Práctica 3 – Servicios avanzados de red: HTTP y HTTPS

1.1 Objetivo

El objetivo de esta práctica es conocer y familiarizarse con las tareas habituales en la administración y configuración de los protocolos HTTP (*HyperText Transfer Protocol*) y HTTPS (HTTP Secure) usando el servidor Apache2 en Linux. Para ello se proponen los siguientes ejercicios:

- Instalación y configuración básica de Apache2
- Configuración de *Virtual Hosts*
- Restricción de accesos a sitios Web
- Captura y análisis del tráfico HTTP
- Generación de un certificado SSL (Secure Socket Layer) autofirmado con la utilidad `openssl` para autenticar un sitio Web
- Configuración de un sitio Web con Apache para su acceso mediante HTTPS
- Captura y análisis del tráfico TLS (*Transport Layer Security*) generado cuando se accede a un sitio web mediante HTTPS

1.2 Información básica para la realización de la práctica

En esta sección se ofrece la información básica y las referencias necesarias para llevar a cabo las tareas que se proponen en la práctica.

1.2.1 Acceso al sistema y elección de sistema operativo

Para la realización de esta práctica, es necesario arrancar el equipo con la opción "Redes"→"Ubuntu 20.04". La práctica se realizará en parejas en donde uno de los equipos actuará como cliente y otro como servidor web. Será en este último en donde se configure adecuadamente Apache2.

Una vez que se haya identificado, puede pasar a modo *superusuario* mediante el siguiente comando, y utilizando la contraseña "**finisterre**":

```
# sudo su
```

1.2.2 Gestión de servicios básicos

Aunque existen varias formas de gestionar servicios de red para el objetivo de esta práctica se utilizará, de forma aislada (*standalone*), un servidor Apache2.

Esto consiste básicamente en, tras editar los preceptivos ficheros de configuración del propio servicio, lanzar un ejecutable (`daemon`) en *background*, controlándolo posteriormente si fuese necesario con el comando `systemctl`.

Para obtener más ayuda sobre la configuración de este servidor, consulte la sintaxis del fichero de configuración en el manual del sistema:

```
# man xinetd.conf
```

1.2.3 Gestión y configuración básica de Apache2 (HTTP)

El protocolo HTTP facilita el acceso desde un cliente (que ofrece una interfaz universal o *navegador*) a objetos web (texto, imágenes, vídeo, etc) situados en un servidor identificado por su dirección IP o su nombre de dominio. El servidor Apache2 una de las implementaciones de HTTP de mayor difusión, ya está instalado en las máquinas Ubuntu 20.04 y se ejecutará en modo *standalone*. Con el siguiente comando podemos gestionar y comprobar el estado del servicio (iniciar, parar, reiniciar, re-ejecutar sin perder conexiones, deshabilitar o habilitar el inicio automático):

```
# sudo systemctl [start|stop|restart|reload|disable|enable] apache2
```

Para comprobar la correcta instalación y funcionamiento del servidor, después de iniciar este, acceder a la URL `http://localhost` o `http://direccion_ip` desde cualquier navegador usando cualquiera de las direcciones IP locales disponibles. La página web que aparecerá será la que se define por defecto durante la instalación de Apache2.

La configuración de Apache2 se lleva a cabo principalmente mediante la edición de una serie de ficheros de texto. Los más relevantes son:

- `/var/www/html`: Aquí se almacena el contenido del sitio web. Se puede modificar en los ficheros de configuración.
- `/etc/apache2/apache2.conf`: Configuración principal de Apache2. Las diferentes características y parámetros se definen mediante directivas.
- `/etc/apache2/ports.conf` : Puertos en los que Apache2 atenderá solicitudes.
- `/etc/apache2/sites-available/`: Directorio donde almacenan los *hosts* virtuales. Ver Sección 1.2.4 para más detalles.
- `/etc/apache2/sites-enabled/`: Directorio donde se almacenan los *hosts* virtuales.
- `/etc/apache2/mods-available/` y `/etc/apache2/mods-enabled/`: Contienen los módulos disponibles y habilitados, respectivamente. Por ejemplo para dar soporte a transacciones MySQL o PHP.
- `/var/log/apache2/access.log`: Fichero de trazas donde se almacenan todas las transacciones.
- `/var/log/apache2/error.log`: Fichero donde se registran los errores.

Para obtener más ayuda sobre la configuración de Apache2 y las directivas disponibles, abra un navegador web y visite la dirección:

```
http://httpd.apache.org/docs/2.4/
```

1.2.4 Configuración de Virtual Hosts en Apache2

Un servidor Apache2 puede dar servicio a varios sitios Web (con diferentes nombres de dominio) simultáneamente como si fueran funcionalmente diferentes, aunque en realidad es un único proceso el que los sirve. A cada uno de estos sitios se les denomina *host* virtual.

Cada *host* virtual puede tener su propio directorio raíz, su propia política de seguridad y en definitiva, su propia configuración.

La configuración comienza creando un directorio raíz diferente para cada uno de los sitios web o *hosts* virtuales a configurar:

```
# sudo mkdir -p /var/www/midominio1.com/
# sudo mkdir -p /var/www/midominio2.com/
```

Dentro de cada dominio debe crearse un fichero `index.html` como por ejemplo:

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">Mi dominio 1</title>
  </head>
  <body>
  <h1>Home de mi dominio 1</h1>
  </body>
</html>
```

Es necesario cambiar el propietario de los directorios raíz y de sus ficheros al usuario de Apache2 `www-data`, por ejemplo:

```
# chown -R www-data:www-data /var/www/midominio1.com
```

En el directorio `/etc/apache2/sites-available` editamos los ficheros de configuración de los diferentes *hosts* virtuales a crear. Es habitual nombrar los ficheros con el correspondiente nombre de dominio, por ejemplo `midominio1.com.conf`

```
  <VirtualHost *:80>
    ServerName midominio1.com
    ServerAlias www.midominio1.com
    ServerAdmin webmaster@midominio1.com
    DocumentRoot /var/www/midominio1.com/

    <Directory /var/www/midominio1.com/>
```

```
        Options -Indexes +FollowSymLinks
        AllowOverride All
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/midominio1.com-error.log
    CustomLog ${APACHE_LOG_DIR}/midominio1.com-access.log combined
</VirtualHost>
```


Para obtener más ayuda sobre la configuración de Apache2 y las directivas disponibles, abra el navegador web y visite la dirección:

```
http://httpd.apache.org/docs/2.4/vhosts/
```

Una vez creados los ficheros de configuración, hay que crear enlaces simbólicos desde el directorio `sites-available` al directorio `sites-enabled`. Esto se puede hacer directamente con el comando `ln`, o bien usando un *script* especialmente preparado en la distribución de Apache2:

```
# sudo a2ensite midominio1.com
```

Comprobamos si la configuración tiene algún error:

```
# sudo apachectl configtest
```

La creación de *hosts* virtuales no implica que se creen automáticamente las entradas para que DNS sepa resolver adecuadamente los nombres de los dominios creados. Para ello, habría que configurar adecuadamente DNS para los nombres de dominios virtuales. No obstante, para hacer pruebas, esto se puede hacer localmente editando el fichero `/etc/hosts`, añadiendo los nombres de dominios virtuales creados a una de las IP locales, por ejemplo la IP de la interfaz de datos. Una vez hecho esto, reiniciamos el servicio:

```
# sudo systemctl restart apache2
```

Y finalmente, desde un navegador comprobamos que los diferentes dominios creados son servidos sin problema y de acuerdo con la configuración deseada.

1.2.5 Restricción de accesos

En muchas ocasiones es conveniente restringir el acceso ciertas zonas o directorios del sitio web con algún nivel de protección. Apache2 proporciona funcionalidad para ello. En este ejercicio aprenderemos a restringir el acceso al directorio `http://<dominio>/mi_zona_restringida`.

En primer lugar, hay que crear el directorio `/var/www/<dominio>/mi_zona_restringida`

Las directivas para restringir accesos, se pueden definir en el fichero de configuración principal del servidor (típicamente en la sección `<Directory>` de `httpd.conf`), o en cada uno de directorios a restringir mediante ficheros `.htaccess`

Si se hace con `.htaccess`, en el servidor hay que permitir que se puedan definir directivas de autenticación en estos ficheros. Esto se hace con la directiva `AllowOverride`, la cual especifica qué directivas pueden ser definidas en el fichero de configuración del directorio asociado al sitio desplegado. Por ejemplo, si queremos permitir la configuración de directivas de autenticación con `.htaccess`, modificaremos el fichero `/etc/apache2/apache2.conf` tal y como sigue:

```
AllowOverride All
```

Además, es necesario crear un fichero de contraseñas, para ello usaremos la utilidad `htpasswd` con la siguiente sintaxis, en la que la opción `-c` es para crear por primera vez el fichero si no existe:

```
# sudo htpasswd -c /usr/local/passwords.pd miusuario
```

Una vez hecho esto, el siguiente paso se puede realizar editando el fichero `.htaccess` ubicado en `/var/www/<dominio>/mi_zona_restringida` como se muestra a continuación

```
AuthType Basic
AuthName "Directorio con control de acceso"
# (las siguientes directivas son opcionales)
AuthUserFile "/usr/local/passwords.pd"
Require user miusuario
```

`AuthType` define la metodología de autenticación. Nótese que `Basic` implica un mecanismo de identificación más que de autenticación, ya que `Basic` consiste en enviar un *login* y un *password* en texto plano desde el cliente, procedimiento este vulnerable a ataques de repetición. Para evitar esta vulnerabilidad, es recomendable usar el módulo `mod_ssl` que encripta toda la transacción.
`AuthName` define un nombre para la zona de seguridad. Una vez que nos hayamos autenticado en esta zona, el cliente reintentará automáticamente las mismas credenciales en todas las zonas protegidas con el mismo nombre en este servidor.
`AuthUserFile` define el fichero de contraseñas.
`Require` define al usuario al que se le permite el acceso. Alternativamente se puede usar `Require valid-user` lo que implicaría que se permite el acceso genérico a cualquier usuario definido en el fichero de contraseñas.

Finalmente, para que el proceso se reconfigure leyendo las nuevas directivas, tras la edición realizada es necesario re-arrancar el proceso mediante la utilidad `systemctl`.

1.2.6 Creación de un certificado SSL

Un certificado digital típicamente se expide por una Autoridad de Certificación (entidad de confianza) que vincula de forma fehaciente a una entidad con su clave pública. Se trata de una especie de tarjeta de visita en donde la Autoridad de Certificación garantiza que esa vinculación es cierta e irrevocable. El certificado está firmado con la clave privada de la Autoridad de Certificación, de tal manera que vincula datos asociados al sitio web (su identidad) y la clave pública del mismo. De este modo, es posible autenticar a un sitio web en Internet, ya que si suponemos la hipótesis de que la autoridad certificadora es de confianza, si enviamos los mensajes cifrados con la clave pública del sitio web (garantizada por la autoridad) obtenida del certificado, nadie excepto el que posea la clave privada podrá descifrar los mensajes cifrados, por lo que con este cifrado estaremos autenticando al servidor.

En esta práctica se usará la utilidad de línea de comandos `openssl` para crear el certificado SSL autofirmado (no estará expedido por una Autoridad de Certificación). En concreto usaremos el siguiente comando para crear el certificado:

```
# sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
```

Consulte la página de manual y compruebe las diferentes opciones y argumentos del comando anterior.

A continuación, le pedirá la siguiente información para incluirla en el certificado:

```
Country Name (2 letter code) [XX]: SP
State or Province Name (full name) []: Granada
Locality Name (eg, city) [Default City]: Granada
Organization Name (eg, company) [Default Company Ltd]: UGR
Organizational Unit Name (eg, section) []: DTSTC
Common Name (eg, your name or your server's hostname) []: frdominioseguro.com
Email Address []: webmaster@frdominioseguro.com
```

Las principales opciones empleadas para generar el certificado SSL se explican a continuación:

- `req -x509`: Selección de solicitud de firma de certificados X.509 (formato del certificado).
- `-nodes`: Indicamos que no queremos proteger el certificado con contraseña dado que Apache necesitará leerlo sin intervención del usuario.
- `-days 365`: Periodo de validez del certificado de un año.
- `-newkey rsa:2048`: Indicamos que queremos generar una clave privada para el certificado (porque no la hemos creado anteriormente). En concreto se creará una clave RSA de 2048 bits de longitud.

- `-keyout`: Para indicar el directorio en el cual queremos almacenar la clave privada creada.

- `-out`: Para indicar el directorio en el que almacenaremos el certificado a crear.

1.2.7 Configuración de Apache para habilitar el acceso a un sitio por HTTPS

Una vez disponemos de un certificado válido, es necesario configurar Apache para usar SSL. En primer lugar, es necesario habilitar el módulo Apache2 `mod_ssl`. Para ello se puede usar el *script* `a2enmod` incluido en la distribución de Apache2 como sigue:

```
# sudo a2enmod ssl
```

A continuación, Apache se debe reiniciar para activar el módulo habilitado:

```
# sudo systemctl restart apache2
```

Ahora vamos a crear un *virtual host* para que funcione con HTTPS haciendo uso del certificado SSL. En primer lugar, en el directorio `/etc/apache2/sites-available` crearemos el fichero de configuración del *virtual host* con la siguiente configuración mínima:

```
<VirtualHost *:443>
ServerName frdominioseguro.com
DocumentRoot /var/www/frdominioseguro.com

SSLEngine on
SSLProtocol -all +TLSv1.2
SSLCertificateFile          /etc/ssl/certs/apache-selfsigned.crt
SSLCertificateKeyFile     /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
```

Hay que tener en cuenta que el valor `Common Name` que configuró para el certificado ha de coincidir con el valor de la directiva `ServerName` en la configuración de arriba.

En el directorio especificado con la directiva `DocumentRoot` crearemos un archivo HTML muy simple para testeo. Primero, comenzamos creando dicho directorio raíz para el sitio:

```
# sudo mkdir -p /var/www/frdominioseguro.com
```

Ahora creamos dentro de /var/www/frdominioseguro.com el archivo HTML index.html con un editor de texto con el siguiente contenido:

```
<h1> FR DOMINIO SEGURO <\h1>
```

Es necesario cambiar el propietario de los directorios raíz y de sus ficheros al usuario de Apache2 `www-data`, por ejemplo:

```
# chown -R www-data:www-data /var/www/frdominioseguro.com
```

Una vez creados los ficheros de configuración, hay que crear enlaces simbólicos desde el directorio `sites-available` al directorio `sites-enabled`. Esto se puede hacer directamente con el comando `ln`, o bien usando un *script* especialmente preparado en la distribución de Apache2:

```
# sudo a2ensite frdominioseguro.com.conf
```

Comprobamos si la configuración tiene algún error:

```
# sudo apachectl configtest
```

Puede que le aparezca un mensaje de advertencia indicando que la directiva `ServerName` no está configurada a nivel global. Opcionalmente, para eliminar este mensaje, puede configurar `ServerName` en el nombre de dominio o la dirección IP de su servidor en `/etc/apache2/apache2.conf`. Sin embargo, este mensaje es sólo un aviso y no causará problemas si el resultado contiene `Syntax OK` (no hay errores de sintaxis en su archivo de configuración).

La creación de *hosts* virtuales no implica que se creen automáticamente las entradas para que DNS sepa resolver adecuadamente los nombres de los dominios creados. Para ello, habría que configurar adecuadamente DNS para los nombres de dominios virtuales. No obstante, para hacer pruebas, esto se puede hacer localmente editando el fichero `/etc/hosts`, añadiendo los nombres de dominios virtuales creados a una de las IP locales, por ejemplo, la IP de la interfaz de datos . Una vez hecho esto, reiniciamos el servicio:

```
# sudo systemctl reload apache2
```

Y finalmente, desde un navegador comprobamos que el dominio creado es accesible usando `https://` al principio (`https://frdominioseguro.com`). El navegador mostrará una advertencia de seguridad porque estamos usando un certificado autofirmado y, por tanto, no está firmado por ninguna autoridad de certificación confiable. Permite el acceso al sitio haciendo click en `avanzado` y luego eligiendo la opción `aceptar riesgo y continuar`.

1.3 Realización práctica

1) Habilite el servicio HTTP en su equipo servidor. Abra un navegador web y pruebe a visitar la página de inicio desde dicho equipo (`http://localhost` o http://127.0.0.1). Modifique el contenido de la página de inicio, y compruebe que la dirección de su servidor es accesible.
2) Inspeccione el fichero `apache2.conf` e identifique las directivas más relevantes.
3) Cree 2 hosts virtuales con páginas de inicio diferentes y compruebe que son servidos convenientemente ante peticiones desde el cliente. Abra Wireshark e identifique los mensajes principales que se intercambian entre el cliente HTTP (navegador web) y el servidor HTTP (Apache2).
4) Cree una página de acceso restringido (es decir, que requiera usuario y contraseña antes de mostrarla) en *http://<dominio>/restringida/*. Utilice como credenciales de acceso el usuario *admin* y la contraseña *1234*.

 CHECKPOINT: Avise al profesor cuando termine esta tarea.

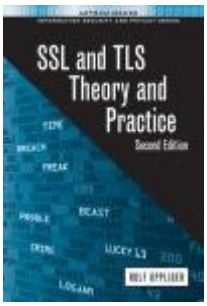
5) Cree un certificado SSL con la utilidad `openssl` para asociarlo al sitio `frpracticahttps.com`. Nombre el fichero del certificado como `frpracticahttps.crt` y el nombre del fichero de la clave privada como `frpracticahttps.key`.
6) Inspeccione los ficheros `.crt` y `frpracticahttps.key`.
7) Cree un host virtual con una página de inicio que muestre el mensaje "FR HTTPS" y configúrelo para que funcione con `HTTPS` haciendo uso del certificado creado anteriormente. Compruebe su correcto funcionamiento usando un navegador.
8) Abra Wireshark en su equipo y capture los mensajes que se generan cuando accede al sitio creado anteriormente. ¿Qué mensajes `TLS` se intercambian la aplicación cliente HTTPS (navegador web) y el servidor HTTPS (Apache2) durante el inicio de la conexión? ¿Qué información relevante se intercambia en esos mensajes? ¿Es posible ver los mensajes del protocolo HTTP ?

 CHECKPOINT: Avise al profesor cuando termine esta tarea.

BIBLIOGRAFÍA

[1] Library: OpenSSL Cookbook, 3ed By Ivan Ristić
https://www.feistyduck.com/library/openssl-cookbook/

[2] SSL and TLS: Theory and Practice, Second Edition por Rolf Oppliger. Accesible on line desde la biblioteca de UGR.



[3] https://httpd.apache.org/docs/2.4/es/ssl/

[4] Opcionalmente para instalar LAMP (Linux + Apache + MySQL + PHP) consultar https://ubunlog.com/lamp-instala-apache-mariadb-php-ubuntu-20-04/