee97 A self-repair.m0
13	2.590400365	192.168.56.103	192.168.56.102	DNS	83 Standard query 0xa696 AAAA self-repair
14	2.590908498	192.168.56.102	192.168.56.103	DNS	461 Standard query response 0xee97 A self-
15	2.590983273	192.168.56.102	192.168.56.103	DNS	214 Standard query response 0xa696 AAAA se

▶ Frame 3: 183 bytes on wire (1464 bits), 183 bytes captured (1464 bits) on interface 0
 ▶ Ethernet II, Src: PcsCompu_74:ef:ba (08:00:27:74:ef:ba), Dst: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc)
 ▶ Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.103
 ▶ User Datagram Protocol, Src Port: 53, Dst Port: 38569
 ▼ Domain Name System (response)
 Transaction ID: 0x62cc
 Flags: 0x8580 standard query response, No error
 Questions: 1
 Answer RRs: 2
 Authority RRs: 2
 Additional RRs: 2
 ▼ Queries
 ▶ www.nebulanexus1345.com: type A, class IN
 ▼ Answers
 ▶ www.nebulanexus1345.com: type CNAME, class IN, cname nebulanexus1345.com
 ▶ nebulanexus1345.com: type A, class IN, addr 192.168.56.102
 ▼ Authoritative nameservers
 ▶ nebulanexus1345.com: type NS, class IN, ns dns2.nebulanexus1345.com
 ▶ nebulanexus1345.com: type NS, class IN, ns dns1.nebulanexus1345.com
 ▼ Additional records
 ▶ dns1.nebulanexus1345.com: type A, class IN, addr 192.168.56.102
 ▶ dns2.nebulanexus1345.com: type A, class IN, addr 192.168.56.103
 [Request in: 1]
 [Time: 0.000465829 seconds]

Figure 19: Paquete 2 DNS

4 Servicio SMTP/POP

4.1 Descripción del servicio

1. **SMTP (Simple Mail Transfer Protocol)** SMTP es un protocolo de red utilizado para el envío de correos electrónicos entre servidores de correo. Funciona en el nivel de aplicación de la pila de protocolos de Internet y es fundamental para el funcionamiento del correo electrónico en Internet.
2. **POP (Post Office Protocol)** POP es un protocolo de aplicación utilizado para recuperar correos electrónicos de un servidor de correo a un cliente de correo. POP está diseñado para permitir que un cliente de correo descargue los correos electrónicos desde el servidor de correo y los almacene localmente en el dispositivo del usuario.

Cuando un cliente de correo quiere recibir sus correos electrónicos, se conecta al servidor de correo entrante (POP3) y proporciona sus credenciales de acceso. El servidor POP3 verifica las credenciales y permite al cliente descargar los correos electrónicos almacenados en el buzón del usuario.

En resumen, SMTP se utiliza para enviar correos electrónicos entre servidores, mientras que POP se utiliza para descargar correos electrónicos desde un servidor de correo a un cliente de correo. Ambos protocolos son fundamentales para el funcionamiento del correo electrónico en Internet.

4.2 Descripción del escenario

En esta práctica se emplea Exim4. Exim4 es un agente de transporte de correo altamente configurable. Su función principal es enrutar y entregar correos electrónicos entre diferentes servidores de correo. En el desarrollo de esta práctica utilizaremos Dovecot, un servidor de acceso a correo electrónico (POP3/IMAP) diseñado para proporcionar a los usuarios un acceso seguro y eficiente a sus buzones de correo electrónico.

4.3 Configuración

4.3.1 Exim4

Después de comprobar que Exim4 está correctamente instalado, debemos configurarlo para nuestra organización. Habrá que configurarlo de la siguiente manera: Utilizando la configuración rápida 'dpkg-reconfigure exim4-config' introduciremos los datos relativos a nuestra organización.

```
Tipo general del servidor: // Primera opción
(Internet site)
Nombre del sistema de correo: // nebulanexus1345.com
Direcciones IP en las que recibir conexiones SMTP// (en blanco); Cualquier IPs
Destinos de los que se acepta correo: // nebulanexus1345.com
Dominio para que se puede reenviar correo: // nebulanexus1345.com; um.es
Máquinas para las cuales reenviar correo: // 192.168.56.102/24
Limitar consultas DNS: // NO
Formato de buzón de correo: // Maildir
Dividir ficheros de configuración: // NO
```

Para que los cambios tengan efecto habrá que reiniciar el servicio 'service exim4 restart'.

4.3.2 Dovecot

Respecto a la instalación de Dovecot 'apt-get install dovecot-pop3d'. Posteriormente debemos hacer cambios en algunos de los archivos.

- En '/etc/dovecot/conf.d/10-auth.conf': Debemos permitir la autenticación débil basada en texto plano (por defecto está desactivada)

```
disable_plaintext_auth = no
auth_mechanisms = plain // (activar autenticación
débil basada en texto plano)
```

- En '/etc/dovecot/conf.d/10-mail.conf' habrá que especificar el formato de los buzones de correo.

```
mail_location = maildir:~/Maildir
```

4.3.3 Configuración de cliente SMTP/POP

En el equipo cliente hemos de crear dos usuarios en Thunderbird, en nuestro caso estos dos usuarios será alfredo_45@nebulanexus1345.com y edu_13@nebulanexus1345.com y las configuraciones quedarían:

- Datos de configuración del servicio POP necesarios para el cliente

```
Nombre del servidor POP: pop.nebulanexus1345.com
Puerto: 110
Nombre de usuario: usuario1
Seguridad de la conexión (SSL): ninguna
Método de identificación: Contraseña, transmitida de manera insegura
```

- Datos de configuración del servicio SMTP necesarios para el cliente

```
Nombre del servidor: smtp.nebulanexus1345.com
Puerto: 25
Seguridad de la conexión (SSL): ninguna
Método de autenticación: Contraseña, transmitida de manera insegura
Nombre de usuario: alfredo_45@nebulanexus1345.com
```

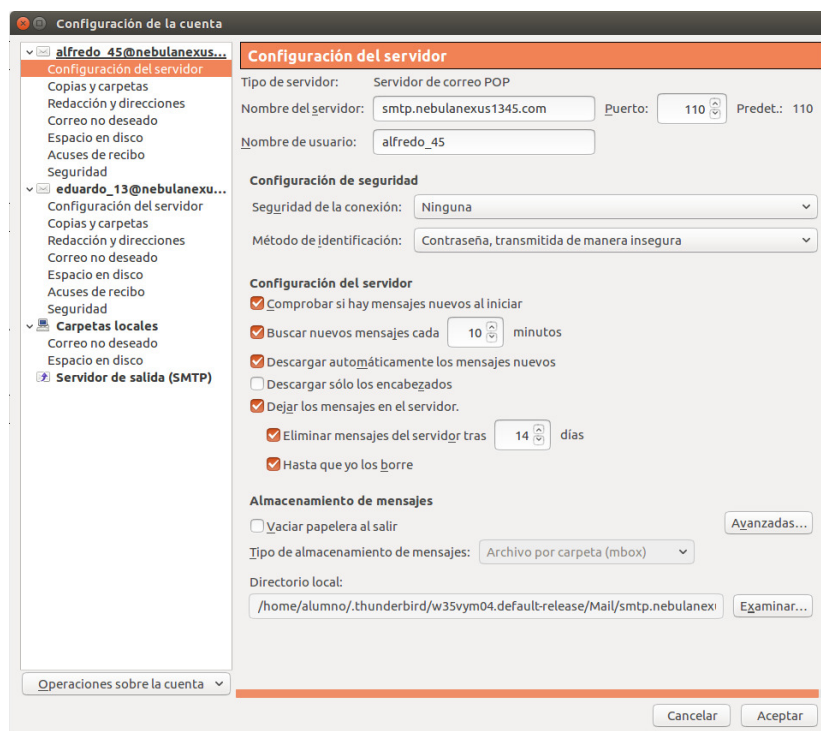


Figure 20: Configuración Servidor en Thunderbird

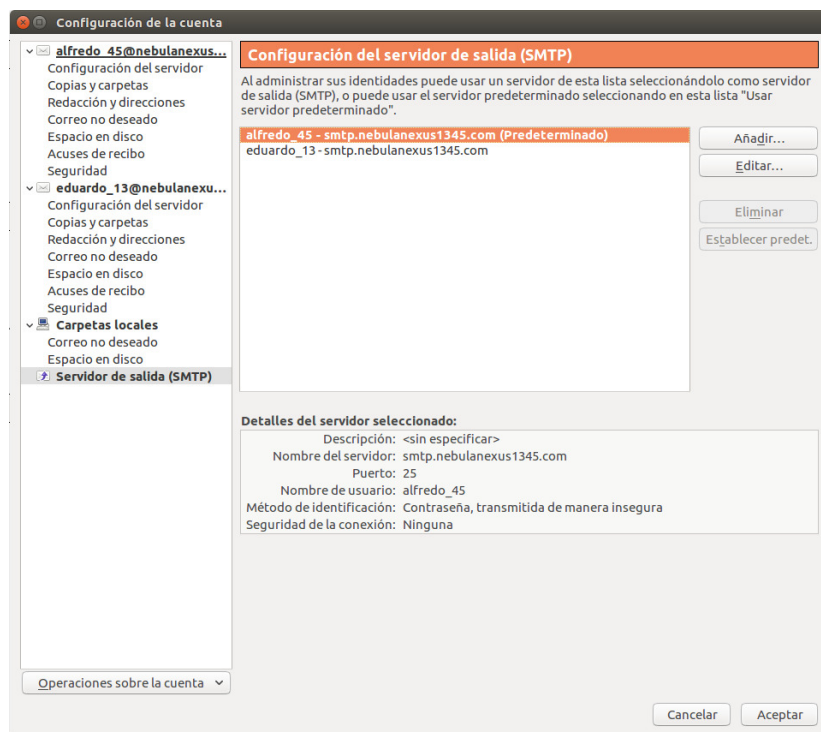


Figure 21: Configuración Servidor de Salida (SMTP)

4.4 Análisis

Los protocolos implementados de SMTP y POP son utilizados por el cliente de servidor ThunderBird. Podemos añadir las cuentas a este y enviar correos entre las cuentas para comprobar que funciona. De esta manera, no podríamos asegurar que tanto SMPT como POP se encuentren instalados correctamente.

4.4.1 Ejecución

Dentro de Thunderbird vamos a crear a enviar un mensaje del correo eduardo.13@nebulanexus1345.com para el correo alfredo.45@nebulanexus1345.com. El contenido del mismo es irrelevante. Al enviar este correo estaremos haciendo uso del protocolo SMTP.

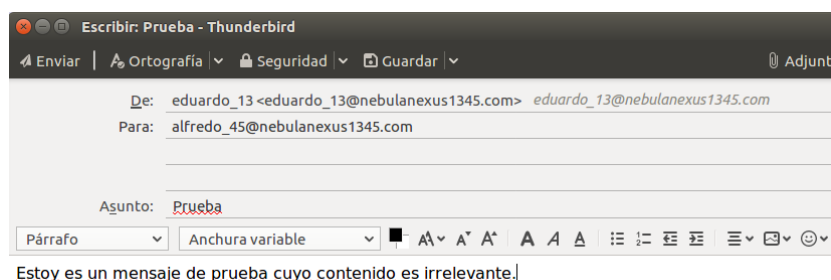


Figure 22: Escribimos un correo para enviar por SMTP

Para recibirlos tenemos que colocarnos en el correo alfredo.45@nebulanexus1345.com y pulsar el botón recibir mensajes situado en la zona izquierda de la barra superior de herramienta. Pulsar este botón provocará que ThunderBird inicie una sesión de POP con el servidor donde consultará los correos presentes en este y recuperará aquellos que nos tenga ya guardados. Para nosotros, el correo recién enviado es el único nuevo y debería aparecernos en la bandeja de entrada con un símbolo de destello para indicar su reciente descarga.

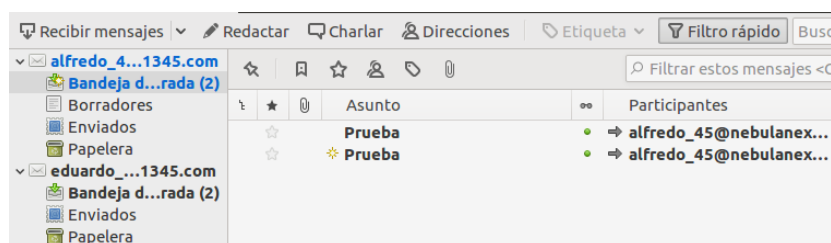


Figure 23: Recibimos el correo por POP

4.4.2 Trazas

Las transacciones realizadas son dos conversaciones diferentes, una de SMTP coloreada en salmón y otra de POP, de color lavanda. Se encuentra toda la estructura del protocolo SMTP: los mensajes 4,6 y 8 pertenecen a la fase del greeting o handshaking esto son los de la información del servidor, el saludo del cliente (client hello) y el saludo respuesta del servidor (server hello); a continuación pasa directamente a la entrega del mensaje

siendo los mensajes del 9 al 14 los que contiene la información básica sobre quién envía el correo y quien lo debe recibir, siendo el número 15 el más importante pues contiene el correo electrónico en su totalidad, esta parte pertenece a la fase de transferencia de los mensajes de correo; y finalmente, la fase de cierre termina en los dos últimos mensajes con el cliente ejecutando el comando QUIT y el servidor confirmandolo.

smtp or pop and not dns					
No.	Time	Source	Destination	Protocol	Length Info
4	0.002035321	192.168.56.102	192.168.56.103	SMTP	135 S: 220 server ESMTP Exim 4
6	0.014181934	192.168.56.103	192.168.56.102	SMTP	89 C: EHLO [192.168.56.103]
8	0.014769984	192.168.56.102	192.168.56.103	SMTP	187 S: 250-server Hello [192.1
9	0.045193177	192.168.56.103	192.168.56.102	SMTP	133 C: MAIL FROM:<eduardo_13@n
10	0.045771446	192.168.56.102	192.168.56.103	SMTP	74 S: 250 OK
11	0.046031936	192.168.56.103	192.168.56.102	SMTP	108 C: RCPT TO:<alfredo_45@neb
12	0.046328264	192.168.56.102	192.168.56.103	SMTP	80 S: 250 Accepted
13	0.046570139	192.168.56.103	192.168.56.102	SMTP	72 C: DATA
14	0.046821583	192.168.56.102	192.168.56.103	SMTP	122 S: 354 Enter message, endi
15	0.048609478	192.168.56.103	192.168.56.102	SMTP L	520 from: eduardo_13 <eduardo_
16	0.054750147	192.168.56.102	192.168.56.103	SMTP	94 S: 250 OK id=1s6BZR-0000jf
17	0.055611619	192.168.56.103	192.168.56.102	SMTP	72 C: QUIT
18	0.055924898	192.168.56.102	192.168.56.103	SMTP	97 S: 221 server closing conn

Figure 24: Interacción de Thunderbird con el servidor SMTP

Por su lado, el protocolo POP también se ejecuta correctamente, aunque no de la misma manera que se ha visto en clase. Durante la primera fase, la de autorización, el cliente usa un comando AUTH (línea 40) , donde también especifica PLAIN para que se sepa que no tiene cifrado. Este comando le permite introducir el usuario y contraseña en una sola cadena (línea 42) con el cifrado especificado. Prosigue la ejecución con la fase de transacción (de la línea 44 a la 53) donde el cliente lista los mensajes presentes y recupera el último recibido, probablemente porque sea el único que no tiene en local, sin borrar ninguno. La línea más importante aquí es la 51 donde el servidor responde con el mensaje. Las dos últimas líneas están dedicadas a la salida del protocolo.

35	69.309763702	192.168.56.102	192.168.56.103	POP	86 S: +OK Dovecot ready.
37	69.310078815	192.168.56.103	192.168.56.102	POP	72 C: CAPA
39	69.310324247	192.168.56.102	192.168.56.103	POP	149 S: +OK
40	69.310784964	192.168.56.103	192.168.56.102	POP	78 C: AUTH PLAIN
41	69.311007142	192.168.56.102	192.168.56.103	POP IMF	70 +
42	69.311081923	192.168.56.103	192.168.56.102	POP	92 C: AGFsZnJlZG9fNDUAYWx1bW5
43	69.312323451	192.168.56.102	192.168.56.103	POP	92 S: +OK Logged in.
44	69.320423346	192.168.56.103	192.168.56.102	POP	72 C: STAT
45	69.320225695	192.168.56.102	192.168.56.103	POP	78 S: +OK 4 3404
46	69.330725338	192.168.56.103	192.168.56.102	POP	72 C: LIST
47	69.331090125	192.168.56.102	192.168.56.103	POP	114 S: +OK 4 messages:
48	69.333180479	192.168.56.103	192.168.56.102	POP	72 C: UIDL
49	69.33325579	192.168.56.102	192.168.56.103	POP	154 S: +OK
50	69.334509160	192.168.56.103	192.168.56.102	POP	74 C: RETR 4
51	69.334845403	192.168.56.102	192.168.56.103	POP	889 S: +OK 804 octets
52	69.367870708	192.168.56.103	192.168.56.102	POP	72 C: QUIT
53	69.370835628	192.168.56.102	192.168.56.103	POP	84 S: +OK Logging out.

Figure 25: Interacción de Thunderbird con el servidor POP

Si damos un vistazo más de cerca a la línea 15 de la conversación de SMTP donde se está enviando el correo, podemos ver todas las cabeceras que Thunderbird ha insertado durante su escritura, así como el mensaje en sí.

15	0.048609478	192.168.56.103	192.168.56.102	SMTP L	520 from: eduardo_13 <eduardo_
16	0.054750147	192.168.56.102	192.168.56.103	SMTP	94 S: 250 OK id=1s6BZR-0000jf
17	0.055611619	192.168.56.103	192.168.56.102	SMTP	72 C: QUIT
18	0.055924898	192.168.56.102	192.168.56.103	SMTP	97 S: 221 server closing conn
▶ Frame 15: 520 bytes on wire (4160 bits), 520 bytes captured (4160 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc), Dst: PcsCompu_74:0f:ba (08:00:27:74:0f:ba) ▶ Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.102 ▶ Transmission Control Protocol, Src Port: 53998, Dst Port: 25, Seq: 139, Ack: 269, Len: 454 ▶ Simple Mail Transfer Protocol ▶ Internet Message Format ▶ To: alfredo_45@nebulanexus1345.com, 1 item ▶ From: eduardo_13 <eduardo_13@nebulanexus1345.com>, 1 item ▶ Item: eduardo_13 <eduardo_13@nebulanexus1345.com>\r\n Subject: Prueba Message-ID: <e1da52d2-2fd3-8aa7-8fff-f1772248838c@nebulanexus1345.com> Date: Sun, 12 May 2024 18:23:57 +0200 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Thunderbird/68.10.0 MIME-Version: 1.0 Content-Type: text/plain; charset=utf-8; format=flowed Content-Transfer-Encoding: 7bit Content-Language: en-US					

Figure 26: Correo siendo enviado por SMTP

De la misma manera, si nos acercamos al packet 51 donde se ha envía la respuesta del servidor conteniendo

el correo entero, podemos verlo entero.

```

51 60 633015402 192.168.56.102 192.168.56.103 POP 880 S: +OK 804 octets
52 69 367879708 192.168.56.103 192.168.56.102 POP 72 S: QUIT
53 69 370835628 192.168.56.102 192.168.56.103 POP 84 S: +OK Logging out.

Frame 51: 880 bytes on wire (7112 bits), 880 bytes captured (7112 bits) on interface 0
Ethernet II, Src: PcsCompu_74:0f:ba (08:00:27:74:0f:ba), Dst: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc)
Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.103
Transmission Control Protocol, Src Port: 110, Dst Port: 59888, Seq: 272, Ack: 71, Len: 823
Post Office Protocol
  +OK 804 octets\r\n
    Response indicator: +OK
    Response description: 804 octets
    Return-path: <eduardo_13@nebulanexus1345.com>\r\n
    Envelope-to: alfredo_45@nebulanexus1345.com\r\n
    Delivery-date: Sun, 12 May 2024 18:23:57 +0200\r\n
    Received: from [192.168.56.103]\r\n
    \tby server with esmtip (Exim 4.86_2)\r\n
    \t(envelope-from <eduardo_13@nebulanexus1345.com>)\r\n
    \tid 1s6BZR-0000jf-TV\r\n
    \tfor alfredo_45@nebulanexus1345.com; Sun, 12 May 2024 18:23:57 +0200\r\n
    To: alfredo_45@nebulanexus1345.com\r\n

```

Figure 27: Correo siendo recibido por POP

Finalmente, nos podemos estar preguntando que si hemos especificado que la autorización sea “plana”, sin cifrado, porque durante el protocolo el cliente ha enviado una cadena de texto ilegible. Esto es así, porque se encuentra codificado en Base64. Con la ayuda de una herramienta como cyberchef, que entre otras cosas puede descifrarla, podemos observar que contiene el usuario y la contraseña del usuario que se está verificando.

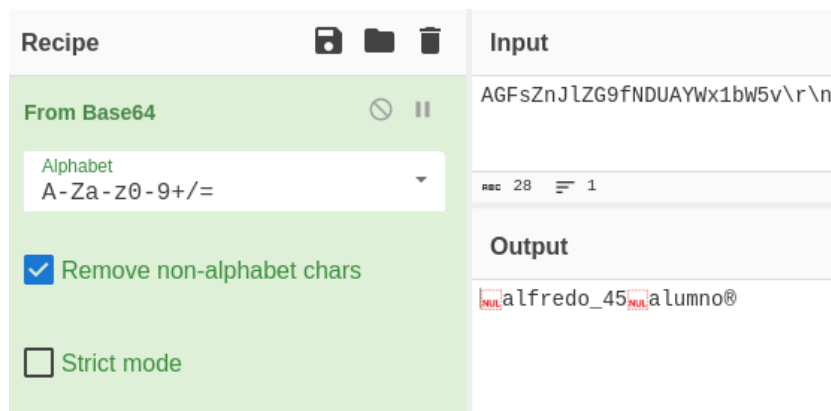


Figure 28: Descifrando el contenido del AUTH con cyberchef

5 Servicios HTTP/HTTPS

5.1 Descripción del servicio

Apache SSL (Secure Sockets Layer) HTTPS es una configuración de Apache que proporciona una capa adicional de seguridad para las comunicaciones entre el servidor web Apache y los clientes, como navegadores web.

1. **Cifrado de datos:** HTTPS utiliza SSL/TLS para cifrar los datos que se transmiten entre el servidor web y el cliente. Esto asegura que los datos estén protegidos contra posibles interceptaciones y garantiza la privacidad y la integridad de la información transmitida.
2. **Certificados SSL/TLS:** Para habilitar HTTPS en Apache, necesitas obtener un certificado SSL/TLS válido de una autoridad de certificación (CA) reconocida. Este certificado se utiliza para autenticar la identidad del servidor web ante los clientes y para establecer una conexión segura mediante el intercambio de claves públicas.
3. **Puerto estándar:** HTTPS utiliza el puerto estándar 443 en lugar del puerto HTTP estándar 80. Cuando un cliente accede a un sitio web a través de HTTPS, el navegador web establece una conexión segura con el servidor utilizando el puerto 443.

5.2 Descripción del escenario

En esta parte de la práctica crearemos nuestra propia PKI (Infraestructura de clave pública) para administrar los certificados necesarios para el resto de servicios de este proyecto.

Cada PKI constará de una Autoridad de Certificación (CA) desplegada en el equipo servidor. Para esta práctica el equipo servidor también realizará las funciones de autoridad de registro (RA).

Trás la generación de los certificados para los servicios los configuramos para hacer uso de ellos y establecer conexiones seguras.

5.3 Configuración

5.3.1 Servidor

Lo primero es preparar el entorno para la PKI. Debemos generar una estructura de archivos para la CA y definir los valores por defecto con los que se nombran las entidades.

1. Creamos el directorio demoCA que contendrá la estructura.
2. Dentro de demoCA debemos crear los siguientes ficheros:
 - clnnumber con valor 00 (números de serie para CLRs).
 - index.txt vacío.
 - serial con valor 1 (número de serie de los certificados)

3. Creamos también tres carpetas: 'private', 'newcerts' y 'certs'

El siguiente paso es modificar el fichero 'openssl.cnf'. Para nuestra organización la sección [CA_default] queda de la siguiente forma:

```
#####
[ CA_default ]

dir               = /home/alumno/demoCA/           # Where everything is kept
certs             = $dir/certs                    # Where the issued certs are kept
crl_dir           = $dir/crl                      # Where the issued crl are kept
database          = $dir/index.txt                # database index file.
#unique_subject   = no                           # Set to 'no' to allow creation of
                                                    # several ctificates with same subject.
new_certs_dir     = $dir/newcerts                  # default place for new certs.

certificate       = $dir/cacert.pem               # The CA certificate
serial            = $dir/serial                    # The current serial number
crlnumber         = $dir/crlnumber                 # the current crl number
                                                    # must be commented out to leave a V1 CRL
crl               = $dir/crl.pem                   # The current CRL
private_key       = $dir/private/cakey.pem         # The private key
```

Figure 29: [CA_default] en openssl.cnf

Esta parte del fichero se refiere a la configuración predeterminada de la Autoridad de Certificación y contiene parámetros que se aplican de manera global a todas las operaciones relacionadas con la CA. Algunos de los parámetros más relevantes son:

- **Directorio de trabajo (dir):** Especifica el directorio de trabajo predeterminado donde OpenSSL buscará archivos de configuración, certificados y claves privadas relacionadas con la CA.
- **Algoritmo de firma (default_days):** Define la duración predeterminada de validez de los certificados emitidos por la CA, en días.
- **Política de uso básico (default_crl_days):** Especifica el número de días después de los cuales una lista de revocación de certificados (CRL) emitida por la CA caducará y deberá ser renovada.

Dentro de este fichero también hemos de editarlo para configurar la estructura de nombre, esto lo haremos en la sección de [req_distinguished_name] que quedará así:

```
[ req_distinguished_name ]
countryName               = Country Name (2 letter code)
countryName_default       = ES
countryName_min           = 2
countryName_max           = 2

stateOrProvinceName       = State or Province Name (full name)
stateOrProvinceName_default = Murcia

LocalityName              = Locality Name (eg, city)

0.organizationName        = Organization Name (eg, company)
0.organizationName_default = UMU

# we can do this but it is not needed normally :-)
#1.organizationName       = Second Organization Name (eg, company)
#1.organizationName_default = World Wide Web Pty Ltd

organizationalUnitName     = Organizational Unit Name (eg, section)
organizationalUnitName_default = NN

commonName                 = Common Name (e.g. server FQDN or YOUR name)
commonName_max             = 64

emailAddress               = Email Address
```

Figure 30: [req_distinguished_name] en openssl.cnf

Y por lo tanto la estructura de nombres para nuestra organización quedaría como:

```
c=ES, st=Murcia, o=UMU, ou=NN
```

Una vez hemos establecido la estructura de nombres para PKI el siguiente paso es generar los certificados correspondientes.

```
alumno@server:~/demoCA$ ls
cacert.pem      clientcert.pfx  crlnumber      index.txt.attr.old  private        servercert.pem
certs          clientcsr.pem   index.txt      index.txt.old       serial         servercsr.pem
clientcert.pem  clientkey.pem   index.txt.attr newcerts           serial.old     serverkey.pem
```

Figure 31: Directorio /home/alumno/demoCA

- 'cakey.pem': contiene la clave privada de la CA, protegida con la palabra de paso (hay que moverla a 'demoCA/private' una vez generada).
- 'cacert.pem' contiene el certificado X.509 de la CA (que incluye la clave pública).

Con el siguiente comando generamos los dos ficheros:

```
cd /home/alumno/demoCA/ openssl req -x509 -newkey rsa:2048 -keyout cakey.pem \
-out cacert.pem -days 3650
```

Ahora hemos de certificar nuestro servicio web. Para generar un certificado para un servidor web el administrador del servicio debe personarse en una Autoridad de Registro (RA) y emitir un solicitud de certificación. Para esta práctica simularemos la RA mediante comandos OpenSSL (la RA está situada en nuestro equipo servidor).

- 'serverkey.pem': contiene la clave privada RSA para el servicio web.
- 'servercsr.pem': contiene una solicitud de certificación en formato PKCS#10.

Con el siguiente comando generamos los dos ficheros:

```
cd /home/alumno/demoCA/ openssl req -new -nodes -newkey rsa:2048  
-keyout serverkey.pem -out servercsr.pem
```

La solicitud de certificación ha de ser enviada a la CA para que la firme y genere el certificado X.509. Como en nuestro caso el Administrador esta en el mismo equipo la solicitud está ya disponible para la CA.

Por último la CA debe firmar la solicitud para generar el certificado X.509:

```
cd /home/alumno/demoCA/ openssl req -new -nodes -newkey rsa:2048  
-keyout serverkey.pem -out servercsr.pem
```

Este comando genera el certificado 'servercert.pem' y se almacena en /home/alumno/demoCA/newcerts/<serial_number>.pem.

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 3 (0x3)
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=ES, ST=Murcia, O=UMU, OU=NN, CN=ca.nebulanexus1345.com/emailAddress=edu@nebulanex$
  Validity
    Not Before: Apr 29 15:58:49 2024 GMT
    Not After : Jun  3 15:58:49 2025 GMT
  Subject: C=ES, ST=Murcia, O=UMU, OU=NN, CN=www.nebulanexus1345.com
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    Modulus:
      00:ec:01:1c:76:4c:94:4d:4e:7d:a8:48:84:b4:26:
      6f:2d:43:87:2a:12:a0:f6:24:f8:89:61:19:d7:9f:
      ea:23:08:9f:ca:5b:d9:ad:76:02:94:86:3c:ef:dd:
      5d:41:a5:ef:3c:8c:b0:89:6e:04:e5:18:fa:56:6c:
      a5:d8:78:60:95:1b:32:76:19:aa:86:67:35:1f:91:
      8f:3c:89:6f:ca:e8:59:d3:bb:9c:39:ff:08:f3:ad:
      ac:59:57:97:f0:d5:64:ad:8d:d6:2d:bf:54:ff:60:
      b6:20:8d:a1:00:23:34:c0:f6:62:5a:d2:b4:39:8d:
      15:1f:1b:7e:e4:8e:55:28:cc:8b:ac:26:79:11:f3:
      da:44:33:43:31:ac:46:9d:87:ef:b6:26:99:82:6d:
      63:15:17:21:9f:39:6f:31:a6:ec:ee:18:b0:9b:a6:
      b0:ee:0d:0e:ad:f3:5c:44:c2:78:32:7a:e3:71:00:
      a2:fa:fe:39:7b:a7:5b:fe:b5:ee:f2:ab:20:07:ae:
      8c:c0:33:02:cd:cf:97:8e:ff:41:4f:51:35:1b:76:
      61:89:32:d6:86:56:11:01:b0:55:25:b8:4b:e2:d6:
```

Figure 32: Contenido servercert.pem

Por último ya solo queda instalar el material criptográfico en el servidor web Apache. Para lo cual debemos habilitar el puerto 443 en 'etc/apache2/ports.conf'.

```
Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

Figure 33: Puerto 443

En '/etc/apache2/sites-enabled/nebulanexus.conf' añadiremos la configuración relacionada con SSL.

```
alumno@server:/etc/apache2/sites-available$ cat nebulaNexus.conf
<VirtualHost *:443>
    ServerAdmin edu@nebulanexus1345.com
    ServerName nebulanexus1345.com
    DocumentRoot /var/www/mi_web
    <Directory /var/www/mi_web>
        AllowOverride AuthConfig
        AuthType Basic
        AuthName "Acceso restringido a trabajadores"
        AuthBasicProvider file
        AuthUserFile /etc/apache2/passwords
        AuthGroupFile /etc/apache2/groups
        Require group trabajadores
        Order allow,deny
        allow from all
    </Directory>
    SSLEngine on
    SSLCertificateFile /home/alumno/demoCA/servercert.pem
    SSLCertificateKeyFile /home/alumno/demoCA/serverkey.pem
    SSLCACertificateFile /home/alumno/demoCA/cacert.pem
    SSLVerifyClient require
    SSLVerifyDepth 10
</VirtualHost>
```

Figure 34: SSL en nebulanexus.conf

Las dos últimas directivas son necesarias para especificar los requisitos de autenticación del cliente. El servidor Apache solo permitirá la conexión si el cliente proporciona un certificado válido durante el proceso de handshake SSL/TLS y el servidor Apache verificará hasta 10 certificados de la cadena de certificados del cliente para asegurarse de que todos son válidos y de confianza antes de aceptar la conexión.

5.3.2 Cliente

Al igual que para el servidor seguimos los mismos pasos para generar el certificado para el cliente.

```
cd /home/alumno/demoCA/
openssl req -new -nodes -newkey rsa:2048 -keyout clientkey.pem \
-out clientcsr.pem
openssl ca -keyfile private/cakey.pem -in clientcsr.pem \
```

```
-out clientcert.pem {days 400
```

Los navegadores esperan esta información en unico archivo en formato PKCS#12 (PFX). Para generar este archivo utlizamos el comando (en el servidor):

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

El archivo generado contiene el certificado X.509 del cliente, la clave privada y el certificado X.509 de CA. Por último movemos el fichero .pfx al equipo cliente y lo importamos en el navegador.

5.4 Análisis

El protocolo TLS se ha añadido a nuestro servidor para poder añadir un acceso mediante HTTPS a nuestro sitio web. Si recordamos la configuración de nuestro servidor HTTP mostrada anteriormente, este ya estaba preparado para conectarnos mediante HTTPS a él y lo único que nos ha hecho falta ha sido obtener el certificado del cliente e importarlo en nuestro navegador Firefox. Por ello, para probarlo tendremos que acceder a nuestra página web con HTTPS en vez de con HTTP como ya hemos hecho anteriormente.

5.4.1 Ejecución

Conforme nos conectamos a la página, nos saltará una ventana emergente que nos pedirá que seleccionemos un certificado. Como solo tenemos el que hemos obtenido del servidor, estará ya seleccionado en la combo-box subrayada.

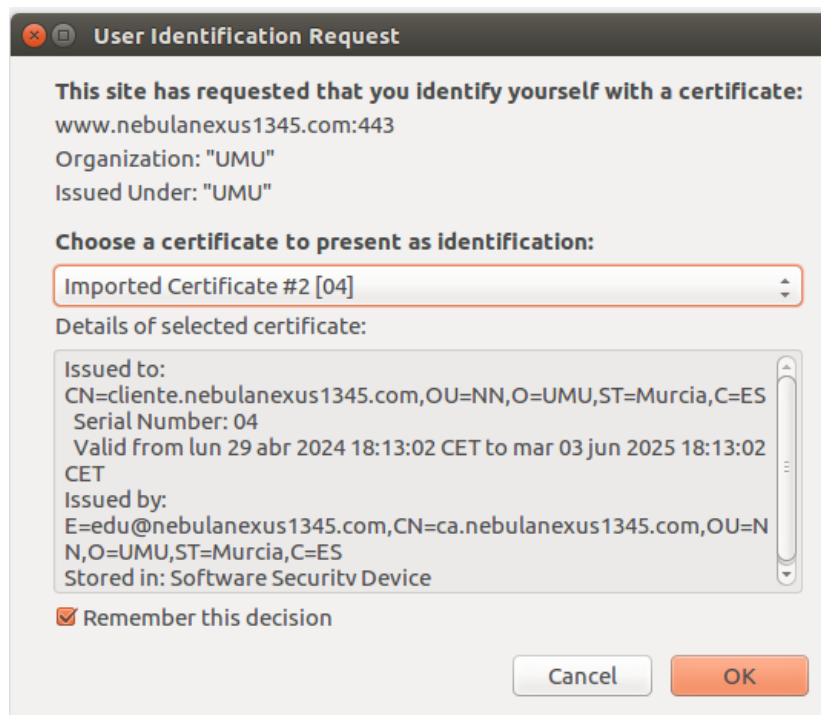


Figure 35: Utilizamos el certificado generado por el servidor

Una vez entramos, sabremos que nos encontramos en HTTPS porque aparecerá en la barra de búsqueda. Si no podemos verla porque, por alguna razón, el navegador la oculta, podremos verificarlo por el símbolo verde de un candado al lado de la URL.



Figure 36: El candado verde y la URL confirman que nos encontramos en HTTPS

5.4.2 Trazas

Las trazas generadas no solo utiliza TLS, sino que también contiene una gran cantidad de mensajes TCP, ya que se utilizan de la misma manera que en HTTP sin cifrar, pero no son de suficiente nivel en la pila TCP/IP como para ser cifrados por este protocolo. Por norma general, en un intercambio el iniciador envía TCP ACK tras terminar el intercambio para indicar que ha recibido correctamente la información pedida.

No.	Time	Source	Destination	Protocol	Length	Info
6	0.000000000	192.168.56.183	192.168.56.103	TCP	74	40324 → 40324 [SYN, ACK] Seq=654321234 Win=0 Len=0 MSS=1460 SACK_PERM=1 TSval=3388123 TSecr=40380 WS=138
7	0.000000000	192.168.56.183	192.168.56.103	TCP	66	40324 → 443 [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=40380 TSecr=3388123
8	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	260	Client Hello
9	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [ACK] Seq=1 Ack=188 Win=0 Len=0 TSval=3388123 TSecr=40380
10	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	2060	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
11	0.000000000	192.168.56.183	192.168.56.103	TCP	66	40324 → 443 [ACK] Seq=188 Ack=2587 Win=0 Len=0 TSval=3388123 TSecr=3388123
12	0.000000000	192.168.56.183	192.168.56.103	TCP	66	[TCP Keep-Alive] 40324 → 443 [ACK] Seq=188 Ack=2587 Win=0 Len=0 TSval=3388123 TSecr=3388123
13	0.000000000	192.168.56.183	192.168.56.103	TCP	66	[TCP Keep-Alive] 443 → 40324 [ACK] Seq=2587 Ack=188 Win=0 Len=0 TSval=3388123 TSecr=40380
14	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	1454	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
15	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	480	Application Data
16	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [ACK] Seq=2587 Ack=188 Win=0 Len=0 TSval=3388123 TSecr=40380
17	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	1340	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
18	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	66	40324 → 443 [ACK] Seq=188 Ack=3879 Win=0 Len=0 TSval=44664 TSecr=3388123
19	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	882	Application Data
20	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	66	40324 → 443 [ACK] Seq=188 Ack=4885 Win=0 Len=0 TSval=44664 TSecr=3388123
21	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	782	Application Data
22	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	66	40324 → 443 [ACK] Seq=2289 Ack=5411 Win=0 Len=0 TSval=44664 TSecr=3388123
23	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	8989	Application Data
24	0.000000000	192.168.56.183	192.168.56.103	TCP	66	40324 → 443 [ACK] Seq=2289 Ack=5411 Win=0 Len=0 TSval=44664 TSecr=3388123
25	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	626	Application Data
26	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	410	Application Data
27	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	626	Application Data
28	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	66	40324 → 443 [ACK] Seq=3832 Ack=15874 Win=0 Len=0 TSval=45856 TSecr=3388123
29	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	440	Application Data
30	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	626	Application Data
31	0.000000000	192.168.56.183	192.168.56.103	TLSv1.2	97	Encrypted Alert
32	0.000000000	192.168.56.183	192.168.56.103	TCP	66	40324 → 443 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
33	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
34	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
35	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
36	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
37	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
38	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
39	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
40	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
41	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
42	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
43	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
44	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
45	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
46	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
47	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
48	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
49	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
50	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
51	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123
52	0.000000000	192.168.56.183	192.168.56.103	TCP	66	443 → 40324 [FIN, ACK] Seq=16460 Ack=3412 Win=0 Len=0 TSval=3388123 TSecr=3388123

Figure 37: Interacción completa de TLS y TCP

Solo vamos a analizar la traza del TLS handshake, pues el resto del código se encuentra cifrado por las claves determinadas en este handshake o es TCP. Si nos fijamos, aunque el protocolo requiere de una gran cantidad de mensajes, solo se han pasado cuatro packets durante este. Esto ocurre porque, como optimización, en la práctica se juntan todos los mensajes que un equipo va a enviar seguidos en uno solo. Es por esto que, por ejemplo, el ServerHello, Certificate y ServerHelloDone se encuentran dentro de un solo packet. Aquí abajo podemos ver un diagrama que representa cómo es la interacción.

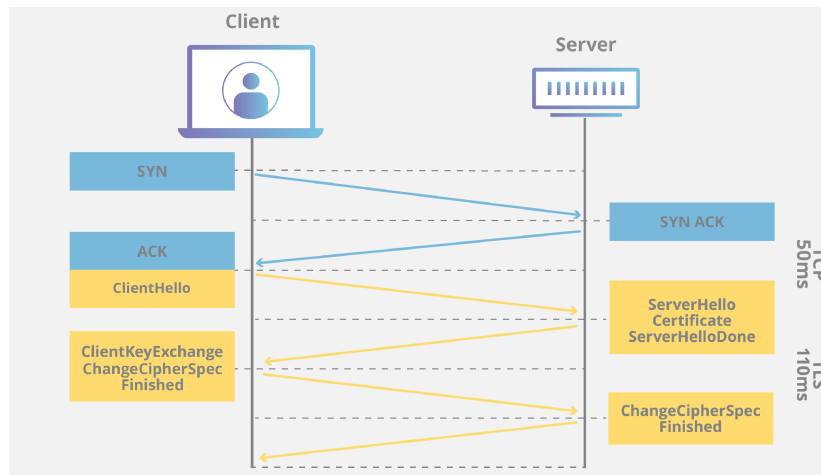


Figure 38: source: cloudflare.com

La primera traza que vamos a ver es la de del Client Hello

8	0.001076624	192.168.56.103	192.168.56.102	TLSv1.2	263 Client Hello
10	0.003360461	192.168.56.102	192.168.56.103	TLSv1.2	2662 Server Hello
22	18.254560243	192.168.56.103	192.168.56.102	TLSv1.2	1454 Certificate
23	18.254783664	192.168.56.103	192.168.56.102	TLSv1.2	400 Application I
25	18.255882411	192.168.56.102	192.168.56.103	TLSv1.2	1348 New Session
27	18.256201460	192.168.56.102	192.168.56.103	TLSv1.2	882 Application I
29	22.903596665	192.168.56.103	192.168.56.102	TLSv1.2	435 Application I
30	22.904729953	192.168.56.102	192.168.56.103	TLSv1.2	782 Annlication I

```

▶ Frame 8: 263 bytes on wire (2104 bits), 263 bytes captured (2104 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc), Dst: PcsCompu_74:0f:ba (08:00:27:74:0f:ba)
▶ Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.102
▶ Transmission Control Protocol, Src Port: 49324, Dst Port: 443, Seq: 1, Ack: 1, Len: 197
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 192
    ▼ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 188
      Version: TLS 1.2 (0x0303)
      ▶ Random: 3d1d47b419d9a8a61e068bad6eb7059297c6892bd710107...
      Session ID Length: 0
      Cipher Suites Length: 22
      Cipher Suites (11 suites)
      Compression Methods Length: 1
      Compression Methods (1 method)
      Extensions Length: 125
      ▶ Extension: server_name (len=28)
      ▶ Extension: renegotiation_info (len=1)
      ▶ Extension: supported_groups (len=8)
      ▶ Extension: ec_point_formats (len=2)
      ▶ Extension: SessionTicket TLS (len=0)
      ▶ Extension: next_protocol_negotiation (len=0)
      ▶ Extension: application_layer_protocol_negotiation (len=23)
      ▶ Extension: status_request (len=5)
      ▶ Extension: signature_algorithms (len=22)

```

Figure 39: Primer mensaje del handshake

A continuación, el servidor responde con un packet que contiene tres mensajes, uno de los cuales contiene dos a su vez, haciendo un total de cuatro. Estos mensajes son: el certificado del servidor, el intercambio de claves, la petición del certificado del cliente y la finalización de la fase de saludo.

10	0.003360461	192.168.56.102	192.168.56.103	TLSv1.2	2662 Server Hello
22	18.254560243	192.168.56.103	192.168.56.102	TLSv1.2	1454 Certificate
23	18.254783664	192.168.56.103	192.168.56.102	TLSv1.2	400 Application I
25	18.255882411	192.168.56.102	192.168.56.103	TLSv1.2	1348 New Session
27	18.256201460	192.168.56.102	192.168.56.103	TLSv1.2	882 Application I
29	22.903596665	192.168.56.103	192.168.56.102	TLSv1.2	435 Application I
30	22.904729953	192.168.56.102	192.168.56.103	TLSv1.2	782 Annlication I

```

▶ Frame 10: 2662 bytes on wire (21296 bits), 2662 bytes captured (21296 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_74:0f:ba (08:00:27:74:0f:ba), Dst: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc)
▶ Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.103
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 49324, Seq: 1, Ack: 198, Len: 2596
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 76
    ▶ Handshake Protocol: Server Hello
    ▶ TLSv1.2 Record Layer: Handshake Protocol: Certificate
    ▶ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
    ▼ TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 181
      ▶ Handshake Protocol: Certificate Request
      ▶ Handshake Protocol: Server Hello Done

```

Figure 40: Segundo paquete del handshake

Este tercer paquete contiene también vario mensajes, concretamente: el certificado del cliente, el intercambio de claves del cliente y la verificación del certificado. Además, el siguiente mensajes que envía el cliente ya va cifrado, sin ni siquiera esperar el final del propio handshake, pues ya posee todas la información necesario para calcular la clave simétrica.

No.	Time	Source	Destination	Protocol	Length	Info
8	0.001076624	192.168.56.103	192.168.56.102	TLSv1.2	263	Client Hello
10	0.003360461	192.168.56.102	192.168.56.103	TLSv1.2	2662	Server Hello, Certificate, Server
22	18.254500243	192.168.56.103	192.168.56.102	TLSv1.2	1454	Certificate, Client Key Exchange
23	18.254783664	192.168.56.103	192.168.56.102	TLSv1.2	400	Application Data
25	18.255882411	192.168.56.102	192.168.56.103	TLSv1.2	1348	New Session Ticket, Change Cipher
27	18.256201460	192.168.56.102	192.168.56.103	TLSv1.2	882	Application Data, Application Dat
29	22.903596665	192.168.56.103	192.168.56.102	TLSv1.2	435	Application Data
30	22.904729053	192.168.56.102	192.168.56.103	TLSv1.2	782	Application Data, Application Dat
▶ Frame 22: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc), Dst: PcsCompu_74:0f:ba (08:00:27:74:0f:ba) ▶ Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.102 ▶ Transmission Control Protocol, Src Port: 49324, Dst Port: 443, Seq: 198, Ack: 2597, Len: 1388 ▶ Secure Sockets Layer						
▼ TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 1332 ▶ Handshake Protocol: Certificate ▶ Handshake Protocol: Client Key Exchange ▶ Handshake Protocol: Certificate Verify ▶ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec ▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 40 Handshake Protocol: Encrypted Handshake Message						

Figure 41: Tercer paquete del handshake

Finalmente, el servidor devuelve al servidor un paquete con tres mensajes: un nuevo ticket de sesión, el mensaje de confirmación de cambio a mensajes cifrados y un mensaje cifrado que, como sigue el estándar, sabemos que el mensaje de confirmación de fin del handshake. A partir, de recibir el mensaje anterior del cliente, el servidor ya puede empezar a responder las peticiones que le llegaron incluso antes de terminar el handshake.

No.	Time	Source	Destination	Protocol	Length	Info
8	0.001076624	192.168.56.103	192.168.56.102	TLSv1.2	263	Client Hello
10	0.003360461	192.168.56.102	192.168.56.103	TLSv1.2	2662	Server Hello, Certificate, Server
22	18.254500243	192.168.56.103	192.168.56.102	TLSv1.2	1454	Certificate, Client Key Exchange
23	18.254783664	192.168.56.103	192.168.56.102	TLSv1.2	400	Application Data
25	18.255882411	192.168.56.102	192.168.56.103	TLSv1.2	1348	New Session Ticket, Change Cipher
27	18.256201460	192.168.56.102	192.168.56.103	TLSv1.2	882	Application Data, Application Dat
29	22.903596665	192.168.56.103	192.168.56.102	TLSv1.2	435	Application Data
30	22.904729053	192.168.56.102	192.168.56.103	TLSv1.2	782	Application Data, Application Dat
▶ Frame 22: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_a6:e0:bc (08:00:27:a6:e0:bc), Dst: PcsCompu_74:0f:ba (08:00:27:74:0f:ba) ▶ Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.102 ▶ Transmission Control Protocol, Src Port: 49324, Dst Port: 443, Seq: 198, Ack: 2597, Len: 1388 ▶ Secure Sockets Layer						
▼ TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 1332 ▶ Handshake Protocol: Certificate ▶ Handshake Protocol: Client Key Exchange ▶ Handshake Protocol: Certificate Verify ▶ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec ▼ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 40 Handshake Protocol: Encrypted Handshake Message						

Figure 42: Último paquete del handshake

6 IPsec

6.1 Descripción del servicio

IPsec es una tecnología fundamental para garantizar la seguridad de las comunicaciones a nivel de red en Internet, proporcionando autenticación, integridad y confidencialidad de los datos transmitidos a través de la capa de Internet.

Para el desarrollo de esta práctica haremos uso de Strongswan una implementación OpenSource de IPSec, lleva una implementación de IKEv2.

6.2 Descripción del escenario

Para nuestra organización hemos configurado una sencilla implementación de IPsec con claves simétricas previamente compartidas para proteger todo el tráfico que es enviado entre dos equipos. Para esto utilizamos la herramienta strongswan que descargamos en ambos equipos, servidor y cliente.

6.3 Configuración

Respecto a los comandos utilizados para strongswan:

- `ipsec start`: arranca strongswan.
- `ipsec stop`: para el demonio strongswan.
- `ipsec restart`: reinicia strongswan.

6.3.1 Ficheros de configuración

1. `/etc/ipsec.conf` Fichero de configuración de strongswan. Aquí se definen las características de la asociación de seguridad IPsec (AH, ESP, túnel o transporte).

```
#Para el servicio
config setup
#Para cualquier conexión
conn %default
    ikelifetime=60m # Tiempo de vida de una IKE SA
    keylife=20m # Tiempo de vida de una Ipsec sa
    rekeymargin=3m
    keyingtries=1
    mobike=no
    keyexchange=ikev2 #Usamos IKEv2
    authby=pubkey #Autenticación con PSK
    esp=null-sha1_160
#Para la conexión específica entre estos dos hosts
conn host-host # "host-host" es simplemente una etiqueta
    left=192.168.56.102 #La IP de un equipo
    leftcert=/etc/ipsec.d/certs/servercert.pem
    leftid="C=ES, ST=Murcia, O=UMU, OU=NN, CN=www.nebulanexus1345.com"
    right=192.168.56.103 #La IP de otro equipo
    rightid="C=ES, ST=Murcia, O=UMU, OU=NN, CN=cliente.nebulanexus1345.com"
    type=transport # modo transporte
    auto=start # IKEv2 se ejecuta en el momento que haya un ipsec start
```

Figure 43: Fichero /etc/ipsec.conf

- **ikelifetime:** Establece el tiempo de vida de una Asociación de Seguridad de Intercambio de Claves de Internet (IKE SA).
- **keylife:** Establece el tiempo de vida de una Asociación de Seguridad de Protocolo IPsec (IPsec SA).
- **rekeymargin:** Define el tiempo restante antes de que una SA expire, cuando se inicia el proceso de rekeying.
- **keyingtries:** Especifica el número de intentos de rekeying antes de fallar.
- **mobike:** Indica si la movilidad (Mobility) debe ser habilitada o no.
- **keyexchange:** Especifica el protocolo de intercambio de claves a utilizar, en este caso, IKEv2.
- **authby:** Especifica el método de autenticación a utilizar, en este caso, autenticación con
- **esp:** Define el algoritmo de cifrado y hash para las Asociaciones de Seguridad de Protocolo de Encapsulación de Seguridad (ESP). En este caso cofrado nulo e integridad.
- **left:** La dirección IP del host local.
- **right:** La dirección IP del host remoto.
- **type:** Especifica el modo de conexión, en este caso, modo transporte, que se utiliza para la protección de datos a nivel de transporte.
- **auto:** Indica si la conexión se iniciará automáticamente cuando se ejecute el comando ipsec start.

2. **/etc/ipsec.secrets:** contiene información confidencial y claves de seguridad para el servicio IPsec.

```
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.
192.168.56.102 192.168.56.103 : RSA /etc/ipsec.d/private/serverkey.pem
```

Figure 44: Fichero /etc/ipsec.secrets

- **192.60.56.102 y 192.160.56.103:** Esta línea especifica las direcciones IP de los hosts entre los cuales se aplicará esta configuración de seguridad.
- **RSA /etc/ipsec.d/private/serverkey.pem:** Esta línea contiene la información de autenticación y encriptación para la conexión entre los dos hosts.

6.3.2 Uso de certificados X.509

Utilizamos los certificados de cliente y servidor descritos anteriormente. En el fichero 'ipsec.conf' habrá que en la directriz 'authby' cambiar el contenido a:

```
conn %default
... #configuración ejemplo anterior pero cambia...
authby=pubkey
```

Los directorios que utilizamos para guardar los certificados serán:

- /etc/ipsec.d/certs/ //Certificado de esta entidad IKE
- /etc/ipsec.d/private/ //Clave privada
- /etc/ipsec.d/cacert/ //Certificado de la CA

Y por último añadiremos al fichero 'ipsec.secrets' la directriz:

```
: RSA /etc/ipsec.d/private/clientkey.pem
```

6.4 Análisis

Al igual que TLS, IPsec se trata de un protocolo que se encarga de cifrar el tráfico (salvo el nuestro porque lo hemos puesto en nulo durante la configuración) por lo que no se debería poder leer el contenido de este. Por esta razón, vamos a analizar principalmente el contenido del handshake que se encuentra en texto plano y a continuación, continuaremos con unos cuantos ejemplos de mensajes de protocolos anteriores mientras se encuentran en paquetes ESP propios de IPsec. Para iniciar la conexión en IPsec ambos dispositivos tienen que tener StrongSwan funcionando. Para ellos podemos escribir el siguiente comando:

```
sudo ipsec start
```

Si el programa ya se encuentra funcionando no habrá ningún problema.

6.4.1 Trazas

Al igual que TLS, IPsec se trata de un protocolo que se encarga de cifrar el tráfico (salvo el nuestro porque lo hemos puesto en nulo durante la configuración) por lo que no se debería poder leer el contenido de este. Por esta razón, vamos a analizar principalmente el contenido del handshake que se encuentra en texto plano y, a continuación, seguiremos con unos cuantos ejemplos de mensajes de protocolos anteriores mientras se encuentran en paquetes ESP propios de IPsec.

No.	Time	Source	Destination	Protocol	Length	Info
28	2.139954658	192.168.56.103	192.168.56.102	ISAKMP	1168	IKE_SA_INIT MID=00 Initiator Request
29	2.148327424	192.168.56.102	192.168.56.103	ISAKMP	525	IKE_SA_INIT MID=00 Responder Response
31	2.153793791	192.168.56.103	192.168.56.102	ISAKMP	352	IKE_AUTH MID=01 Initiator Request
33	2.158279692	192.168.56.102	192.168.56.103	ISAKMP	112	IKE_AUTH MID=01 Responder Response
38	2.165797577	192.168.56.102	192.168.56.103	ESP	116	ESP (SPI=0xc12becdc)

Figure 45: Handshake de IKE completo

En la siguiente traza podemos ver, aunque no se vea muy bien, que las partes que en la parte que se encuentra centrada y por lo tanto se puede traducir a ASCII, que se menciona el nombre del dominio de la web, además de mencionar dns1 y dns2. Estos términos nos recuerdan al fichero db del servicio DNS que especificaba cuáles eran los registro SOA. Es por esto, que deducimos que los más probable es que este paquete se trate de la respuesta del servidor DNS que contiene, entre otras cosas las direcciones de los servidores dns.

52	73.671668805	192.168.56.102	192.168.56.103	ESP	216	ESP (SPI=0xc12becdc)
53	73.671697432	192.168.56.102	192.168.56.103	ESP	172	ESP (SPI=0xc12becdc)
54	73.671942987	192.168.56.102	192.168.56.103	ESP	856	ESP (SPI=0xc12becdc)
55	73.672006634	192.168.56.102	192.168.56.103	ESP	400	ESP (SPI=0xc12becdc)

Frame 52: 216 bytes on wire (1728 bits)	0000	00 00 00 01 00 06 08 00	27 74 0f ba 00 00 00 00t.....
Linux cooked capture v1	0010	45 00 00 c8 2b 6d 00 00	40 32 5c 79 c0 a8 38 66	E...+m...02y..8f
Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.103	0020	c0 a8 38 67 c1 2b ec dc	00 00 00 06 00 35 90 9c	..8g+...5...
Encapsulating Security Payload	0030	00 95 dc 11 5c f0 85 80	00 01 00 02 00 02 00 02+.....
ESP SPI: 0xc12becdc (324088137)	0040	03 77 77 0f 6e 65 62	75 6c 61 6a 65 78 75 73	..www.neb..ulanexu
ESP Sequence: 6	0050	11 33 34 35 03 63 6f 6d	00 00 01 00 01 c0 0c 00	..1345.com.....
	0060	05 00 01 00 00 3a 89 00	02 c0 10 c0 10 00 01 00:.....
	0070	01 00 09 3a 80 00 04 c0	a8 38 66 c0 10 00 02 00:8f.....
	0080	01 00 09 3a 80 00 07 04	64 6e 73 31 c0 10 c0 10:dns1.....
	0090	00 02 00 01 00 09 3a 80	00 07 04 64 6e 73 32 c0:dns2.....
	00a0	10 c0 53 00 01 00 01 00	09 3a 80 00 04 c0 a8 38	..:S.....:8
	00b0	66 c0 66 00 01 00 01 00	09 3a 80 00 04 c0 a8 38	..f.f.....:8
	00c0	07 01 01 11 e0 88 f2 82	38 dc 86 47 61 96 00 00	..g.....8-Ga.k
	00d0	ac bb 5e 25 bd db 10 a0		..%.....

Figure 46: El paquete ESP contiene un paquete de DNS

En la traza de la Figura 47 podemos ver mencionado “GET /HTTP/1.1”, así como varias palabras reconocibles tales como “Ubuntu”, “Mozilla”, etc. Esto es porque probablemente sea un paquete de petición del index.html y estas los términos que siguen al GET sean las cabeceras.

49	73.671110845	192.168.56.103	192.168.56.102	ESP	516	ESP (SPI=0xc997e4b)
50	73.671133590	192.168.56.103	192.168.56.102	ESP	116	ESP (SPI=0xc997e4b)
51	73.671137750	192.168.56.103	192.168.56.102	ESP	116	ESP (SPI=0xc997e4b)

Frame 49: 516 bytes on wire (4128 bits)	0000	00 04 00 01 00 06 08 00	27 a5 e0 bc 00 00 00 00@.....g
Linux cooked capture v1	0010	45 00 01 f4 2f db 40 00	40 32 16 df c0 a8 38 67	E.../...02...8g
Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.102	0020	c0 a8 38 66 ce 99 7e 4b	00 00 00 07 91 56 00 50	..8f...-K.....V-P
Encapsulating Security Payload	0030	12 49 e8 ee ed 06 be 2b	80 18 00 f0 38 a7 00 00	..I.....+.....8...
ESP SPI: 0xc997e4b (3466165835)	0040	01 01 08 0a 00 09 54 14	00 3b 2c 76 47 45 54 20T...;vGET
ESP Sequence: 7	0050	2f 20 48 54 54 50 2f 31	2e 31 00 0a 48 6f 73 74	.. / HTTP/1..i-Host
	0060	3a 20 77 77 77 2e 6e 65	62 75 6c 61 6e 65 78 75	..: www.neb..ulanexu
	0070	73 31 33 34 35 2e 63 6f	6d 0d 0a 55 73 65 72 d2	..s1345.co m-User-
	0080	41 67 65 6e 74 3a 20 4d	6f 7a 69 6c 6c 61 2f 35	..Agent: M ozilla/5
	0090	2e 30 20 28 58 31 31 3b	20 55 62 75 6e 74 75 3b	..0 (X11; Ubuntu;
	00a0	20 4c 69 6e 75 78 20 78	38 36 5f 36 3a 3b 20 72	..Linux x 86_64; r
	00b0	76 3a 34 35 2e 30 29 20	47 65 63 6b 6f 2f 32 30	..v:45.0) Gecko/20
	00c0	31 30 30 31 30 31 20 46	69 72 65 66 6f 78 2f 34	..199191 F irefox/4
	00d0	35 2e 30 0d 0a 41 63 63	65 70 74 3a 20 74 65 78	..5.0 -Acc ept: tex
	00e0	74 2f 68 74 6d 6c 2c 61	70 79 6c 69 63 61 74 69	..t/html,a pplicati
	00f0	6f 6e 2f 78 68 74 6d 6c	2b 78 6d 6c 2c 61 70 70	..on/xhtml+xml,app
	0100	6c 69 63 61 74 69 6f 6e	2f 78 6d 6c 3b 71 3d 30	..lication /xml;q=0
	0110	2e 39 2c 2a 2f 2a 3b 71	3d 30 2e 38 0d 0a 41 63	..9,/*;q =0.8 -Ac
	0120	63 65 70 74 2d 4c 61 6e	67 75 61 67 65 3a 20 65	..cept-Lan guage: e
	0130	6e 2d 55 53 2c 65 6e 3b	71 3d 30 2e 35 0d 0a 41	..n-US,en;q=0.5 -A
	0140	63 63 65 70 74 2d 45 6e	63 6f 64 69 6e 67 3a 20	..ccept-En coding:
	0150	67 7a 69 70 2c 20 64 65	66 6c 61 74 65 0d 0a 43	..gzip, de flate -C
	0160	6f 6e 6e 65 63 74 69 6f	6e 3a 20 6b 65 65 70 7d	..onnectio n: keep-

Figure 47: El paquete ESP contiene un paquete de HTTP

En la traza de la Figura 48 observamos la presencia de lo que claramente es un correo electrónico. Podemos ver los campos de escritor y remitente, así como todas las cabeceras y el contenido (que solo es la palabra “Prueba2”). De esto se deduce que se trata de un envío de correo por SMTP, este se diferencia de una recepción de correo de POP en que este último tendría una cadena +OK como podemos observar en la figura 49.

93	116.319319842	192.168.56.102	192.168.56.103	ESP	156 ESP (SPI=0xc12becdc)
94	116.320328534	192.168.56.103	192.168.56.102	ESP	556 ESP (SPI=0xce997e4b)
95	116.334441749	192.168.56.102	192.168.56.103	ESP	128 ESP (SPI=0xc12becdc)
96	116.351865538	192.168.56.103	192.168.56.102	ESP	104 ESP (SPI=0xce997e4b)
97	116.363333333	192.168.56.103	192.168.56.102	ESP	104 ESP (SPI=0xc12becdc)
▶ Frame 94: 556 bytes on wire (4448)					
▶ Linux cooked capture v1					
▶ Internet Protocol Version 4, Src:					
▼ Encapsulating Security Payload					
ESP SPI: 0xce997e4b (346616583)					
ESP Sequence: 30					
0000	00 04 00 01 00 06 08 00	27 a6 e9 bc 00 00 08 00@2K...8g		
0010	45 00 02 1c fa f4 40 00	40 32 4b 9d c0 a8 38 67	..8f...K.....		
0020	c0 a8 38 66 ce 99 7e 4b	00 00 00 1e d2 9c 00 19w.....		
0030	77 d0 ce af a6 99 d6 e7	80 18 00 e5 f7 b1 00 00;ZNT0:		
0040	01 01 08 0a 00 09 7d ba	00 3b 5a 4e 54 6f 3a 20eduardo_13@nebul		
0050	65 64 75 61 72 64 6f 5f	31 33 40 6e 65 62 75 6canexus13 45.com		
0060	61 0e 65 78 75 73 31 33	34 35 2e 63 6f 6d 0d 0aFrom: al fredro.45		
0070	46 72 6f 6d 3a 20 61 6c	66 72 65 64 6f 5f 34 35<alfred o.45@neb		
0080	20 3c 61 6c 66 72 65 64	6f 5f 34 35 40 6e 65 62ulanexus 1345.com		
0090	75 6c 61 6e 65 78 75 73	31 33 34 35 2e 63 6f 6d> Subje ct: Prue		
00a0	3e 0d 0a 53 75 62 6a 65	63 74 3a 20 50 72 75 65ba2- Mes sage-ID:		
00b0	62 61 32 0d 0a 4d 65 73	73 61 67 65 2d 49 44 3a<3fb7cb 39-75d9-		
00c0	20 3c 33 66 62 37 63 62	33 39 2d 37 35 64 39 2d39c2-a04 5-e3d3cd		
00d0	33 39 63 32 2d 61 30 34	35 2d 65 33 64 33 63 643bdac0@n ebulanex		
00e0	33 62 64 61 63 30 40 6e	65 62 75 6c 61 6e 65 78		

Figure 48: El paquete ESP contiene un paquete de SMTP

122	127.532561161	192.168.56.102	192.168.56.103	ESP	924 ESP (SPI=0xc12becdc)
123	127.554725406	192.168.56.103	192.168.56.102	ESP	104 ESP (SPI=0xce997e4b)
124	127.558035368	192.168.56.102	192.168.56.103	ESP	116 ESP (SPI=0xc12becdc)
125	127.561707753	192.168.56.103	192.168.56.102	ESP	100 ESP (SPI=0xce997e4b)
126	127.562119831	192.168.56.102	192.168.56.103	ESP	100 ESP (SPI=0xc12becdc)
Frame 122: 924 bytes on wire (739) [Captured on 2024-08-26 10:21:20.102]					
Linux cooked capture v1					
Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.103					
Encapsulating Security Payload					
ESP SPI: 0xc12becdc (324088137)					
ESP Sequence: 42					
0000	00 00 00 01 00 06 08 00	27 74 0f ba 00 00 08 00t.....		
0010	45 00 03 8c 88 89 40 00	40 32 bc 98 c0 a8 38 66@2....8f		
0020	c0 a8 38 67 c1 2b ec dc	00 00 00 2a 00 6e e9 9e8g+...n...		
0030	2e f5 d5 98 7d 6d ed 9b	80 18 00 e3 8c e3 00 00)m.....		
0040	01 01 08 0a 00 3b 65 40	00 09 88 ad 2b 4f 4b 20;e@+OK		
0050	38 30 36 20 6f 63 74 65	74 73 0d 0a 52 65 74 75806 octe ts- Retu		
0060	72 6e 2d 70 61 74 68 3a	20 3c 61 6c 66 72 65 64rn-path: <alfred		
0070	6f 5f 34 35 40 6e 65 62	75 6c 61 6e 65 78 75 73o.45@nebulanexus13		
0080	31 33 34 35 2e 63 6f 6d	3e 0d 0a 45 6e 76 65 6c1345.com > Envel		
0090	6f 70 65 2d 74 6f 3a 20	65 64 75 61 72 64 6f 5fope-to: eduardo_		
00a0	31 33 40 6e 65 62 75 6c	61 6e 65 78 75 73 31 3313@nebulanexus13		
00b0	34 35 2e 63 6f 6d 0d 0a	44 65 6c 69 76 65 72 7945.com-- Delivery		
00c0	2d 64 61 74 65 3a 20 53	75 6e 2c 20 31 32 20 4d-date: S un, 12 M		
00d0	61 79 20 32 30 32 34 20	31 37 3a 30 31 3a 32 30ay 2024 17:01:20		
00e0	20 2b 30 32 30 30 0d 0a	52 65 63 65 69 76 65 64+0200... Received		
00f0	3a 20 66 72 6f 6d 20 5b	31 39 32 2e 31 36 38 2e: from [192.168.		
0100	25 26 2a 24 20 22 54 04	0a 00 63 70 70 72 65 7956 1021... Subje ct		

Figure 49: El paquete ESP contiene un paquete de POP

7 Conclusiones

7.1 Horas Empleadas

Si bien no se ha realizado un seguimiento riguroso de las horas empleadas, estimamos que hemos empleado de media unas cincuenta horas por persona para la realización de esta práctica. De esas cuarenta y dos horas: alrededor de veintidos se han empleado en la implementación y reimplementación de los protocolos, trece horas se han dedicado a la documentación y obtención de trazas y el resto a resolución de problemas.

7.2 Problemas Encontrados

Durante la implementación de los diferentes protocolos siempre ha habido algún que otro problema, normalmente porque se escapa alguna letra al copiar o poner un nombre. A continuación, explicamos cuáles fueron los errores de mayor consecuencia.

Durante la implementación del servidor de correo no experimentamos ningún problema. Sin embargo, cuando intentamos cambiar el nombre del servicio para que siguiera las indicaciones de la práctica y con el servidor de DNS ya implementado, Thunderbird no permitía el ingreso de los nuevos usuarios (alfredo_45 y edu_15, como dice la práctica) por alguna una razón desconocida. La solución al final fue eliminar todos los usuarios y salidas de servidor configuradas en Thunderbird y comenzar la configuración de la aplicación desde cero.

Un escenario similar ocurrió durante la implementación del servicio DNS se tuvo que volver a implementar porque al cambiar el nombre de sstt.um al de la organización pedido en el enunciado este dejaba de funcionar. Sin embargo, el otro problema que ocurrió es que por alguna extraña razón una vez implementado el DNS y cuando ya habíamos comprobado que funcionaba. Al reiniciar el servidor, el DNS dejó de funcionar, y tras mucha búsqueda, nos dimos cuenta que el fichero db había cambiado los permisos para que no pudiese ser accedido por usuarios de fuera del grupo del creador. Una vez cambiados los permisos volvió a funcionar.

Durante la implementación de certificados CA se nos olvidó la palabra clave y hubo que cargar una imagen antigua y volver a empezar.

Se intentó definir las IPs de las máquinas virtuales en sus interfaces de host a estáticas para que no se cambien y las configuraciones dejen de funcionar. Una vez hecho, se probó y funciona correctamente y se podían conectar bien. La conexión a Internet dejó de funcionar. Se deshizo el cambio.