

HTTPS en Apache2

Alberto Molina Coballes, José Domingo Muñoz Rodríguez y José Luis Rodríguez Rodríguez.

22 de marzo de 2010

En este documento se describe de forma breve la configuración del protocolo HTTPS en un servidor Apache2 en Debian GNU/Linux (Lenny) utilizando tanto un certificado autofirmado como un certificado emitido por la organización CAcert. Este documento forma parte del curso *Servicios en GNU/Linux. Portal Educativo*, organizado por el CEP de Lora del Río (Sevilla) en 2010.



Usted es libre de copiar, distribuir y modificar este documento de acuerdo con las condiciones de la licencia Attribution-ShareAlike 3.0 de Creative Commons. Puede ver una copia de ésta en:

<http://creativecommons.org/licenses/by-sa/3.0/es/>



Índice	2
1. Introducción	3
2. HTTPS con certificado autofirmado	3
2.1. Crear un certificado autofirmado con openssl	6
2.2. Utilización de HTTPS en Apache2 con certificado autofirmado	7
2.3. Configuración del navegador	10
3. HTTPS con un certificado emitido por una CA (CAcert)	11
3.1. Creación de un CSR	11
3.2. Configurar Apache con un certificado emitido por una CA	12
3.3. Configuración del navegador	12



1. Introducción

Una vez que tenemos instalado un gestor de contenidos en nuestro servidor web que incluya autenticación de usuarios, uno de los pasos habituales es cifrar el proceso de autenticación de usuarios para evitar que alguien capture (se suele decir *esnife*) una contraseña de usuario y acceda de forma fraudulenta.

El cifrado de la comunicación entre el navegador y el servidor web se hace mediante el protocolo HTTPS, que tiene las siguientes características principales:

- Utiliza el protocolo SSL (actualmente TLS) para el cifrado de datos.
- El servidor utiliza por defecto el puerto 443/tcp.
- Utiliza mecanismos de cifrado de clave pública y las claves públicas se denominan certificados.
- El formato de los certificados está especificado por el estándar X.509 y normalmente son emitidos por una entidad denominada *Autoridad Certificadora (CA por sus siglas en inglés)*. En el caso de HTTPS, la función principal de la CA es demostrar la autenticidad del servidor y que pertenece legítimamente a la persona u organización que lo utiliza. Dependiendo de los criterios utilizados para comprobar la autenticidad del servidor se emiten diferentes tipos de certificados X.509¹.
- El navegador contiene una lista de certificados de CA en las que confía y acepta inicialmente sólo los certificados de los servidores emitidos por alguna de estas CA.
- Una vez aceptado el certificado de un servidor web, el navegador utiliza éste para cifrar los datos que quiere enviar al servidor mediante el protocolo HTTPS y cuando llegan al servidor sólo éste podrá descifrarlos ya que es el único que posee la clave privada que los descifra.

En este documento veremos dos formas de utilizar el protocolo HTTPS en Apache, la más simple utilizando un certificado autofirmado (certificado que no está emitido por ninguna CA) y posteriormente la utilización de un certificado emitido por la organización sin ánimo de lucro [CAcert](#). Los pasos que habría que seguir para utilizar un certificado X.509 emitido por una CA comercial son exactamente los mismos que con CAcert, lo único que variaría sería el método que tendría que utilizar la CA para validar el servidor, la persona u organismo que solicita el certificado.

2. HTTPS con certificado autofirmado

En Debian Lenny existe un paquete denominado *ssl-cert* que se encarga de generar certificados SSL autofirmados para que ciertos servicios que utilizan protocolos sobre SSL como SMTP, HTTP, LDAP, etc. puedan utilizarse de forma sencilla con estos certificados.

Un certificado SSL autofirmado es el que se realiza sin la intervención de una autoridad certificadora y por tanto no existe ningún mecanismo automático que garantice la autenticidad del servidor. En el caso que nos atañe, cifrar la comunicación entre el servidor web y el navegador en el proceso de autenticación en un gestor de contenidos, es una opción totalmente válida la utilización de certificados autofirmados y lo único que hay que hacer si se quiere

¹Las últimas versiones de los navegadores soportan un tipo de certificado X.509 denominado *Extended Validation Certificate*



garantizar la autenticidad del servidor, es establecer un mecanismo alternativo para enviar el certificado X.509 del servidor a los usuarios de forma segura.

Una vez estamos seguros de que el certificado es auténtico podemos utilizarlo con las mismas garantías que un certificado normal, ya que toda la comunicación entre el cliente y el servidor se realiza cifrada mediante la misma técnica y por tanto el canal de comunicación entre el servidor y el cliente se puede considerar *seguro*.

Seguramente tengamos ya el certificado SSL autofirmado instalado en nuestro equipo, porque *ssl-cert* se instala por dependencias al instalar *apache*. Podemos verificarlo mediante la instrucción:

```
avatar:~$ aptitude search ssl-cert |grep ^i
```

```
i A ssl-cert - Envoltura de Debcof sencilla para OpenSSL
```

Al instalar el paquete *ssl-cert* se ejecuta el script *make-ssl-cert*, que utiliza *openssl* para generar un par de certificados (privado y público), cuyo contenido es²:

/etc/ssl/private/ssl-cert-snakeoil.key

```
1 -----BEGIN RSA PRIVATE KEY-----
2 MIICWwIBAAKBgQDWjPGyYKND0zQc9d8ysAm0w+IGt9kypWgGaPbzfuyX41ozk36e
3 U4xVUfUQ8gz3QPj47pNxCIhxkgGD1ioF7Fq0/J7eetl++7MIoub7cVsBG6yPrRbY
4 65zAq7Tg/BJB/Aa3k7t191spbtMs8umKB0+/rRD4bPsgv9iwTD3L8d3QuwIDAQAB
5 AoGAbPdWhfciI1Ji+/fpuKqs3n3ejIzBasptJcv/6dH7BIP9KP6mohDcArhCa+81D
6 hoxMJvncjmvPUKcL9+5cl3eMNTMzomDLASBPmbThHMZMcr6Agpw1BruCjMs1DoUS
7 lL8AHgNng5gRH9th5fThcduucvhp0iqbdCH76GqwUyhxaUkCQQD3b5syTAtb9NM3
8 zGar3PbwQa22DBd6GfLuZrtdvaX+f/XgmaQjiPgj/0oDmDJhyFtl+gp9/tcbzcu
9 LE+iBkF9AkEA3fn1X2Jqm4fFQyZSDCzpwzk4VMTpqfwacdrhcL6OuXAaSSDREsMG
10 FxcFgB462/WHynQhCGqUIzaxa3djCGjwlwJAbTjNuAELTp cemVXXyGtscZd8V4y0
11 3EgV2pKeic3mgiVvwrCkF5E20wycQg8+uBa3McpJSCVXxnCtWetjZ3D9fQJAXRwW
12 IraDhU3EHtCaJeVJAMylplkasmS/KvbnDYXRBN/TBs2vJTGNA m5zrjP/+Hfbr77C
13 hGdAUCOG6s8fSU3LbQJAX5hMWgvuUakt dpwfdMUOglGg7+pZMbl6CqpKtNIHseNp
14 aL2IOX0aoec02hVIUT5xwqRkQ+VUQBs9Nc8a/PI3TQ==
15 -----END RSA PRIVATE KEY-----
```

/etc/ssl/certs/ssl-cert-snakeoil.pem

```
1 -----BEGIN CERTIFICATE-----
2 MIIBsTCCARoCCQDtJd0YjQsT7DANBgkqhkiG9w0BAQUFADAdMRswGQYDVQQDExJh
3 dmFOYXlIuZXhhbXBsZS5jb20wHhcNMTAwMzIxMTEONjA1WhcNMjAwMzE4MTEONjA1
4 WjAdMRswGQYDVQQDExJhdmFOYXlIuZXhhbXBsZS5jb20wgZ8wDQYJKoZIhvcNAQEB
5 BQADgY0AMIGJAoGBANaM8bJgo0PTNBz13zKwCY7D4ga32TKlaAZo9vN+7JfjWjOT
6 fp5TjFVR9RDyDPdA+Pjuk3EKKHGSAYPWKgxSWrT8nt562X77swii5vtxWwEbrI+t
7 FtjrnMCrtOD8EkH8BreTu2X3WyluOyzy6YoE77+tEPhs+yC/2LBMPcvx3dC7AgMB
8 AAEdDQYJKoZIhvcNAQEFBQADgYEAWNCqq9X+ykQ7NEQRUvVN06yHhFs+oCz5nPTJ
9 QfkR30xtM6qI1wzMiRbakkIUTp5Mni/Nw5t1PTNMKFaGhNZKDnV+LRB1i2cgYk0B
10 uCJEFGFflj8+BxeMUhYAX8hxHam86JpkTawUi2+v8QuHGbuLHszoRe3nYZ/nRul/
11 fgk3Wjk=
12 -----END CERTIFICATE-----
```

El certificado X.509 está en formato PEM (base64), para mostrar su contenido podemos utilizar la siguiente instrucción:

```
avatar:~# openssl x509 -in ssl-cert-snakeoil.pem -inform PEM -text
```

```
Certificate:
```

```
Data:
```

²Se muestra aquí la clave privada de un servidor de pruebas que no tiene uso real

```

Version: 1 (0x0)
Serial Number:
    ed:25:dd:18:8d:0b:13:ec
Signature Algorithm: sha1WithRSAEncryption
Issuer: CN=avatar.example.com
Validity
    Not Before: Mar 21 11:46:05 2010 GMT
    Not After : Mar 18 11:46:05 2020 GMT
Subject: CN=avatar.example.com
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
            00:d6:8c:f1:b2:60:a3:43:d3:34:1c:f5:df:32:b0:
            09:8e:c3:e2:06:b7:d9:32:a5:68:06:68:f6:f3:7e:
            ec:97:e3:5a:33:93:7e:9e:53:8c:55:51:f5:10:f2:
            0c:f7:40:f8:f8:ee:93:71:0a:28:71:92:01:83:d6:
            2a:05:ec:5a:b4:fc:9e:de:7a:d9:7e:fb:b3:08:a2:
            e6:fb:71:5b:01:1b:ac:8f:ad:16:d8:eb:9c:c0:ab:
            b4:e0:fc:12:41:fc:06:b7:93:bb:65:f7:5b:29:6e:
            d3:2c:f2:e9:8a:04:ef:bf:ad:10:f8:6c:fb:20:bf:
            d8:b0:4c:3d:cb:f1:dd:d0:bb
        Exponent: 65537 (0x10001)
Signature Algorithm: sha1WithRSAEncryption
    58:d0:aa:ab:d5:fe:ca:44:3b:34:44:11:52:f5:4d:d3:ac:87:
    84:5b:3e:a0:2c:f9:9c:f4:c9:41:f9:11:df:4c:6d:33:aa:88:
    d7:0c:cc:89:16:da:92:42:14:4e:9e:4c:36:2f:cd:c3:9b:65:
    3d:33:4c:28:56:86:84:d6:4a:0e:75:7e:2d:10:75:8b:67:20:
    62:4d:01:b8:22:44:14:61:5f:96:3f:3e:07:17:8c:52:16:00:
    5f:c8:71:1d:a9:bc:e8:9a:64:4d:ac:14:8b:6f:af:f1:0b:87:
    19:b5:0b:1e:cc:e8:45:ed:e7:61:9f:e7:46:e9:7f:7e:09:37:
    5a:39
-----BEGIN CERTIFICATE-----
MIIBsTCCARoCCQDtJd0YjQsT7DANBgkqhkiG9w0BAQUFADAdMRswGQYDVQQDEExJh
dmFOYXlIuZXhhbXBsZS5jb20wHhcNMTAwMzIxMTEONjA1WhcNMjAwMzE4MTEONjA1
WjAdMRswGQYDVQQDEExJhdmFOYXlIuZXhhbXBsZS5jb20wgZ8wDQYJKoZIhvcNAQEB
BQADgYOAMIGJAoGBANaM8bJgoOPTNBz13zKwCY7D4ga32TKlaAZo9vN+7JfjWjOT
fp5TjFVR9RDyDPdA+Pjuk3EKKHGSAYPWKgXsWrT8nt562X77swii5vtxWwEbrI+t
FtjrnMCrtOD8EkH8BreTu2X3Wylu0zyzy6YoE77+tEPhs+yC/2LBMPcvx3dC7AgMB
AAEwDQYJKoZIhvcNAQEFBQADgYEAWNCqq9X+ykQ7NEQRUVVN06yHhFs+oCz5nPTJ
Qfkr30xtM6qI1wzMiRbakkIUTp5MNI/Nw5t1PTNMKFaGhNZKdNv+LRB1i2cgYk0B
uCJEFGFflj8+BxeMUhYAX8hxHam86JpkTawUi2+v8QuHGbuLHszoRe3nYZ/nRul/
fgk3Wjk=
-----END CERTIFICATE-----

```

que es la misma información que nos facilita el navegador web cuando queremos ver el contenido del certificado.

Si el certificado autofirmado no fuese correcto, por ejemplo porque el FQDN del equipo no estaba bien definido cuando se instaló el paquete *ssl-cert*, podemos generar un nuevo certificado con la instrucción:

```
avatar:~# make-ssl-cert generate-default-snakeoil --force-overwrite
```

Si el certificado que se genera así no tiene todas las características necesarias, habrá que crear un certificado autofirmado directamente mediante *openssl*.



2.1. Crear un certificado autofirmado con openssl

Podemos hacerlo en un solo paso, aunque es más instructivo hacerlo en pasos sucesivos. En primer lugar creamos una clave privada RSA de 2048 bits mediante:

```
avatar:~# openssl genrsa 2048 > /etc/ssl/private/ssl-cert.key
```

```
.....+++
.....+++
e is 65537 (0x10001)
```

/etc/ssl/private/ssl-cert.key

```
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEogIBAAKCAQEA6JiUPj0nUh5S7Pr0+vpi3Bpq9Qx9/g7AJdXD/ExIARbefZCt
3 iQZ0tA+sRW2Wa1Ww8aDZq3JCT+KHLV5H2lzBPGjCS0c8hV1H21kXDXV1bD1VQnju
4 6H+l4nqFFA8ETxX/wBU6PONYgbbCn8i183T0cNXC3iPcZyD/frxrBqyVjthTFxFS
5 MJ15g/QA52kwzGoFWFolIMvo5LASH6vM5iSNyS7QWjTp8UKJFejVlM0tdnuKmVOX
6 quhNbjH13MdFgL0HVyNiGD0VyReAfTT2jrKbumdqiZqKFfpXGfezmlIJc4gTLv1J
7 Y3c0no35E2N0SwQ+gnzwwYwIor4sjkSci4igeQIDAQABAoIBAC738KeIzdHlUbwN
8 CBLiU0hFZwfkD/6/l8mMGyFtffTKW29bsc9DuMzfhVgXwxI1oZ+JbasTTZS4F7fv
9 m+7aYCMEvIbgOvZqivoesWtsVqPeB/t+VDAa5rKPLyRXY/TOakqtfq8g8ZRwBwki
10 qXA6I8pNQCZ/c1hX73/2KF6WpzTK04o11wAJc2977Ujkjv5af3s9KbEMrnODIwEy
11 XKREhBF1fd4tuin8N0zcyPwh10RzfFeNsvhPk5Av9z420ZGD0wZTmJJ4+xB9UIA7
12 Q098awFpETwxAmp80cmev4GwEVG8Pp4qkxf0QMGMcj5I4qS0JDqJ0z/dk6TUM/Cb
13 +EPdfqkCgYEA/jqVbYeBh40HgcWjnyR4ho9FFNX0wHfvataaPxLroMC03p+iaRB8
14 4AGc0EPPGduql5PhwWGP3+JhKy4PWKVnKoJKxIyfHwFs+idohDmhX810h/Cb1MN
15 h0vWntNLQKR5J4FqL0ofqGbY90rbUvfSomJ9/JlfaJkINd0zvzuZevcCgYEA6jdp
16 zWlJtVxWhrLu2fxbPCF48EsG1NrAS/5+96n1VMiumhs5KmpXEIyoHXjhUegP4TRf
17 aU00iG3fvRcuVa5oK5943h3YSq1vQDCMUex1Qs2qe8xqwoX6inCv0aUa6+Che/A8
18 BfJ8pG6e/1TyvKtFjs2ZtgK3YjZV9qK8VUmh9A8CgYB01hXugD3frhEs4gAKYsHp
19 YFZYzrx2Tvr0k5YDhzeAgxYK3U/86rr+sCEkpZKMjT0KgjZYxIs9DeC0PudnScIB
20 cg0jySHUzpTqJHF1h24k6bzj6ytdYFDqqs7kM7u7UwmugYh0Y0/r5fDURudahDVS
21 W96bzWfUBaGRJ5Qu8il3IwKBgFFZu+L/3f0jPF5+yd1m85WaHrJ/rLjD3iTHGXyW
22 SMM0yzb1m65yzzjNchsCgGh9jP5wvW8Kx0NaHmtEIPDbqgz0z/t+LEEL6mdGNYbC
23 9ZulacCkKLS16mapazgK+8XP+0bec0qxhb9r0MiAzergJkHKgCGp017edo86y0Vq
24 tNjbAoGAF+mMJIwPyjHoZPio62vGtJp3RgUayR/mtrmOpUMNqYD05R1mKkiz3YE4
25 Nm15WFbY04i6I8HN0gTzYBuScfjArMLU1+S/AwaLH1pKKdw7Sc29iaw50dGU4h+2
26 xZbPC/IqL1HzEHPp71TbtKYSMvUzMMb1LUJdyIbqvJafddmUuSE=
27 -----END RSA PRIVATE KEY-----
```

Modificamos de forma apropiada los propietarios y permisos:

```
avatar:~# chown root:ssl-cert /etc/ssl/private/ssl-cert.key
avatar:~# chmod 640 /etc/ssl/private/ssl-cert.key
```

Con la clave privada anterior creamos un certificado X.509 de un año de validez. En este proceso openssl nos solicitará cierta información para completar los atributos del certificado:

```
avatar:~# openssl req -new -x509 -nodes -sha1 -days 365 \
-key ssl-cert.key > avatar.pem
```

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]:Sevilla
Locality Name (eg, city) []:Lora del Río
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CEP de Lora \
del Río
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:
```

/etc/ssl/certs/avatar.pem

```
1 -----BEGIN CERTIFICATE-----
2 MIID6TCCA+GgAwIBAgIJAL+w+1GksQnTMAOGCSqGSIb3DQEBBQUAMFYxCzAJBgNV
3 BAYTAkVTMRAwDgYDVQQIEwdTZXZpbGxhMRYwFAyDVQQHFA1Mb3JhIGRlbcBSw61v
4 MR0wGwYDVQQKFBRRDRVAgZGUgTG9yYSBkZWwgUs0tbzAeFw0xMDAzMjIxNzA0MDJa
5 Fw0xMTAzMjIxNzA0MDJAMFYxCzAJBgNVBAYTAkVTMRAwDgYDVQQIEwdTZXZpbGxh
6 MRYwFAyDVQQHFA1Mb3JhIGRlbcBSw61vMR0wGwYDVQQKFBRRDRVAgZGUgTG9yYSBk
7 ZWwgUs0tbzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA0iY1D49J1Ie
8 Uuz6zvr6YtwaavUMff40wCXVw/xMSAEW3n2QrYkGTrQPrEVtlmtVsPGg2atyQk/i
9 hy1eR9pcwTxowkijnPIVZR9tZFW11dWw5VUJ47uh/peJ6hRQPBE8V/8AV0jzjWIG2
10 wp/IpfnO9HDVwt4j3Gcg/368awas1Y7YUxcRUjCZeYPOA0dpMMxqBVhaJSDL60Sw
11 Eoerz0Ykjcku0Fo06fFCiRXo1ZTDrXZ7iplTl6roTW4x9dzHRYC9B1cjYhgZlckX
12 gH009o6ym7pnaomaihX6Vxn3s5pSCX0IEy75SWN3Dp6N+RNjdEsEPoJ88M2MCKK+
13 LI5EnIuIoHkCAwEAa0BuTCBtjAdBgNVHQ4EFgQUUpiqB+NcKq+jW8SmQlxqHo/E
14 R3swgYYGA1UdIwR/MH2AFFKYqgfjXCqvo1vEpkJcah6PxEd7oVqkWBWMQswCQYD
15 VQQGEwJFUzEQMA4GA1UECBMHU2V2aWxsYTEwMBQGA1UEBxQNTG9yYSBkZWwgUs0t
16 bzEdMBsGA1UEChQUQ0VQIGRlIEExvcmEgZGVsIFFLDrW+CCQC/sPtRPLEJ0zAMBGNV
17 HRMEBTADAQH/MAOGCSqGSIb3DQEBBQUAA4IBAQCWzmNKctX4Ph1jh7Sg8FIo6Syq
18 8vcbxK9h3WT2fRTwM3JaisW+KtBLP3RidqacIvPFt4o2qZnxnhRlhqcmWuUw8zU4
19 rtdsTebtV0cwDxhGjmvysLS6SkSLi31X+uxC2Vle+n6zy9JvazwSG+qD36TDcYV/r
20 mqKZu+cz9T95uFIIUesFwQ5P4RNVtugi5vUf1avo4AL+u0+N4/zetElFSKtDdEL5
21 5iMc77VhvnYk3Wr1D+VSIhwpbdsKvYvKdXlC+xykTaqrAPW1ApiFrYwGiarCd3zy
22 nI8C3yKYXAYzlsbaM9EW72vgxjprBbX8WMM4n9vLEKMDNKfywN9nPz7hgUSH
23 -----END CERTIFICATE-----
```

2.2. Utilización de HTTPS en Apache2 con certificado autofirmado

Cuando instalamos apache2 por defecto sólo se activa el protocolo HTTP y se abre el puerto 80/tcp, para utilizar el protocolo HTTPS debemos activar el módulo ssl:

```
avatar:~# a2enmod ssl
```

Enabling module ssl.

See /usr/share/doc/apache2.2-common/README.Debian.gz on how to configure SSL and create self-signed certificates.

Run '/etc/init.d/apache2 restart' to activate new configuration!

Como bien nos indica la salida de la instrucción anterior, ahora es necesario reiniciar el servidor apache:

```
avatar:~# /etc/init.d/apache2 restart
```

Si vemos el contenido del fichero *ports.conf*:

/etc/apache2/ports.conf

```
1 # If you just change the port or add more ports here, you will likely
2 # also have to change the VirtualHost statement in
```

```

3 # /etc/apache2/sites-enabled/000-default
4 # This is also true if you have upgraded from before 2.2.9-3 (i.e.
5 # from Debian etch). See
6 # /usr/share/doc/apache2.2-common/NEWS.Debian.gz and README.Debian.gz
7
8 NameVirtualHost *:80
9 Listen 80
10
11 <IfModule mod_ssl.c>
12     # SSL name based virtual hosts are not yet supported, therefore
13     # no NameVirtualHost statement here
14     Listen 443
15 </IfModule>

```

podemos comprobar que apache sólo escucha peticiones en el puerto 443/tcp cuando el módulo `ssl` está activado (como es el caso ahora y podemos comprobar):

```
avatar:~# netstat -putan |grep apache
```

```

tcp6      0    0  :::80          :::*           LISTEN      3177/apache2
tcp6      0    0  :::443         :::*           LISTEN      3177/apache2

```

Sin embargo, apache todavía no acepta peticiones del tipo <https://avatar.example.com>, porque el único sitio que está activo está definido sólo en el puerto 80/tcp, como podemos comprobar viendo su contenido:

/etc/apache2/sites-available/default

```

1 <VirtualHost *:80>
2     ServerAdmin webmaster@localhost
3
4     DocumentRoot /var/www/
5     <Directory />
6         Options FollowSymLinks
7         AllowOverride None
8     </Directory>
9     <Directory /var/www/>
10        Options Indexes FollowSymLinks MultiViews
11        AllowOverride None
12        Order allow,deny
13        allow from all
14    </Directory>
15
16    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
17    <Directory "/usr/lib/cgi-bin">
18        AllowOverride None
19        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
20        Order allow,deny
21        Allow from all
22    </Directory>
23
24    ErrorLog /var/log/apache2/error.log
25
26    # Possible values include: debug, info, notice, warn, error,
27    # crit, alert, emerg.
28    LogLevel warn
29    CustomLog /var/log/apache2/access.log combined
30

```




```

31     Alias /doc/ "/usr/share/doc/"
32     <Directory "/usr/share/doc/">
33         Options Indexes MultiViews FollowSymLinks
34         AllowOverride None
35         Order deny,allow
36         Deny from all
37         Allow from 127.0.0.0/255.0.0.0 ::1/128
38     </Directory>
39
40 </VirtualHost>

```

Debian Lenny incluye otro sitio predefinido que escucha peticiones en el puerto 443 y que está preparado para utilizar SSL, por tanto lo único que hay que hacer es activar este sitio:

```
avatar:~# a2ensite default-ssl
```

```
Enabling site default-ssl.
```

```
Run '/etc/init.d/apache2 reload' to activate new configuration!
```

```
avatar:~# /etc/init.d/apache2 reload
```

Veamos el contenido del fichero default-ssl:

/etc/apache2/sites-available/default-ssl

```

1 <IfModule mod_ssl.c>
2 <VirtualHost _default_:443>
3     ServerAdmin webmaster@localhost
4
5     DocumentRoot /var/www/
6     <Directory />
7         Options FollowSymLinks
8         AllowOverride None
9     </Directory>
10    <Directory /var/www/>
11        Options Indexes FollowSymLinks MultiViews
12        AllowOverride None
13        Order allow,deny
14        allow from all
15    </Directory>
16
17    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
18    <Directory "/usr/lib/cgi-bin">
19        AllowOverride None
20        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
21        Order allow,deny
22        Allow from all
23    </Directory>
24
25    ErrorLog /var/log/apache2/error.log
26
27    # Possible values include: debug, info, notice, warn, error, crit,
28    # alert, emerg.
29    LogLevel warn
30
31    CustomLog /var/log/apache2/ssl_access.log combined
32
33    Alias /doc/ "/usr/share/doc/"
34    <Directory "/usr/share/doc/">

```



```

35     Options Indexes MultiViews FollowSymLinks
36     AllowOverride None
37     Order deny,allow
38     Deny from all
39     Allow from 127.0.0.0/255.0.0.0 ::1/128
40 </Directory>
41
42 #   SSL Engine Switch:
43 #   Enable/Disable SSL for this virtual host.
44 SSLEngine on
45
46 #   A self-signed (snakeoil) certificate can be created by
47 #   installing the ssl-cert package. See
48 #   /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
49 #   If both key and certificate are stored in the same file, only
50 #   the SSLCertificateFile directive is needed.
51 SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
52 SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
53 ...

```

Sirve los mismos ficheros que mediante HTTP, pero lo hace mediante HTTPS con el certificado y clave que se especifican en las líneas 51 y 52.

2.3. Configuración del navegador

Cuando establecemos por primera vez una conexión HTTPS con servidor web que utiliza un certificado autofirmado, el navegador nos advertirá que no reconoce ese servidor con un mensaje como el que aparece en la figura 1³.



Figura 1: Firefox/Iceweasel advirtiéndole que no se confía en el certificado autofirmado del servidor

³En este caso además es incorrecto el nombre del servidor

La forma de solucionar esto es añadir una excepción, que consiste simplemente en aceptar de forma permanente o no el certificado X.509 que nos está facilitando el servidor web, lo que nos garantiza que la comunicación entre el cliente y el servidor web se realizará de forma cifrada. De esta forma lo que nunca se garantiza es la autenticidad del servidor (que el servidor es quien dice ser), ya que podría darse el caso de que aceptásemos el certificado de un servidor que hubiese suplantado al legítimo y posteriormente le diésemos información relevante⁴.

3. HTTPS con un certificado emitido por una CA (CAcert)

El lema de CAcert es *Free digital certificates for everyone* y es que la utilización de certificados emitidos por CA comerciales no es posible para todos los sitios de Internet debido a su coste, lo que los limita su uso a transacciones económicas o sitios con datos relevantes. CAcert es una organización sin ánimo de lucro que mantiene una infraestructura equivalente a una CA comercial aunque con ciertas limitaciones.

Los pasos que hay que dar para utilizar un certificado X.509 emitido por CAcert son los siguientes:

- Darse de alta como usuario en el sitio <http://cacert.org>
- Dar de alta el dominio para el que queremos obtener el certificado, por ejemplo en nuestro caso `avatar.dynalias.com`. CAcert verifica que podemos hacer uso legítimo del dominio enviando un mensaje de correo electrónico al usuario *hostmaster* o *root* de ese dominio.
- Dar de alta el certificado de un servidor mediante una solicitud de firma certificado (CSR)
- Configurar el servidor web con el certificado X.509 emitido por la CA.

Los dos primeros pasos son triviales, los dos siguientes los explicamos con detalle a continuación.

3.1. Creación de un CSR

CSR son las siglas de *Certificate Signing Request* o solicitud de firma de certificado. La principal diferencia entre un certificado autofirmado como el visto anteriormente y un certificado emitido por una CA es precisamente este paso intermedio. Utilizando una clave privada generada por nosotros y a la que sólo nosotros tenemos acceso, generamos una CSR mediante la instrucción:

```
avatar:~# openssl req -new -key /etc/ssl/private/ssl-cert.key -out \
/etc/ssl/private/avatar.csr
```

/etc/ssl/private/avatar.csr

```
1 -----BEGIN CERTIFICATE REQUEST-----
2 MIICmzCCAQMCAQAwVjELMAkGA1UEBhMCRVMxEDAQBgNVBAgTB1NldmlsbGEeFjAU
3 BgNVBAcUDUxvcmEgZGVsIFlDrW8xHTAbBgNVBAoUFENFUCBkZSBMb3JhIGRlbCBS
4 w61vMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA6JiUPj0nUh5S7PrO
5 +vpi3Bpq9Qx9/g7AJdXD/ExIARbefZCtiQZ0tA+sRW2Wa1Ww8aDZq3JCT+KHLV5H
```

⁴Esto es la base del conocido *phishing*



```

6 21zBPGjCS0c8hV1H21kXDXV1bD1VQnju6H+14nqFFA8ETxX/wBU6PONYgbbCn8i1
7 83T0cNXC3iPcZyD/frxrBqyVjthTFxFSMJ15g/QA52kwzGoFWFolIMvo5LASH6vM
8 5iSNyS7QWjTp8UKJFejVlM0tdnuKmVOXquhNbjH13MdFgLOHVyNiGD0VyReAfTT2
9 jrKbumdqiZqKFfpXGfezmlIJc4gTLvlJY3c0no35E2N0SwQ+gnzwzYwIor4sjkSc
10 i4igeQIDAQABoAAwDQYJKoZIhvcNAQEFBQADggEBAGr16bjcCjvdc9UmDvE6d++2
11 4V8YHawuZPOGf1sStS8JoPUJrisCI2dG7/ydBWl++egHKT04j/xs0Jstj8fM3R4c
12 wv6FcUbz0gTJSCiGUQpx4AETyDRQqa+4z+AtV7E30hUNQ4cKDEGZKL+rd28I8J1c
13 7c0yccjorSC+zA16tfgLxMiaNZ1kaZFPCYyiA2sLufECT/4DdqMHphexE5RvlGB1
14 Qvb2+rR62R1v2Cat6+LXXY2zZAJKfqT4qTU6gXxqJJUx1a8D08JI dy2zUnJPG5Ce
15 GwbFSNypVQXhjFqRp9jjh0A/FvCbv2hN5L00tFS34ceCzHx3BUQT26Fbsm7bdmA=
16 -----END CERTIFICATE REQUEST-----

```

y este fichero csr es el que enviamos a la CA a través del formulario web, que lo procesa y obtiene a partir de él un certificado X.509 compatible con nuestra clave privada, pero emitido por la CA. Ese certificado X.509 será el que tengamos que utilizar en nuestro servidor web en lugar del certificado autofirmado anterior.

3.2. Configurar Apache con un certificado emitido por una CA

Una vez que la CA emite el certificado X.509, guardamos éste en un fichero de nuestro sistema, por ejemplo en `/etc/ssl/certs/avatar-cacert.pem` y modificamos las siguientes líneas de nuestro sitio HTTPS:

```

                                /etc/apache2/sites-available/default-ssl
51 SSLCertificateFile           /etc/ssl/certs/avatar-cacert.pem
52 SSLCertificateKeyFile        /etc/ssl/private/ssl-cert.key
53 SSLCertificateChainFile      /etc/ssl/certs/cacert.org.pem

```

donde la última línea indica la ubicación del certificado raíz de CAcert.org, que podemos obtener directamente del sitio o instalando el paquete `ca-certificates`.

3.3. Configuración del navegador

Algunas distribuciones GNU/Linux incluyen los diferentes certificados raíz de CAcert entre los de las autoridades certificadoras en las que confían, como es el caso de Debian que la incluye junto a otras en el paquete `ca-certificates`:

```
avatar:~$ dpkg -L ca-certificates | grep cacert.org
```

```

/usr/share/ca-certificates/cacert.org
/usr/share/ca-certificates/cacert.org/cacert.org.crt
/usr/share/ca-certificates/cacert.org/root.crt
/usr/share/ca-certificates/cacert.org/class3.crt

```

En el caso de que utilicemos un navegador que no incluya el certificado de CAcert, en lugar de hacer una excepción cuando nos conectemos a nuestro sitio, lo lógico es incluir en la lista de certificados en los que confía el navegador el [certificado raíz de cacert](#).

