

```
ventas: sergio maria  
finanzas: carlos roberto
```

Modifica la configuración del ejercicio anterior para que puedan acceder dos grupos de usuarios diferentes a cada directorio con el último método. ¿Puede un usuario estar en más de un grupo?


Añade un tercer directorio y otro grupo. Da acceso al tercer directorio a dos de los tres grupos. ¿Cómo se hace esto?

El protocolo HTTPS

Ya hemos visto que el protocolo HTTP es muy inseguro para transmitir información sensible. Es muy fácil capturar esta información y leer datos como nombres de usuarios o claves. Cualquier información que se envía va sin cifrar por lo que se transmite como un texto que cualquiera que capture nuestra comunicación puede leer.

El protocolo HTTPS se utiliza para evitar este problema. Sus siglas se corresponden con *Hypertext Transfer Protocol* **Secure** lo que ya indica que añade seguridad a HTTP. Realmente no es un protocolo sino que se basa en introducir una capa con el protocolo SSL (o derivados, más adelante se habla de todo esto) entre la capa de transporte y la capa de aplicación en el conjunto de protocolos TCP/IP.

El protocolo HTTPS proporciona **autenticación** de los usuarios con el servidor con el que se conecta. Por ejemplo cuando nos conectamos a nuestro correo web (por ejemplo Gmail)

suele hacerse mediante una conexión HTTPS.  Con el uso de este protocolo también se consigue **encriptación** de la información que enviamos y recibimos. Por ello cuando nos conectamos a un servidor que solo utiliza HTTPS para la autenticación el navegador nos avisa (una vez nos hemos conectado) de que el resto de la comunicación no usa HTTPS y por lo tanto la transferencia de información puede ser

insegura. Esto nos protege contra los ataques de tipo *Man-In-The-Middle*. Todo lo que se envían en un mensaje HTTPS está encriptado, incluyendo las cabeceras. Por supuesto la encriptación también está sujeta a ataques que intenten descifrar la clave y el algoritmo usados para cifrar el contenido.

Investiga qué algoritmos y qué tipo de claves se utilizan en Internet actualmente.

*¿Qué son los algoritmos de clave simétrica y los algoritmos de clave asimétrica?
¿Qué con las claves privadas y claves públicas? ¿Cómo se utilizan?*

[Esta página de la Wikipedia](#) te puede ayudar a comprender estos conceptos y los que veremos a continuación. Necesitarás tener una idea de los puntos anteriores de este ejercicio para entenderla.

Un sitio que use HTTPS debería tener todos sus contenidos protegidos por este protocolo para evitar posibles ataques o robos de información a través de las partes inseguras.

HTTPS es un poco menos eficiente que HTTP por lo que si se utiliza en sitios donde la transferencia de información es muy grande puede notarse en el rendimiento. Por supuesto en el caso de que la información transferida sea sensible, las ventajas compensan con creces a los inconvenientes.

El protocolo HTTPS usa el **puerto 443** por defecto. Al igual que sucedía con HTTP y el puerto 80, en caso de que se utilice este puerto no es necesario que el cliente lo indique en la barra de direcciones del navegador.

Certificados Digitales

El protocolo HTTPS encripta la comunicación para que quien capture tramas de ella no pueda ver los contenidos. Los navegadores web actuales basan el uso de HTTPS en el conocimiento de Autoridades Certificadoras que emiten Certificados Digitales para asegurar

que el servidor al que nos conectamos es quien dice ser. Estas Autoridades Certificadoras son o agentes dedicados a ello específicamente o empresas como Microsoft.

El uso de HTTPS se basa en la confianza que nos proporcionen las entidades que emiten los certificados. Cuando usamos un navegador web determinado, la empresa que lo ha desarrollado ya ha introducido en él determinadas Autoridades Certificadoras que considera de confianza. Estas entidades se pueden consultar en el propio navegador y podemos modificarlas. En la imagen puedes ver parte de la lista incluida en Firefox.

Investiga en los diferentes navegadores que tengas instalados cómo se consulta la lista de Autoridades Certificadoras en las que se confía.

En la captura de imagen se pueden ver otras pestañas para el administrador de certificados. ¿Para qué sirve cada una de ellas?

Existen muchos servidores que usan certificados no emitidos por estas autoridades. En esos casos depende del usuario el aceptarlos o no. El navegador muestra una advertencia cuando vamos a conectarnos y tendremos que decidir si queremos seguir o preferimos no conectarnos con el destino.

En la imagen podemos ver un ejemplo con varias partes:

1. Si no confiamos en el destino debemos pulsar el botón para salir.
2. Podemos obtener más información. En este caso vemos que la entidad es gov.es por lo que podemos decidir confiar en él.
3. Si queremos confiar debemos leer los riesgos que conlleva.
4. Podemos añadir una excepción de seguridad para confiar en este sitio a partir de ahora.

¿Qué es la firma digital y cuál es su relación con los certificados digitales?

Un **Certificado Digital** es un documento electrónico que enlaza una clave pública con una Firma Digital e información personal sobre la persona u organización que quiere usar la clave pública. Sirve para asegurar que esa clave pertenece a dicha persona u organización. La información personal que se adjunta suele ser el nombre, la dirección, el correo electrónico, etc. La Firma Digital suele ser la de una Entidad Certificadora reconocida para que los clientes puedan confiar en que la Clave Pública y la información personal corresponden a la misma persona/organización. Este tipo de certificados se conocen como **Certificados Raíz** ya que se encuentran en el nivel más alto de un árbol de certificados. El usuario debe fiarse de la entidad emisora del Certificado Raíz ya que es el que asegura la autenticidad de todos los certificados emitidos por ella. La lista de certificados que se incluyen por defecto en un navegador web corresponden a esta categoría y el usuario confía en los desarrolladores del navegador para que se aseguren de que esos certificados son de confianza.

¡Como se puede ver la seguridad en Internet se basa en muchos niveles de confianza! Nadie debe asustarse. Las autoridades que incluyen los principales navegadores están verificadas y con revisadas con regularidad.

En esta [página de la Wikipedia](#) se puede obtener mucha más información sobre los Certificados Digitales.

En [esta imagen](#) se puede ver el uso de certificados digitales.

Los **Servidores de Certificados** son los que se encargan de validar o certificar las claves.

En la actualidad se utilizan varios formatos de Certificados Digitales en Internet. El más extendido es el X.509

Lee [esta página](#) y comenta en clase los puntos que consideres más interesantes sobre la información obtenida.

En nuestro navegador podemos consultar los detalles de un certificado. Si nos ponemos sobre el certificado determinado y pulsamos "Ver" suele aparecer una opción "Detalles" en la que podemos consultar datos como el algoritmo y el valor de la firma.

Obtener un certificado digital

Para poder utilizar HTTPS en nuestro servidor es necesario disponer de un Certificado Digital. Podemos obtener un Certificado Digital de una Autoridad Certificadora (*certificate authority, CA*) o crear nuestra propia Autoridad Certificadora y generar nuestros Certificados Digitales.

En España, podemos obtener Certificados Digitales a través de la [Fábrica Nacional de Moneda y Timbre](#) o mediante empresas como [Verisign](#).

Para obtener un Certificado de una Autoridad Certificadora generalmente hay que demostrar que somos quienes decimos ser (cómo lo establece cada autoridad) y generar una petición para nuestro servidor (*certificate signing request, CSR*) que se debe enviar a la autoridad. Cuando nuestra petición haya sido aceptada, ya podemos instalar el certificado en nuestro servidor.

SSL/TSL

Secure Socket Layer (SSL) fue desarrollado por Netscape en los años 90. Han existido tres versiones de SSL pero actualmente se utiliza la 3.0 aunque deberíamos dejar de hacerlo ya que es un [protocolo obsoleto con vulnerabilidades conocidas](#). Ha derivado en otro protocolo *Transport Layer Security* (TLS) que también tiene tres versiones, la 1.0, la 1.1 y la 1.2. La mayoría de los navegadores actuales usan la versión 1.0

Como la hemos comentado con antelación, HTTPS es una implementación de HTTP sobre SSL o TLS en el servidor.

Este protocolo se coloca sobre la capa de transporte donde los dos protocolos más típicos son TCP y UDP.

Es un protocolo que se utiliza para asegurar **confidencialidad, autenticidad, integridad y no repudio** entre el cliente y el servidor.

Investiga y comenta los cuatro conceptos del párrafo anterior.

Existen dos modos: Uno en el que solo el servidor demuestra su identidad y otro en el que tanto el cliente como el servidor usan Certificados Digitales.

Las aplicaciones del uso de SSL/TSL son múltiples e incluyen la creación de redes privadas virtuales (VPN), el uso en comercio electrónico y en el correo electrónico, etc.

HTTPS en Apache

Apache utiliza un módulo específico basado en un proyecto que se llama [OpenSSL](#); a pesar del nombre [implementa también TLS](#). Para utilizar HTTPS en Apache es necesario que el módulo [mod_ssl](#) esté activo.

```
sudo a2enmod ssl
service apache2 restart
```

Ahora el servidor debería estar escuchando tanto el puerto 80 (http) como en el 443 (https). Si miramos el archivo de configuración de puertos

```
gedit /etc/apache2/ports.conf
```

en el que vemos los diferentes archivos y que podríamos usar [GNUTLS](#).

```
Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

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

veremos que si el módulo *mod_ssl* está activo se añade la orden

```
NameVirtualHost *:80
Listen 80

<IfModule mod_ssl.c>
    # If you add NameVirtualHost *:443 here, you will
    also have to change
    # the VirtualHost statement in /etc/apache2/sites-
    available/default-ssl
    # to <VirtualHost *:443>
    # Server Name Indication for SSL named virtual
    hosts is currently not
    # supported by MSIE on Windows XP.
    Listen 443
</IfModule>
```

Hemos visto que esta distribución de Apache viene con dos sitios por defecto. El que no hemos usado se llama *Default-SSL* precisamente. Si lo activamos tendríamos ya un sitio con dicha configuración que escucharía por conexión segura.

```
sudo a2ensite default-ssl
service apache2 reload
```

Ahora tendríamos disponible la posibilidad de conectarnos de forma segura a ambos servidores. Si no especificamos el protocolo o usamos http se conectará de la forma estándar. Sin embargo, si nos conectamos mediante https nos muestra una excepción de seguridad como la que vimos antes. Esto es debido a que al instalar Apache se crea un certificado autofirmado para el sitio por defecto.

Si abrimos el archivo de configuración del sitio con ssl podemos ver cómo está configurado.

```
#    A self-signed (snakeoil) certificate can be
created by installing
#    the ssl-cert package. See
#    /usr/share/doc/apache2.2-
common/README.Debian.gz for more info.
#    If both key and certificate are stored in the
same file, only the
#    SSLCertificateFile directive is needed.
```

```
SSLCertificateFile      /etc/ssl/certs/ssl-cert-  
snakeoil.pem  
SSLCertificateKeyFile  /etc/ssl/private/ssl-cert-  
snakeoil.key
```

Tras realizar los pasos anteriores, prueba a interceptar tramas de conexiones http y https. ¿Qué diferencias ves? Guarda en un archivo una trama de cada tipo para su discusión en clase.

Si te fijas, los dos sitios por defecto tienen configurado el mismo directorio como raíz para los documentos. Qué pasaría si cambias el directorio raíz de uno de los dos. Pruébalo.

Es una mala opción dejar abierta la posibilidad de acceder al mismo sitio mediante una conexión segura y una insegura. Por ello si habilitamos un sitio con HTTPS no deberíamos tener activo el equivalente con HTTP como sucede con los dos por defecto. **Lo correcto sería deshabilitar el default.**

Si deshabilitamos el sitio por defecto podemos ver que ahora, si no especificamos el protocolo, utiliza HTTPS por defecto.

```
sudo a2dissite default  
service apache2 reload
```

Estudia el resto del archivo de configuración del sitio por defecto con ssl y explica qué hacen el resto de directivas. Los enlaces a continuación contienen toda la información necesaria.

<https://httpd.apache.org/docs/2.4/en/ssl/>

https://httpd.apache.org/docs/2.4/mod/mod_ssl.html

Creando un sitio virtual con HTTPS

Ahora que hemos probado el sitio seguro por defecto, vamos a crear uno desde cero. Voy a dejar habilitado el sitio por defecto para hacer un ejemplo con un sitio con HTTP y otro con HTTPS.

```
sudo a2dissite default-ssl
sudo a2ensite default
service apache2 reload
```

Como ya hemos visto antes, para poder usar SSL en Apache es necesario tener un certificado. El que se instala para el sitio de ejemplo ya no es válido y tenemos que conseguir uno. Podemos adquirir uno de una CA o crear uno autofirmado. Por motivos evidentes nosotros usaremos esta última opción.

Para obtener un certificado es necesario generar una clave privada y para ello necesitamos un nombre de dominio así que lo primero será **configurar el DNS** correctamente. En mi caso voy a llamarlo www.con-ssl.es y habrá que configurarlo como un host virtual por nombre.

Para generar la clave se utiliza el [comando genrsa](#)

```
openssl genrsa -out clavepru.key 2048
```

Se puede generar una contraseña para la clave, pero si se va a utilizar para crear un certificado no es buena idea porque cada vez que el servidor web necesite acceder a la clave habrá que introducir la contraseña. Si no nos importa introducir la clave cada vez que se reinicie el servidor, usaremos la opción

```
openssl genrsa -des3 -out clavepru.key 2048
```

Actualmente se recomiendan longitudes mínimas de clave de 2048 bits.

Lo siguiente será generar una petición para nuestro certificado. Esta petición es la que deberíamos enviar a la CA para obtener un certificado firmado por ellos. Luego esperaríamos a que lo firmaran y nos enviaran el certificado para instalarlo. Nosotros usaremos uno autofirmado.

```
openssl req -new -key clavepru.key -out peticionpru.csr
```

Cuando ejecutamos este comando nos va pidiendo información de la empresa para la que sea el certificado. La vamos rellenando hasta terminar. Los campos que terminan en [] no son obligatorios. Si hubiéramos generado la clave con contraseña nos la pediría antes de rellenar la información.

```
    You are about to be asked to enter information that
will be incorporated
    into your certificate request.
    What you are about to enter is what is called a
Distinguished Name or a DN.
    There are quite a few fields but you can leave some
blank
    For some fields there will be a default value,
    If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Madrid
Locality Name (eg, city) []:Madrid
Organization Name (eg, company) [Internet Widgits Pty
Ltd]:SergioCuesta
Organizational Unit Name (eg, section) []:SC
Common Name (e.g. server FQDN or YOUR name) []:Sergio
Cuesta
Email Address []:una@con-ssl.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Para obtener un certificado autofirmado usaremos el comando

```
openssl x509 -req -days 365 -in peticionpru.csr  
-signkey clavepru.key -out certificadopru.crt
```

Lo que indica que usará el formato X.509 y tendrá una validez de un año. Si todo ha ido bien deberíamos ver algo como lo siguiente

```
Signature ok  
subject=/C=ES/ST=Madrid/L=Madrid/O=SergioCuesta/OU=SC/C  
N=Sergio Cuesta/emailAddress=una@con-ssl.es  
Getting Private key
```

En muchos sitios verás que en lugar de *crt* se crean archivos con la extensión *pem*. En teoría la diferencia es que *crt* solo contiene el certificado mientras que *pem* contiene tanto el certificado como la clave pero en la práctica esto se ignora y da igual usar una que otra.

Ten en cuenta que todos los archivos que se han creado en los pasos anteriores se han generado en el directorio en el que nos encontráramos por lo que hay que moverlos a los sitios adecuados. La petición no es necesaria.

```
sudo mv clavepru.key /etc/ssl/private/  
sudo mv certificadopru.crt /etc/ssl/certs/
```

Crearemos un directorio para el contenido del sitio seguro

```
mkdir /var/www/con-ssl/
```

Procedemos a crear el sitio virtual por nombre

```
<IfModule mod_ssl.c>  
  
NameVirtualHost 192.168.1.36:443  
  
<VirtualHost 192.168.1.36:443>  
    ServerName con-ssl.es  
    ServerAlias www.con-ssl.es  
    ServerAdmin alguien@con-ssl.es  
    DocumentRoot /var/www/con-ssl
```

```
<Directory /var/www/con-ssl>
    DirectoryIndex index.html
    Options -Indexes
    AllowOverride None
    Require all granted
</Directory>

ErrorLog ${APACHE_LOG_DIR}/error_con_ssl.log
LogLevel warn
CustomLog ${APACHE_LOG_DIR}/con_ssl_access.log
combined

# Esta es la parte de SSL
SSLEngine on

SSLCertificateFile
/etc/ssl/certs/certificadopru.crt

SSLCertificateKeyFile /etc/ssl/private/clavepru.key

# Recuerda que lo siguiente es para mantener la
compatibilidad con ciertas
# versiones de Microsoft Internet Explorer
BrowserMatch "MSIE [2-6]" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0

BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>

</IfModule>
```

Habilitamos el sitio y recargamos Apache

```
sudo a2ensite con-ssl.es
service apache2 reload
```

Recuerda que el nombre que tenemos asociado en el DNS va a la dirección IP de la máquina así que **tendremos que usar** <https://www.con.ssl.es> para acceder al sitio seguro. Si usamos HTTP va al sitio por defecto. Para evitar esto deberíamos añadir otra entrada al DNS para que vaya al sitio sin SSL (puerto 80) y modificar la configuración del host virtual para que responda a las peticiones a ese otro nombre de dominio en lugar de a todo (*)

Si los hemos adquirido los certificados a través de una CA de confianza dejaría de aparecer el aviso cuando un cliente se conecta. Evidentemente para un sitio profesional es más que recomendable.

Para casos en los que estemos involucrados en el despliegue en una empresa muy grande o por ejemplo una universidad, puede interesarnos crear **nuestra propia CA** para uso propio. También es posible crear un sitio en el que **los clientes tengan que acceder mediante un certificado propio** como por ejemplo en la Agencia Tributaria. Ambas cosas se pueden hacer usando *OpenSSL* pero escapa totalmente al contenido del curso.

Desde el descubrimiento del **ataque Poodle** se ha hecho más importante [configurar correctamente](#) la versión que usemos de encriptación.

Crea dos sitios diferentes, uno con HTTPS y el otro sin él. Configura todo correctamente para que al ir a un sitio o a otro use el protocolo adecuado sin necesidad de que el usuario lo especifique en la barra de direcciones del navegador web.

Despliegue de aplicaciones sobre servidores web y Empaquetado de aplicaciones web.

Estos dos conceptos no los vamos a estudiar aquí. Las aplicaciones en servidores web suelen utilizar al menos una base de datos y las que se despliegan en Apache se están utilizando constantemente en el módulo de “Desarrollo Web en Entorno Servidor”. El empaquetado de aplicaciones es un concepto que está más relacionado con Tomcat por lo que lo veremos en el tema siguiente.

En los casos en los que no instalemos nuestro propio servidor sino que contratemos un hosting, debemos considerar que tenga las características que necesitemos (por ejemplo

que permita o proporcione el uso de MySQL y PHP). Generalmente accedemos a nuestra estructura de carpetas mediante FTP y a la base de datos mediante algún cliente.