Actividad 4 – Configuración avanzada de servidores web (Protocolo HTTPS)

1. Introducción

Todas las comunicaciones HTTP que se han visto hasta ahora viajan en claro por la red, cualquiera que utilice un software "sniffer" puede ver todo el tráfico. Actualmente eso supone muchos problemas a la hora de implementar aplicaciones que requieren una confidencialidad de los datos. En este punto entra en juego la criptografía que permite cifrar los datos enviados para que nadie pueda ver el contenido real, a no ser que disponga de la clave de cifrado. Como ya se ha visto, el protocolo TLS/SSL habilita las funcionalidades de confidencialidad, integridad y autenticación sobre el protocolo HTTP, utilizando mecanismos de criptografía tanto simétrica como de clave pública.

2. Autoridades certificadoras: Certificados de servidor – Actividad 1 (1 punto)

Contesta razonadamente a las siguientes preguntas:

1.1. ¿Qué es una Autoridad de Certificación (CA)? ¿Qué función desarrolla durante la fase de "handshaking" del protocolo **https**? Explícalo con tus palabras

1.2. Visita las páginas web siguientes y consulta la información del certificado de servidor que se utiliza. Completa el siguiente cuadro:

Web	Autoridad de Certificación	Validez	Algoritmo de firmado
https://www.digicert.com/			
https://www.bbva.es/			
https://www.pccoste.es/			
https://www.adsalsa.com/			

1.3. Investiga qué significado tienen los siguientes símbolos que aparecen al acceder a páginas web, mediante el protocolo https y Firefox:

	

	
	

1.4. ¿Qué diferencia existe entre la criptografía simétrica y la asimétrica?

1.5. ¿Qué significa el concepto de intercambio de claves entre cliente y servidor en el contexto de HTTPS?

3 . Configuración SSL del servidor apache2: certificados de servidor auto-firmados

Para configurar un host virtual en Apache2 que sirva las peticiones haciendo uso del protocolo SSL, necesitaremos hacer uso del módulo `mod_ssl`. Este módulo es una interfaz que habilita el uso de las librerías openssl (implementación open-source del protocolo SSL), con lo que primeramente se han de instalar las librerías y herramientas openssl:

```
# apt-get install openssl libssl-dev
```

Una vez se han instalado las herramientas openssl tendremos que activar el módulo `mod_ssl` del servidor para integrar la nueva funcionalidad en apache2.

```
# sudo a2enmod ssl
```

Como siempre, la documentación inicial nos puede proporcionar más información sobre el módulo que vamos a gastar: http://httpd.apache.org/docs/current/ssl/

3.1 Creación del certificado de servidor (openssl)

Para poder poner en marcha un servidor web que atienda las peticiones con SSL se necesita un certificado de servidor firmado por una CA. Como no disponemos de ninguna CA real, haremos pruebas con un certificado autofirmado.

Podemos crear la clave SSL y los archivos del certificado según el siguiente orden:

```
# openssl req -new -x509 -nodes -days 365 -newkey rsa:2048 -out server.crt -keyout server.key
```

A continuación se describe el comando utilizado:

- **openssl:** es la herramienta de línea de comandos para crear y administrar certificados, claves y otros archivos OpenSSL.
- **req -new -x509:** especifica que se requiere el estándar “X.509” para la administración de claves y certificados.
- **nodes:** indica a OpenSSL que omita la opción para proteger nuestro certificado con una frase de contraseña. Necesitamos que Apache pueda leer el archivo, sin intervención del usuario.
- **-days 365:** esta opción es establece el tiempo durante el cual el certificado será válido. En este caso, se configura para un año. Muchos navegadores modernos rechazarán cualquier certificado válido para más de un año.
- **newkey rsa:2048:** especifica que deseamos generar un nuevo certificado y una nueva clave al mismo tiempo. No creamos la clave que se requiere para firmar el certificado en un paso anterior, por lo que hemos de crearla junto al certificado. La parte **rsa:2048** indica que cree una clave **RSA con 2048 bits** de extensión.
- **keyout**: esta línea a **OpenSSL** dónde colocar el archivo de clave privada generado que estamos creando.
- **-out:** indica a **OpenSSL** dónde colocar el certificado que creamos.

Una vez ejecutado el comando, nos preguntará por la información a incluir en el certificado sobre nuestra empresa y el sitio web que queremos publicar.

```
Country Name (2 letter code) [XX]:ES
State or Province Name (full name) []: Alicante
Locality Name (eg, city) [Default City]: Elche
Organization Name (eg, company) [Default Company Ltd]: Example Inc
Organizational Unit Name (eg, section) []: Example Dept
Common Name (eg, your name or your server's hostname) []:
intranet.ddaw.es
Email Address []: webmaster@intranet.ddaw.es
```

Es muy importante responder correctamente a la pregunta de **Common Name**

dado que tendremos que indicar el nombre del dominio o la IP para la que estamos creando el certificado. El siguiente enlace nos proporciona la información oficial.

Guarda la clave privada y los certificados en `/etc/ssl/nombre-dominio-vhost` donde `nombre-dominio-vhost` para el vhost backoffice.ddaw.es sería `/etc/ssl/es.ddaw.backoffice`

2.2 Configuración del módulo SSL

Una vez disponemos de un certificado de servidor firmado por una CA (en este caso la nuestra), se puede configurar el servidor web para utilizar el protocolo SSL (mod_ssl). El objetivo es conseguir acceder a nuestro servidor mediante SSL, por eso es necesario crear un nuevo VirtualHost que escuche en el puerto 443 (puerto SSL por defecto). Los pasos son los siguientes:

El **servidor ubuntu** proporciona una plantilla para la creación de vhost con ssl en el directorio `/etc/sites_availables/default-ssl.conf` donde aparecen las principales directivas.

Los pasos a realizar son *(comprueba que las directivas no existan ya por defecto)*:

1. Es necesario añadir la directiva ***Listen*** para escuchar el nuevo puerto 443, en el fichero principal `ports.conf`.

```
<IfModule ssl_module>
    Listen 443
</IfModule>
```

2. En el **VirtualHost** correspondiente utilizaremos las siguientes directivas para activar ssl**.**

```
<VirtualHost:443>
        SSLEngine On
        SSLCertificateFile /path/to/server.crt
        SSLCertificateKeyFile path/to/server.key
```

```
</VirtualHost>
```

NOTA: No olvides activar el nuevo virtual host mediante el comando a2ensite.

3. Reiniciar el servidor y comprobar el funcionamiento al acceder al nuevo virtual host (*https*).

Actividad 2 (2 puntos)

• Configura el vhost que atiende al dominio **backoffice.ddaw.es** configurado en la práctica 1 de la unidad para que solo atienda a peticiones https. Para eso tendrás que:

1. Crear un certificado autofirmado y la clave privada correspondiente.
2. Crear un nuevo vhost que atienda las peticiones https para el dominio correspondiente.
3. Crear una redirección permanente del tráfico del vhost, de http a https. En este enlace puedes encontrar información de cómo hacerlo.

Documenta con capturas de pantalla y explicaciones los pasos que has dado.

4. Bibliografía / Webgrafía

- Documentación oficial apache server. http://httpd.apache.org/docs/current/ssl/ssl_faq.html#selfcert. **Apache Software Foundation.**
- Glass E. **“**Como crear un certificado auto-firmado en ubuntu 20.04” https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-20-04-es. **Digital Ocean**
- Jay laCroix. "Mastering Ubuntu Server. Second edition”. (2018). **Packt Publishing**