Configuración HTTPS de un servidor Apache

Para aumentar la seguridad, podemos usar el protocolo HTTPS (*HyperText Transfer Protocol Secure*), que se basa en una capa de software protegida, SSL (*Secure Socket Layer*) o TLS (*Transport Layer Secure*, sucesor de SSL), para la comunicación entre el cliente y el servidor Apache. El protocolo de seguridad toma en cuenta la autenticación de los participantes a través de los certificados, el cifrado y el control de la integridad de los datos intercambiados.

TLS ha reemplazado a partir de 1999 la versión 3 de SSL, abandonada a causa de problemas de seguridad. Desde entonces usamos los términos SSL, SSL/TLS o TLS para designar el protocolo TLS. A partir de ahora usaremos el término SSL, conforme a la terminología usada en la certificación LPIC-2.

El protocolo HTTPS usa el puerto bien conocido 443. Tiene tendencia a convertirse en el estándar de Internet y a reemplazar a HTTP.

1. Criptografía y certificados

El protocolo SSL se basa en certificados para autenticar a los participantes en la comunicación. El servidor envía su certificado al cliente para que este se asegure de su identidad. El cliente puede, de manera opcional, poseer también un certificado que transmitirá al servidor para declararle su identidad.

El protocolo asegura el cifrado y descifrado de los datos intercambiados, con un par de claves, pública y privada, según la técnica de criptografía asimétrica.

En el marco de la certificación LPIC-2 se pide un conocimiento general de estos conceptos.

a. Criptografía simétrica

Los algoritmos de cifrado simétrico usan una clave única que permite cifrar y descifrar datos.

Presentan la ventaja de necesitar pocos recursos y de ser muy rápidos. Permiten, por lo tanto, intercambiar eficazmente grandes cantidades de datos. Sin embargo, presentan una dificultad: los participantes tienen que conocer la clave de cifrado/descifrado. Por lo tanto, todas las entidades que posean la clave podrán descifrar los mensajes, incluso aquellas para las que el mensaje no ha sido destinado.

Para intercambiar la clave de manera segura, la ciframos usando un algoritmo asimétrico, que es más seguro.

b. Criptografía asimétrica

Los algoritmos de cifrado simétrico reposan en un par de claves: una pública, que permite cifrar la información; otra privada, que permite descifrar la información que ha sido cifrada con la clave pública.

La clave privada solo la conoce el propietario. La clave pública se transmite a las entidades que tienen que cifrar mensajes. La criptografía asimétrica necesita muchos recursos del procesador, por lo tanto, se utiliza esencialmente para cifrar pequeñas cantidades de datos, en particular el intercambio de claves. La clave pública permite cifrar una clave, y la clave privada se usa para firmar digitalmente el mensaje.

El eslabón más débil reside en el envío de la clave pública por el propietario del par de claves. En efecto, si el emisor de la clave pública ha usurpado una identidad, podrá descifrar mensajes que no le han sido realmente destinados. Es la razón por la que el protocolo SSL exige que el emisor de la clave pública pruebe su identidad presentando un certificado de identidad, bajo la forma de un certificado digital.

c. Los certificados digitales X.509

Los certificados digitales que siguen la norma X.509, definida por la Unión Internacional de las Telecomunicaciones (UIT), tiene por objetivo garantizar la relación entre una identidad (nombre, dirección IP, etc.) y una clave pública. Los establece un organismo "tercero de confianza", llamado autoridad de certificación (CA, *Certificate Authority*). Estos organismos

pueden ser públicos o privados. En general, la obtención de un certificado digital mediante una autoridad de certificación es de pago, pero existen CA sin ánimo de lucro (Let's Encrypt, por ejemplo).

También es posible que una organización emita sus propios certificados (certificado autofirmado). En este caso, las entidades que reciben el certificado deben reconocer al emisor del certificado como una autoridad de certificación.

2. Funcionamiento de una conexión HTTPS

El cliente HTTPS (un navegador de Internet) solicita una conexión TCP en el puerto bien conocido 443 del servidor HTTPS. El cliente y el servidor usan el protocolo SSL (en realidad TLS) para establecer la conexión segura por donde pasará el intercambio de datos, según las etapas siguientes:

- El cliente envía una solicitud de conexión SSL.
- EL servidor envía al cliente su certificado X.509, que contiene su clave pública, su información de identidad (nombre DNS, dirección postal, correo electrónico del administrador, etc.), la firma digital mediante la autoridad de certificación y su clave pública.
- El cliente comprueba la firma digital del certificado, usando las claves públicas de las autoridades de certificación que reconoce. Si alguna de las claves permite autenticar el certificado, el cliente deduce el nombre de la autoridad de certificación. El cliente le envía una solicitud de comprobación de la validez del certificado para asegurarse de que el certificado no ha sido revocado. Si ninguna clave funciona, el cliente intenta comprobar la identidad del servidor usando la clave pública enviada por el mismo servidor, si la clave funciona, se trata de un certificado autofirmado. El cliente mostrará entonces un mensaje para informar al usuario de que el certificado no está garantizado por una autoridad de certificación.
- Si ninguna clave funciona, el certificado no es válido y el cliente abandona la conexión.
- Algunos servidores HTTPS exigen un certificado al cliente. En este caso, el cliente envía su certificado X.509 al servidor, el cual lo valida según el

procedimiento descrito anteriormente.

- Si el certificado es válido, el cliente genera una clave de cifrado simétrica, la clave de sesión, la cifra con la clave pública del servidor HTTPS y se la envía. El servidor descifra la clave de sesión con su clave privada.
- La conexión SSL/TLS se ha establecido, el intercambio de datos se puede hacer, cifrados con la clave de sesión.
- Cuando la conexión se termina, el servidor HTTPS procede a la revocación de la clave de sesión, para invalidarla.

3. Configuración SSL de un servidor Apache

Para poder usar HTTPS, el servidor Apache debe tener un certificado X.509, expedido por una autoridad de certificación o autofirmado. Después tendrá que instalar el certificado y configurar el servidor HTTP Apache para que pueda funcionar en modo HTTPS.

Para las comprobaciones, utilizaremos generalmente un certificado autofirmado, generado por las herramientas de la biblioteca `openssl`, proporcionada por el proyecto OpenSSL.

Para obtener una certificación de una autoridad de certificación, hay que proporcionarle la clave pública de un par de claves y una solicitud de firma de certificado, en formato CSR (*Certificate Signing Request*), que tienen que contener los campos siguientes:

`Common Name (CN=)`	El nombre completo (FQDN).
`Organisation (O=)`	Nombre de la organización.
`Department/Organizational Unit (OU=)`	Departamento o servicio de la organización (dirección u otro servicio).
`Locality (L=)`	Ciudad.
`State/Province (S=)`	Región o estado.
`Country (C=)`	Código ISO de dos letras del país.
`E-mail`	Dirección de correo electrónico del responsable del certificado.

a. Generación de un certificado autofirmado

El comando `openssl` del paquete de software `openssl` permite crear un certificado X.509 autofirmado.

La sintaxis habitual es la siguiente:

```
openssl req -x509 -nodes -newkey rsa:NúmBits -keyout ArchivoClave  -out
ArchivoCert
```

Donde:

`req`	Subcomando de solicitud de creación.
`-x509`	Creación de un certificado autofirmado.
`-nodes`	La clave no estará protegida por contraseña.
`-newkey rsa:NœmBits`	Clave asimétrica `RSA`, `NœmBits` : tamaño de la clave en bits (>= `2048`).
`-keyout ArchivoClave`	`ArchivoClave` : camino de acceso del archivo de clave privada.
`-out ArchivoCert`	`ArchivoCert` : camino de acceso del archivo certificado.

El comando genera un certificado y la clave privada correspondiente, solicitando la introducción de los datos necesarios en los diferentes campos de una solicitud de firma de certificado (CSR, *Certificate Signing Request*).

El campo `Common Name` debe presentar obligatoriamente el nombre completo DNS que será especificado en la URL de acceso al servidor. En caso contrario, el navegador del cliente mostrará una alerta de seguridad, después de haber comprobado el certificado del servidor.

Esta obligación tiene un impacto en el uso de HTTPS en los hosts virtuales, porque sus URL de acceso no corresponden al nombre DNS del servidor, entonces el control del certificado por el cliente falla porque el nombre de host no corresponde al especificado en la URL.

Ejemplo

Generación de un certificado autofirmado para un servidor HTTP Apache en une distribución Debian 10.

```
openssl req -x509 -nodes -newkey rsa:2048 -keyout certificado.clave -out
certificado.crt
Generating a RSA private key
.....++                    +++
...+++++
writing new private key to 'certificado.clave'
-----
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]:ANDALUCIA
Locality Name (eg, city) []:Almería
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Mi Empresa
Organizational Unit Name (eg, section) []:Informática
Common Name (e.g. server FQDN or YOUR name) []:debian10.midns.es
Email Address []:pba@debian10.midns.es
```

Contenido del archivo de clave privada:

```
cat certificado.clave
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAMJPqAOC+hJslKX9
hLg3Wk2r03F35hTEIpkB8I0LeHQ/A75PiMMBRRtJC9vGdm6xmo5YJPlhGNaHlfH6
JdMGf6CXP+YueQtfdCk5XRWsKr+vcF7KUwUmzD1xsZYEhJAzvh4A79QVUgGLXywd
MeEmQz6S4LKCLHQysZi84zfYHdI5AgMBAAECgYASkDOX6kpjujXVScwFIVyiMPkp
TPARKc+4WGQXxXHDgUWlR8nj/1oPykG0xFgKKTN/x9H4dRs/W4KqtGAqseM9VGBe
ced7LmF5JO9bUUskGKWSI3bCJRKI6/ptfWvGJa6t6OAblo/5fVT6Un7yvkbFl6dL
U72qQAZlSqo5gUhmBQJBAP02gAs5ECOIyNKrbY6qgUcm8lQ8NZBm13quYPV6zwo2
cveLUUxdgiTyEjA86H+rwDuBllYQ7DCTpEbEXshvYl8CQQDEcy8Ca3uWVkb+NF2M
vRG5tWB6y6I/492AIv+/Nr/A3RmTRDC8dhHZOH4U+ankJelzWGiXW3CNHCSVitOF
```

```
R4JnAkAAsA80+Ji0oAB7of+QLaJgQRjTZb53f/AB40tcH1NEodU6GuTDolViKE4o
V2ICRsMk5jrjpPQfRadikdH4FWVdAkEAidnZzvTlUGVSg+bzDmNOIOapwQFL5VFc
iSIGHIOqoweDfteG63hepfMCZm2bTcFcw9fx7OP5Ysuw0E8OhqPbWQJAHBqW+N6a
mJLp0JziykuFqtEBOe/XNtX7ENpCl0DVQRU/0fKGusEjLFt/hJrW1dQmcv+HDj5l
tAS+zjUoVOGFYA==
-----END PRIVATE KEY-----
```

Contenido del archivo certificado:

cat certificado.crt

```
-----BEGIN CERTIFICATE-----
MIIDBDCCAm2gAwIBAgIUQ1eBBL5LK1eu7ZjoWuTFtSPqGScwDQYJKoZIhvcNAQEL
BQAwgZMxCzAJBgNVBAYTAkZSMQwwCgYDVQQIDANJREYxDjAMBgNVBAcMBVBhcmlz
MRAwDgYDVQQKDAdNeSBjb21wMRAwDgYDVQQLDAdzZWN0aW9uMRswGQYDVQQDDBJk
ZWJpYW4xMC5tb25kbnMuZnIxJTAjBgkqhkiG9w0BCQEWFnBiYUBkZWJpYW4xMC5t
b25kbnMuZnIwHhcNMjAwNTE4MDkyNzA5WhcNMjAwNjE3MDkyNzA5WjCBkzELMAkG
A1UEBhMCRlIxDDAKBgNVBAgMA0lERjEOMAwGA1UEBwwFUGFyaXMxEDAOBgNVBAoM
B015IGNvbXAxEDAOBgNVBAsMB3NlY3Rpb24xGzAZBgNVBAMMEmRlYmlhbjEwLm1v
bmRucy5mcjElMCMGCSqGSIb3DQEJARYWcGJhQGRlYmlhbjEwLm1vbmRucy5mcjCB
nzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAwk+oA4L6EmyUpf2EuDdaTavTcXfm
FMQimQHwjQt4dD8Dvk+IwwFFG0kL28Z2brGajlgk+WEY1oeV8fol0wZ/oJc/5i55
C190KTldFawqv69wXspTBSbMPXGxlgSEkDO+HgDv1BVSAYtfLB0x4SZDPpLgsoIs
dDKxmLzjN9gd0jkCAwEAAaNTMFEwHQYDVR0OBBYEFEDRBSw/4Vs75JNbyUYWdfqp
M90PMB8GA1UdIwQYMBaAFEDRBSw/4Vs75JNbyUYWdfqpM90PMA8GA1UdEwEB/wQF
MAMBAf8wDQYJKoZIhvcNAQELBQADgYEAtqW0Ux7fLwEFdeawW0tzuZExkc5NIPYE
YVU3icWZ9/7qkxHUV60yFZPYosy+QR1VB+Zs8LvL3PMOAdNEvczgZZzORk6sNJpW
JSibtlfU1yQzJ1nORZjYyuEsxKMIfzZ5dqdRWIvksq6bh5DEXFbTaF8MjdQfVVRt
yX7j3XrKaIk=
-----END CERTIFICATE-----
```

Para generar un certificado autofirmado, también podemos usar los scripts `CA.sh` o `CA.pl`, proporcionados en el paquete de software `OpenSSL`.

b. Carga del módulo SSL

El módulo que gestiona el protocolo SSL es `mod_ssl`. Tiene que estar configurado para ser cargado en el archivo de configuración del servidor HTTP Apache.

Ejemplo

Configuración del módulo para un servidor HTTP Apache2 en una distribución Debian 10.

Comprobamos que el módulo está instalado:

```
ls /lib/apache2/modules/*ssl*
/lib/apache2/modules/mod_ssl.so
find /etc/apache2 -name '*ssl*'
/etc/apache2/sites-available/default-ssl.conf
/etc/apache2/mods-available/ssl.conf
/etc/apache2/mods-available/ssl.load
cat /etc/apache2/mods-available/ssl.load
# Depends: setenvif mime socache_shmcb
LoadModule ssl_module /usr/lib/apache2/modules/mod_ssl.so
```

El módulo está instalado pero no está cargado. Hay que copiar los archivos de configuración en el directorio de los archivos de configuración activos, incluyendo algunos módulos de los que depende:

```
cp /etc/apache2/mods-available/ssl.conf /etc/apache2/mods-enabled
cp /etc/apache2/mods-available/ssl.conf /etc/apache2/mods-enabled
cp /etc/apache2/mods-available/socache_shmcb.load /etc/apache2/mods-enabled
```

Comprobamos la configuración:

```
apache2 -t
Syntax OK
```

Recargamos la configuración y comprobamos la carga del módulo:

```
systemctl reload apache2
apache2 -M | grep ssl
ssl_module (shared)
```

El módulo está cargado.

El módulo `mod_ssl` crea, durante su instalación, el directorio `/etc/pki` y diferentes subdirectorios para almacenar archivos (certificado y clave de certificado por defecto, etc.).

c. Configuración de las claves del servidor

Hay que declarar el certificado y la clave privada en el archivo de configuración del servidor HTTP Apache, usando las directivas siguientes:

```
SSLCertificateFile ArchivoCert
SSLCertificateKeyFile ArchivoClave
```

Donde :

`ArchivoCert`	Camino de acceso al archivo que contiene el certificado.
`ArchivoClave`	Camino de acceso al archivo que contiene la clave privada.

Ejemplo

Continuación de la configuración del ejemplo anterior.

```
vi /etc/apache2/apache2.conf
```

```
[...]
SSLCertificateFile /root/apache/certificado.crt
SSLCertificateKeyFile /root/apache/certificado.clave
[...]
```

Comprobamos la configuración:

```
apache2 -t
Syntax OK
```

Recargamos la configuración:

```
systemctl reload apache2
```

d. Activación del modo HTTPS

El servidor HTTP Apache tiene que escuchar en el puerto bien conocido 443 HTTPS, además de en los otros puertos. También debe tener activado el motor SSL, que gestionará los intercambios SSL/TLS.

Para ello, hay que especificar dos líneas de directivas:

```
Listen 443
SSLEngine on
```

Si el servidor tiene que estar accesible en HTTPS y en HTTP, podemos definir un host virtual asociado al puerto 443, que contenga las directivas necesarias para gestionar el modo HTTPS.

Ejemplo

Continuación de la configuración del ejemplo anterior:

El servidor HTTP Apache2 de una distribución Debian 10 gestiona los puertos en el archivo de configuración `/etc/apache2/ports.conf` *. Si el módulo* `mod_ssl` *está activo, la directiva* `Listen 443` *se configura automáticamente:*

```
vi ports.conf
Listen 80
<IfModule ssl_module>
    Listen 443
</IfModule>
[...]
```

Solamente es necesario añadir la activación del motor SSL:

```
vi /etc/apache2/apache2.conf
[...]
SSLEngine on
[...]
```

Comprobamos la configuración:

```
apache2 -t
Syntax OK
```

Recargamos la configuración:

```
systemctl reload apache2
```

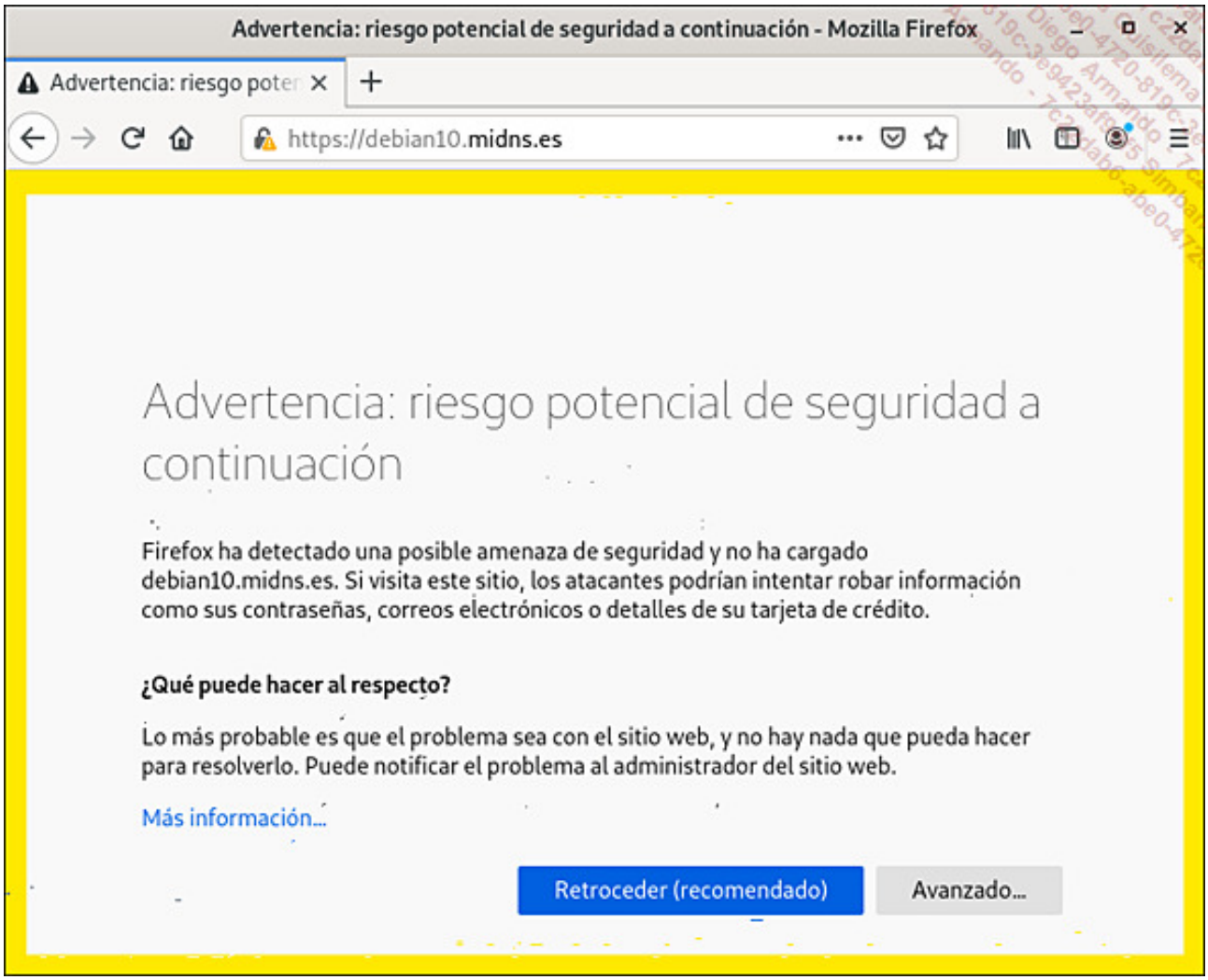
e. Comprobación del servidor HTTPS

Para comprobar el modo HTTPS del servidor HTTP Apache, vamos a usar el navegador de Internet y solicitamos la URL HTTPS del servidor que tengamos que comprobar.

Ejemplo

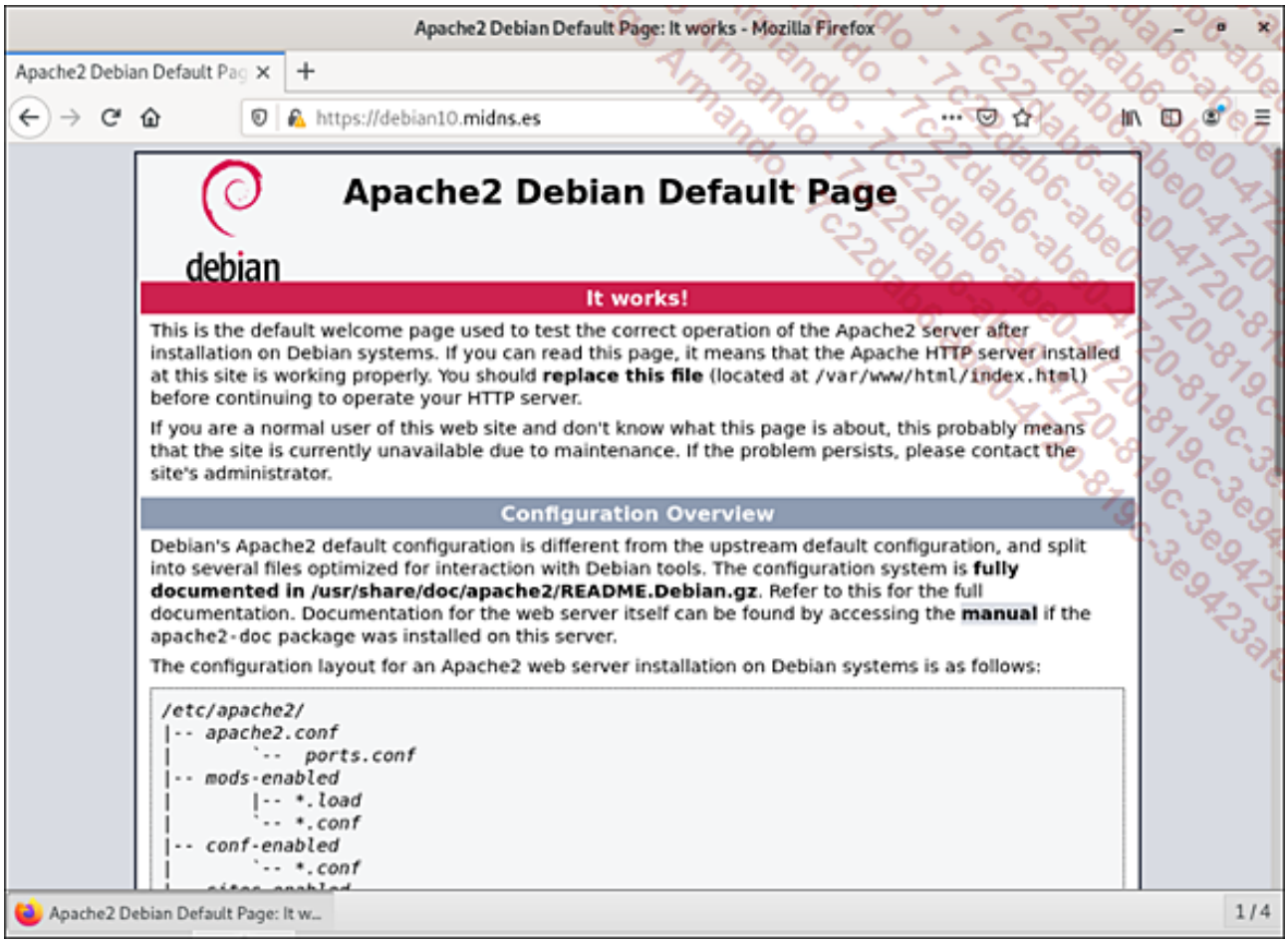
Comprobación del servidor HTTP Apache configurado en los ejemplos anteriores.

Solicitamos la URL https://debian10.midns.es.



El navegador ha validado el certificado, pero como está autofirmado, muestra una alerta y solicita una confirmación.

Después de confirmar la excepción, el navegador muestra la página de inicio del servidor:



f. Autenticación de los clientes por certificado

Si el servidor HTTPS tiene que solicitar la autenticación del cliente por certificado X.509, hay que copiar en el servidor el archivo de certificado de la autoridad que ha firmado el certificado del cliente, y añadir a continuación las directivas siguientes en el archivo de configuración:

```
SSLVerifyClient require
SSLCACertificateFile ArchivoCert-ca
```

Donde:

`ArchivoCert-ca` : archivo que contiene el conjunto de los certificados de las autoridades de certificación que firman los certificados de los clientes.

También podemos usar un archivo por certificado de cliente, estos tendrán que almacenarse en el directorio indicado por la directiva `SSLCACertificatePath` .

g. Uso de SSL con los hosts virtuales

Como el certificado X.509 está asociado a un nombre DNS, HTTPS provoca problemas con los hosts virtuales. Estos últimos están identificados por nombres específicos en la URL, por lo tanto no pueden utilizarse en el certificado X.509 del servidor, porque el nombre no corresponde obligatoriamente con la URL solicitada por el cliente.

Una solución a este problema es la de implementar una extensión del protocolo SSL/TLS, llamada SNI (*Server Name Indication*) y definida en la RFC 4366. Esta extensión le permite al cliente especificar, en su solicitud de certificado al servidor, el nombre DNS asociado al certificado. El servidor puede entonces enviarle el certificado del host virtual solicitado.

En este modelo, cada host virtual accesible en HTTPS tiene que disponer de un certificado X.509 específico.

El servidor HTTP Apache integra esta extensión del protocolo SSL/TLS, si está compilado con una versión reciente de la biblioteca `OpenSSL` (`0.9.8f` o superior).

h. Directivas para reforzar la seguridad de los intercambios del servidor

Incluso en modo HTTPS, un servidor Apache puede presentar elementos de riesgo que pueden comprometer su seguridad. Se aconseja desactivar las antiguas versiones de los protocolos SSL/TLS y de los protocolos de cifrado. También se aconseja limitar la información ofrecida por el servidor, en particular su versión. También puede ser útil personalizar las respuestas del servidor, para utilizarlas en la autenticación.

Distintas directivas permiten controlar estos elementos de seguridad, entre ellas:

`SSLProtocol`	Versiones del protocolo SSL/TLS autorizadas.
`SSLCipherSuite`	Algoritmos de cifrado autorizados.
`ServerTokens`	Campos del encabezado `Server` enviados al cliente.
`ServerSignature`	Pie de página para los documentos generados por el servidor.
`TraceEnable`	Autoriza o no las solicitudes de registro enviadas por el cliente.