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

List of Figures

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 organizacion, sobre el escenario de red que se muestra a continuación:

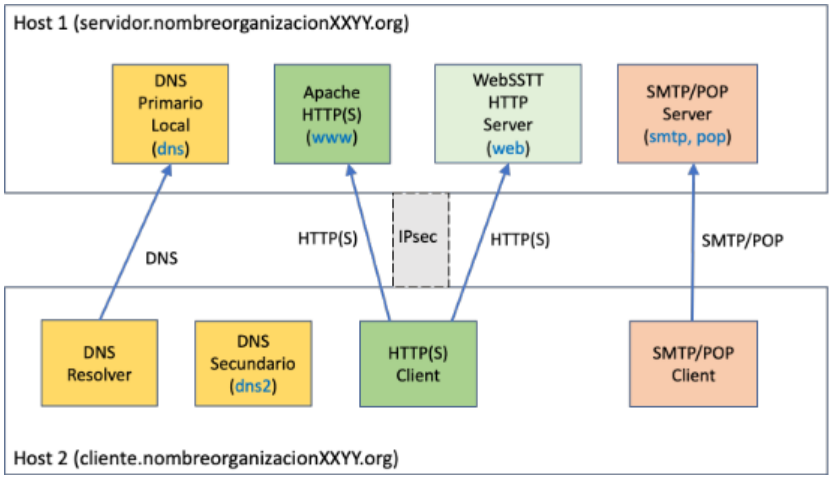

Figure 1: Escenario Objetivo

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:

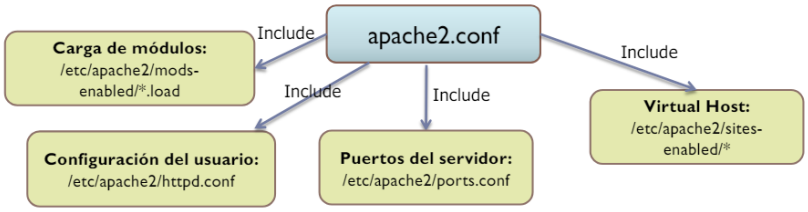

Figure 2: Módulos apache2.conf

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 la 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 los 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.
- Especifíca 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 especifica 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 lineas 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 entran 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:

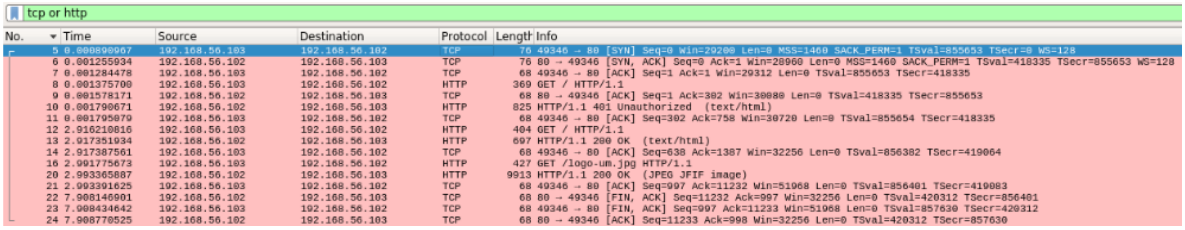

Figure 8: Traza HTTP

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 siguiente 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.

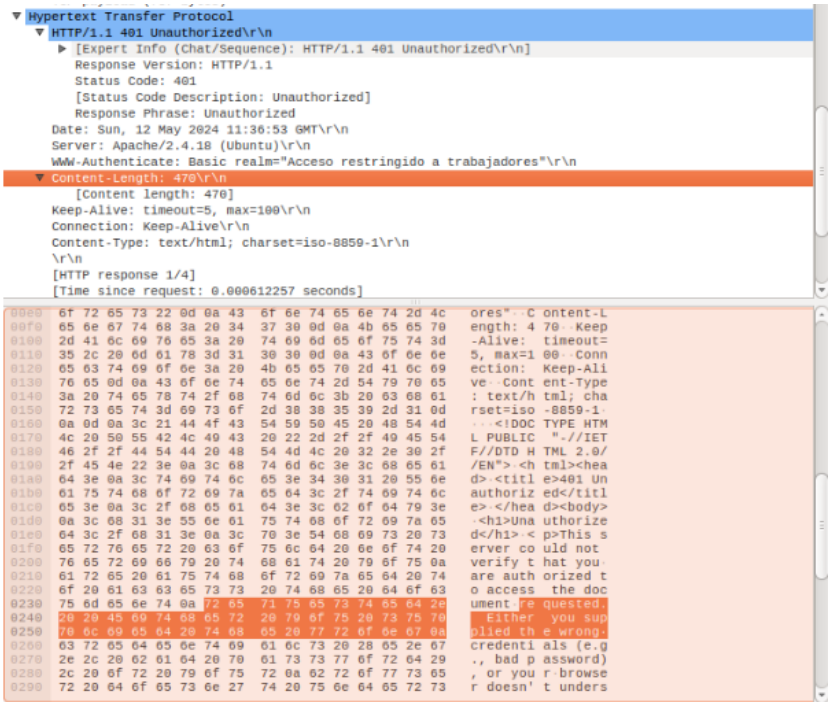

Figure 9: Contenido HTTP

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.

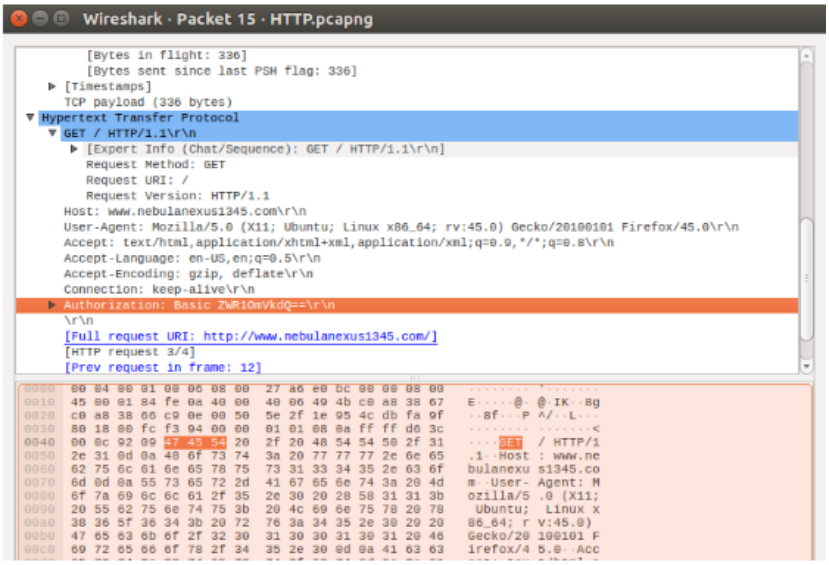

Figure 10: Petición GET con credenciales correctas

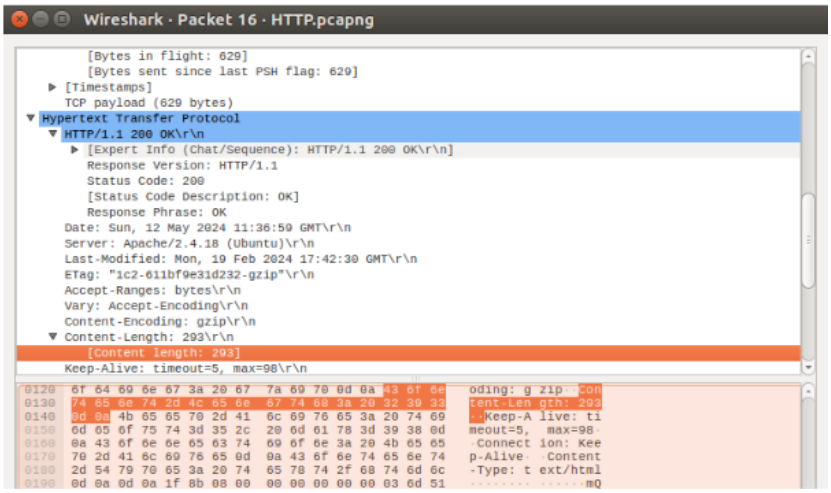

Figure 11: Respuesta del servidor 200 OK

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 maquina 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:

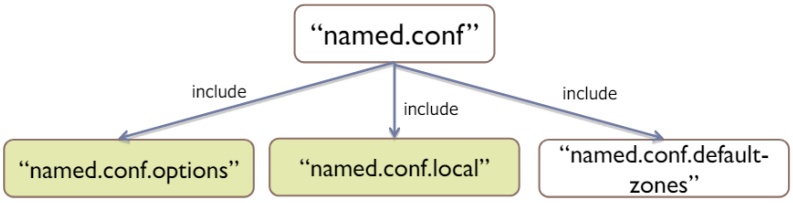

Figure 13: Jerarquía /etc/bind

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

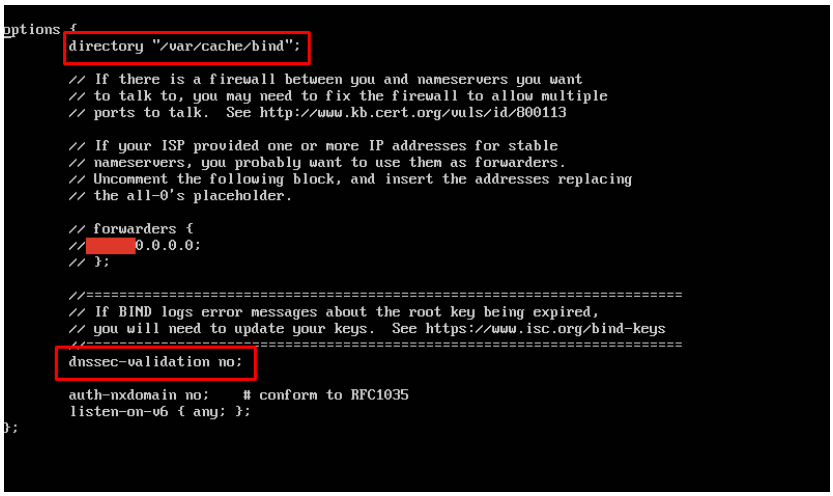

Figure 14: Fichero /etc/bind/named.conf.options

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

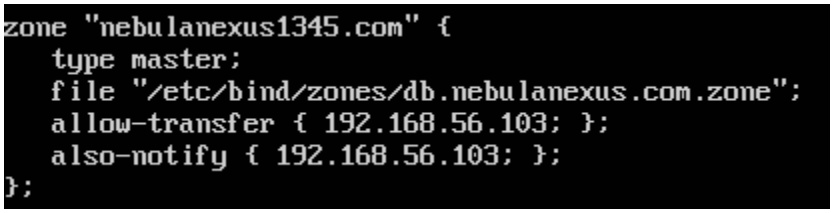

Figure 15: Fichero /etc/bind/named.conf.local

- **'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.

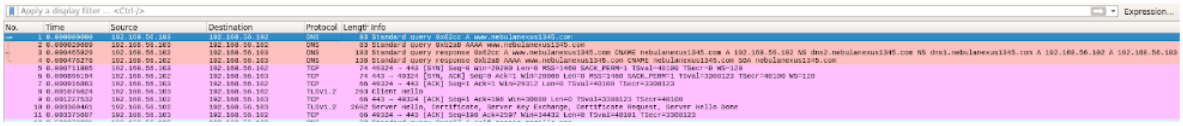

Figure 17: Traza DNS

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.

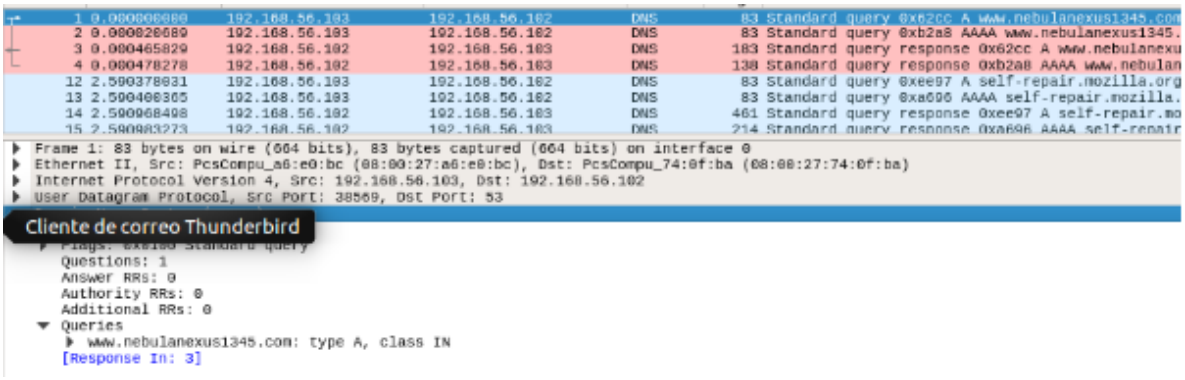

Figure 18: Paquete 1 DNS

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.

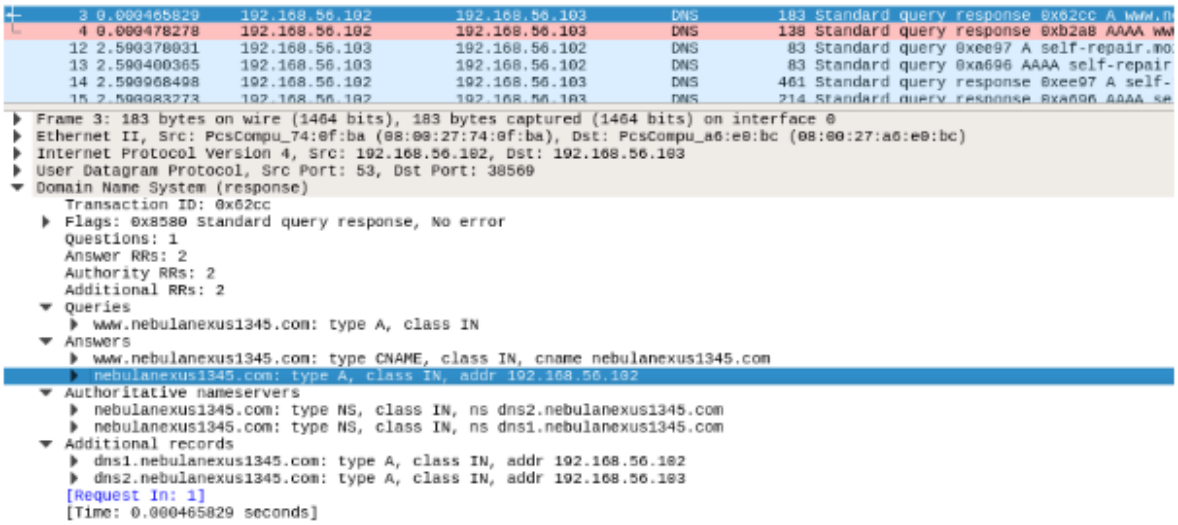

Figure 19: Paquete 2 DNS

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: afredo_45@nebulanexus1345.com
```

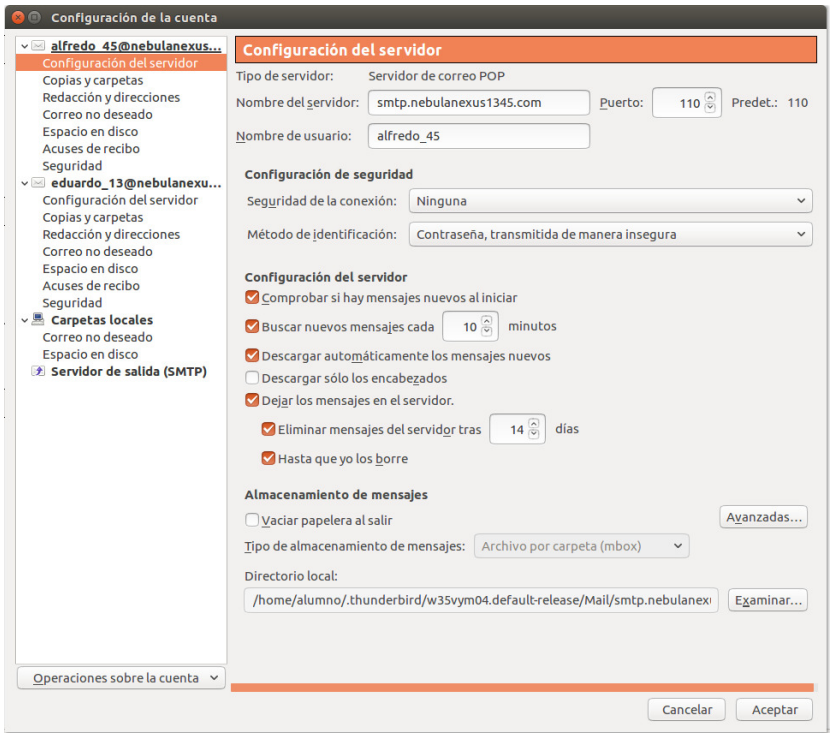

Figure 20: Configuración Servidor en Thunderbird

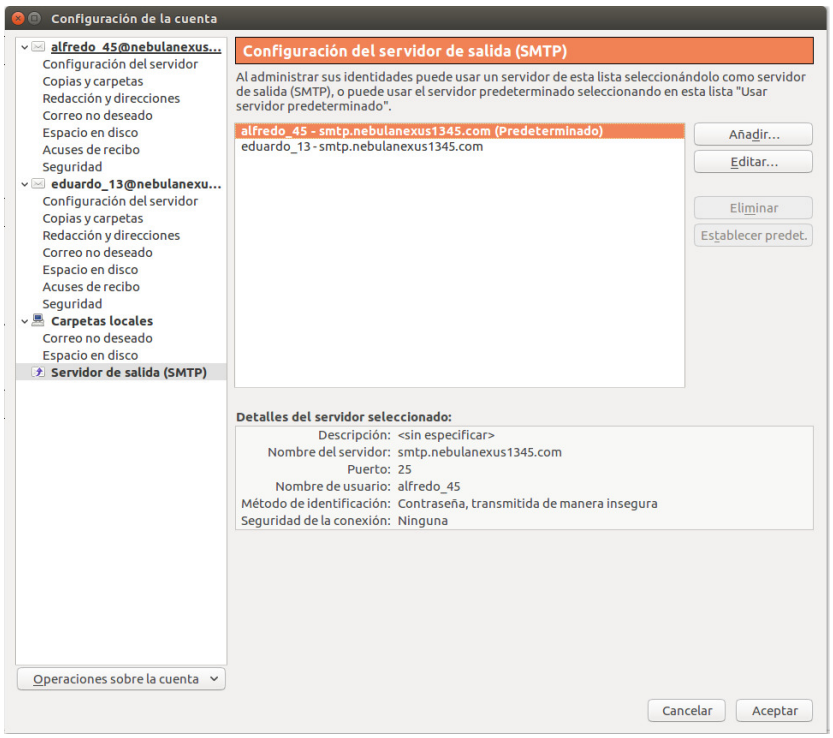

Figure 21: Configuración Servidor de Salida (SMTP)

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.

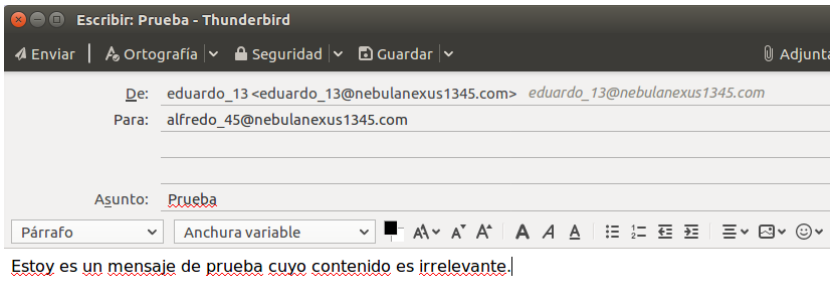

Figure 22: Escribimos un correo para enviar por SMTP

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.

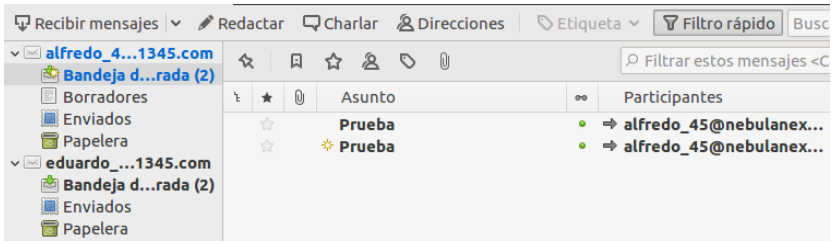

Figure 23: Recibimos el correo por POP

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 las estrucutra 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 pertence 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 confirmacidolo.

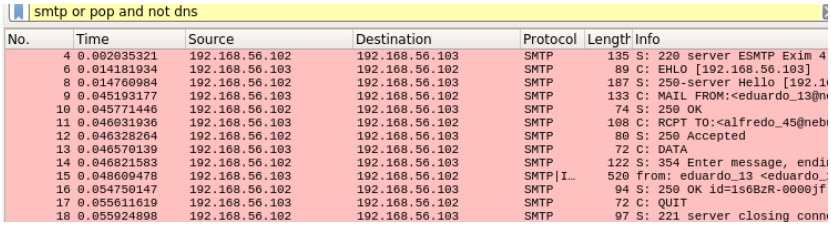

Figure 24: Interacción de Thunderbird con el servidor SMTP

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.

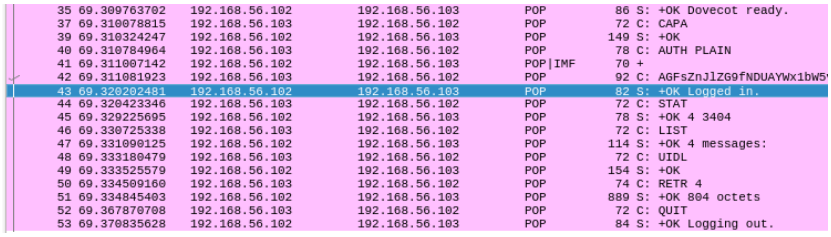

Figure 25: Interacción de Thunderbird con el servidor POP

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.

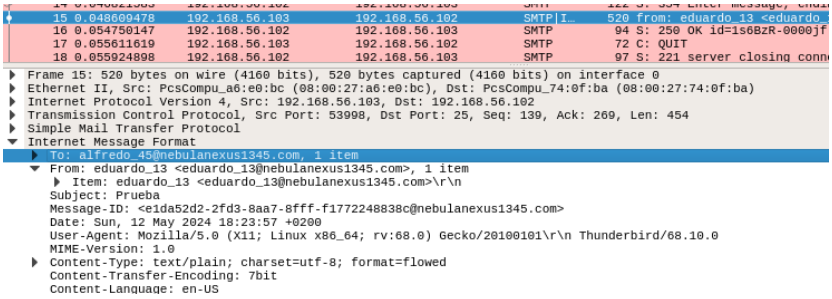

Figure 26: Correo siendo enviado por SMTP

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.

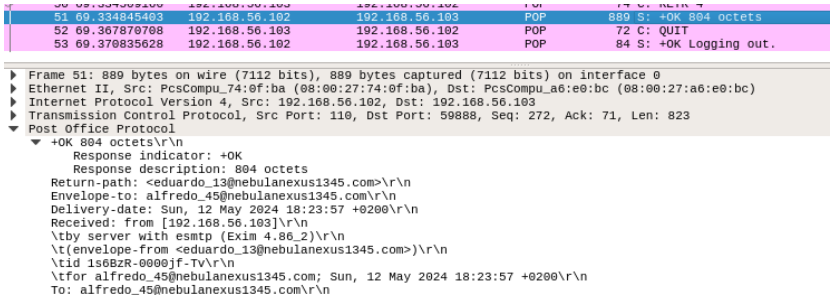

Figure 27: Correo siendo recibido por POP

Finalmente, nos podemos estar preguntando que si hemos especificado que la autorización sea “plana”, sin cifrado, porqué 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.

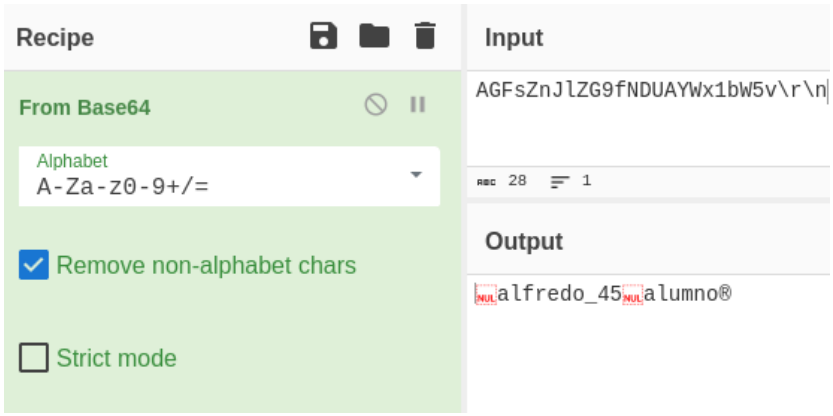

Figure 28: Descifrando el contenido del AUTH con cyberchef

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 tambien 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:
   - clrnumber con valor 00 (números de serie para CLRs).
   - index.txt vacio.
   - 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:

- **Directorrio 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 tambien 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 estrucutra de nombres para PKI el suiguiente 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 inluye 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.
- 'servercrs.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.

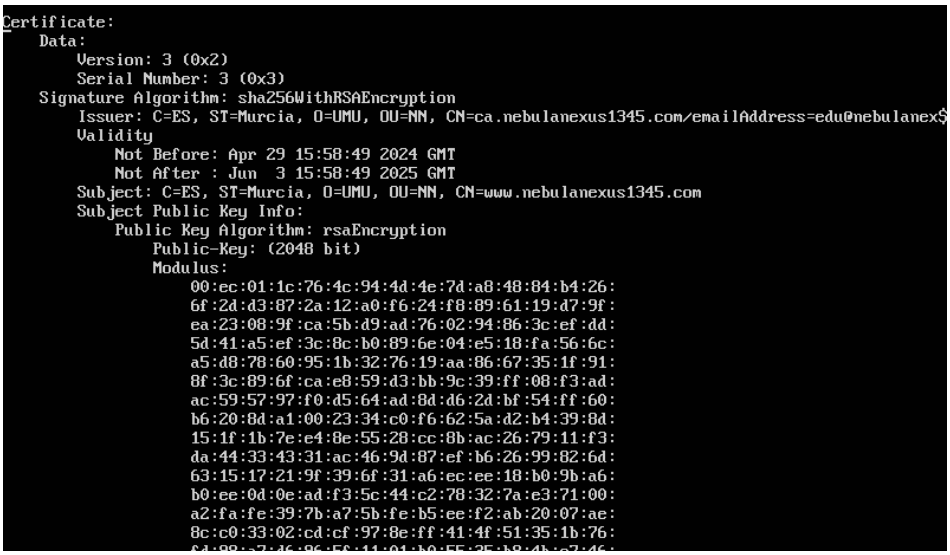

Figure 32: Contenido servercert.pem

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-enable/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.

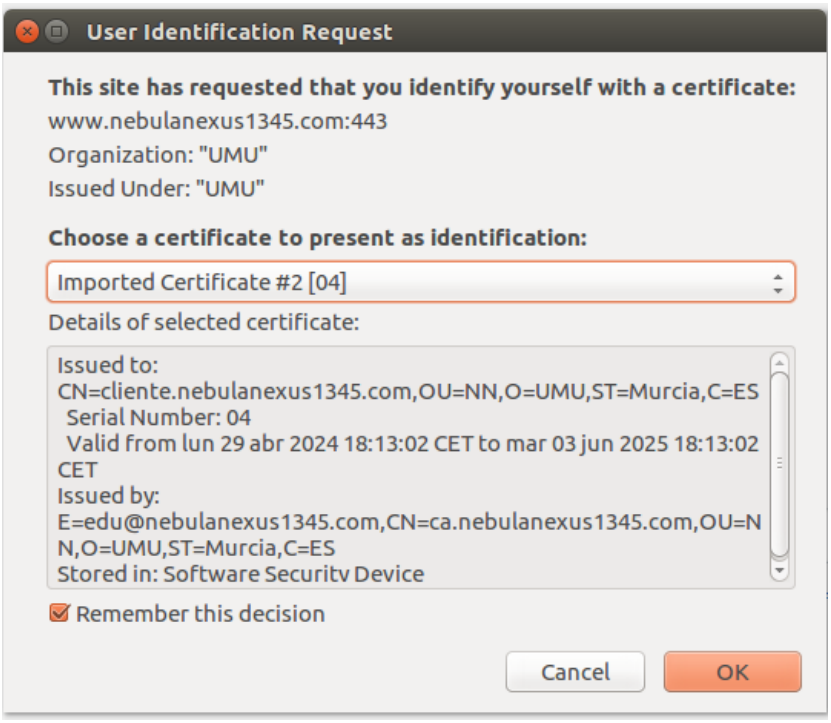

Figure 35: Utilizamos el certificado generado por el servidor

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.

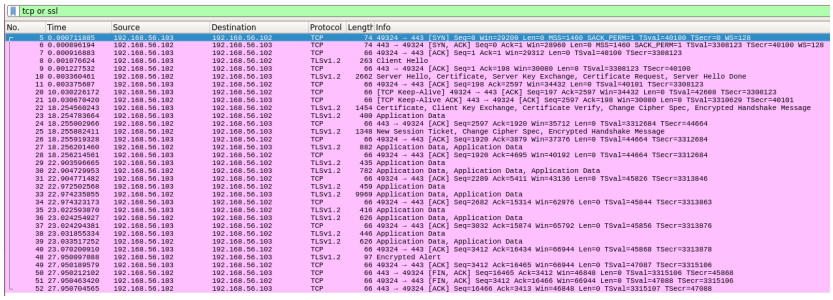

Figure 37: Interacción completa de TLS y TCP

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, Certicate y ServerHelloDone se encuentran dentro de un solo packet. Aquí abajo podemos ver un diagrama que representa cómo es la interacción.

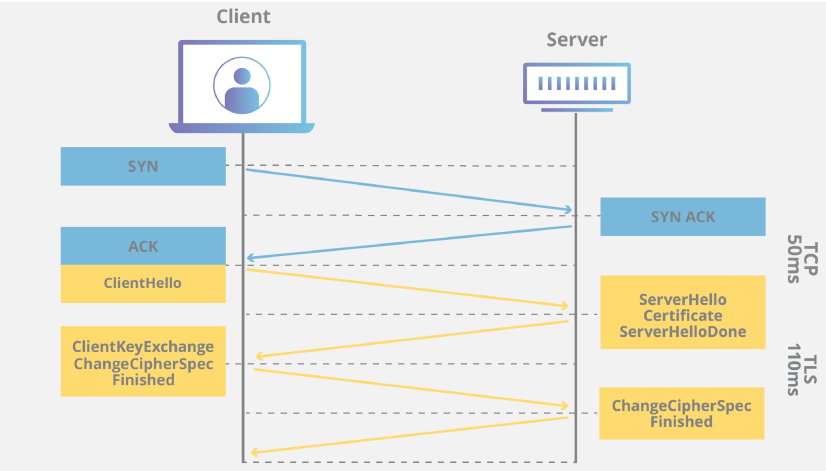

Figure 38: source: cloudfare.com

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

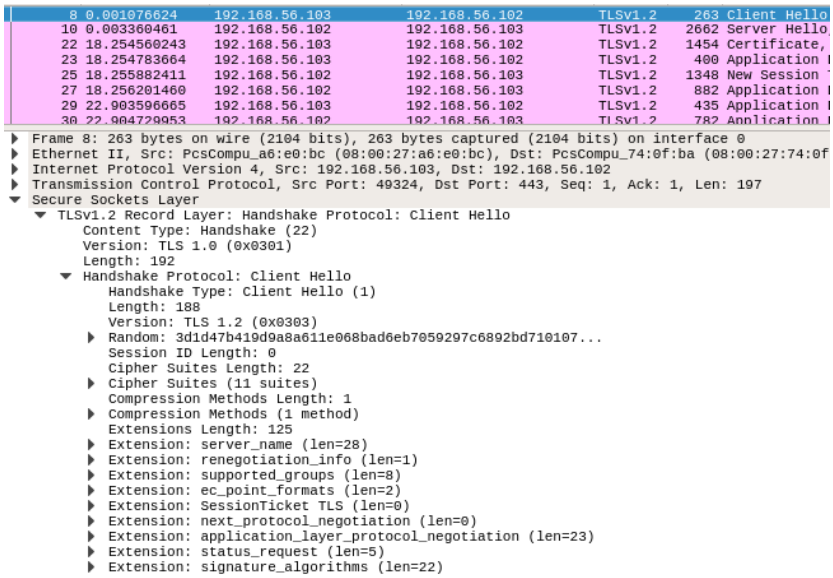

Figure 39: Primer mensaje del handshake

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.

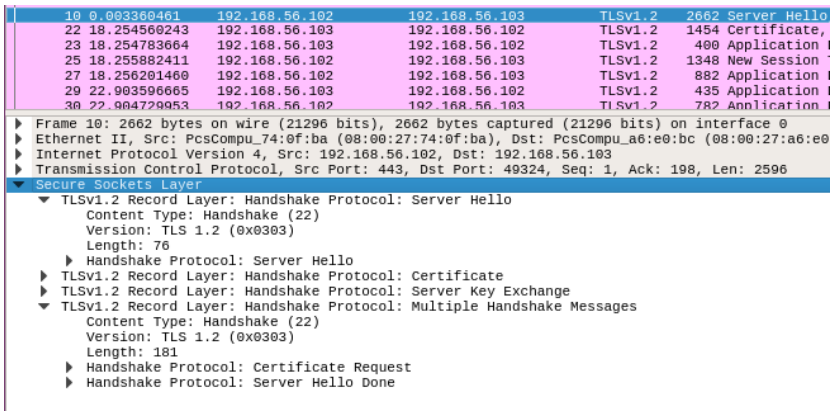

Figure 40: Segundo paquete del handshake

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 siquera esperar el final del propio handshake, pues ya posee todas la información necesario para calcular la clave simétrica.

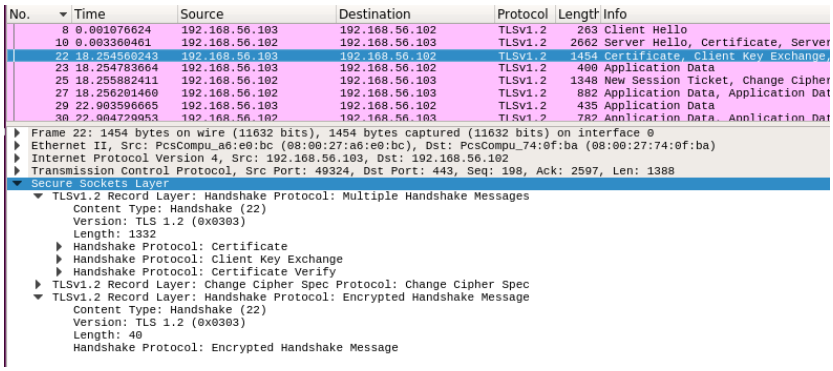

Figure 41: Tercer paquete del handshake

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.

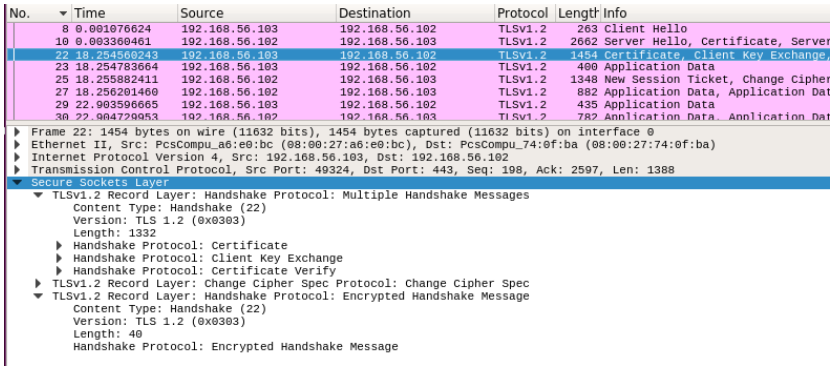

Figure 42: Último paquete del handshake

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 hacmeos 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 herrramienta 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).

```
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.

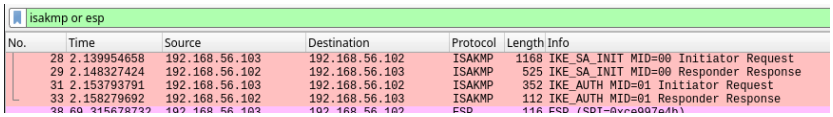

Figure 45: Handshake de IKE completo

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.

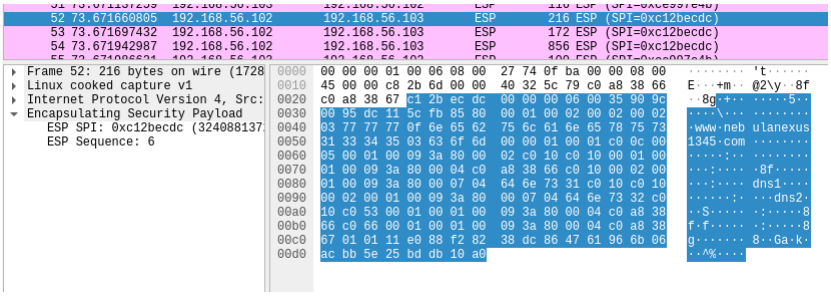

Figure 46: El paquete ESP contiene un paquete de DNS

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.

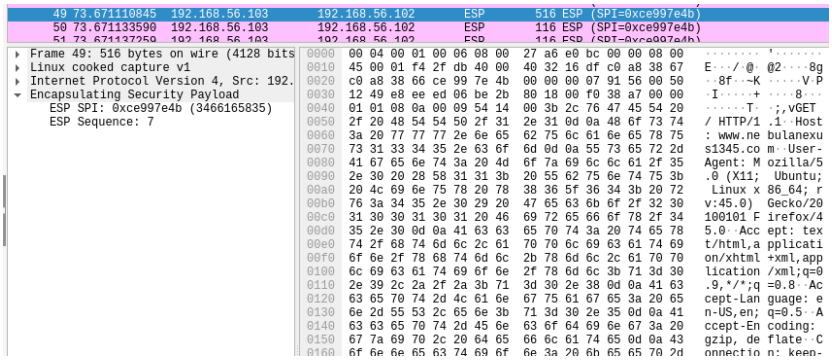

Figure 47: El paquete ESP contiene un paquete de HTTP

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 envio 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.

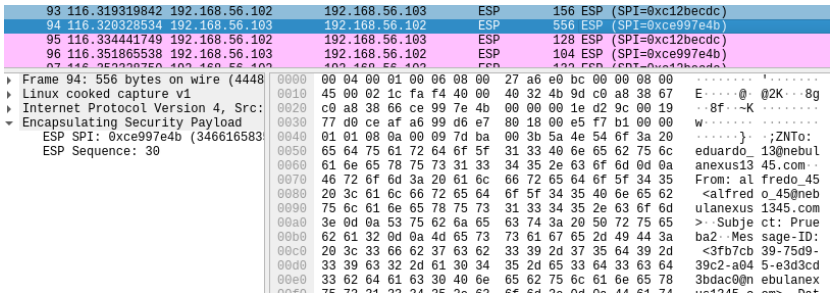

Figure 48: El paquete ESP contiene un paquete de SMTP

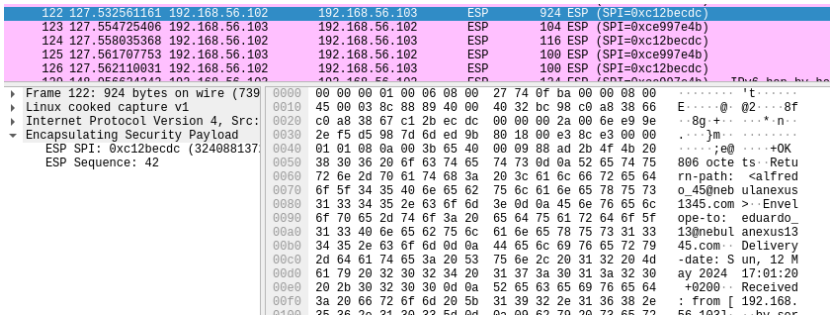

Figure 49: El paquete ESP contiene un paquete de POP

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.