Apache2 + SSL

Alejandro Valdés Jimenez
avaldes@utalca.cl

Índice

1. Objetivos

El alumno al finalizar la experiencia deberá ser capaz de:

- Generar certificados (CSR,CRT)
- Configurar un servidor web seguro
- Verificar como es la comunicación entre el servidor web y un cliente.

2. Introducción

En esta experiencia veremos como generar los certificados digitales para permitir la comunicación de segura entre un servidor web (Apache) y un cliente cualquiera (Firefox, IExplorer, etc). Realizando algunas pruebas con ethereal vamos comprobar que la información que viaja entre el cliente y el servidor, una vez establecida la conexión viaja en modo seguro, lo que garantiza la confidencialidad de los datos.

El laboratorio se realizará utilizando una máquina con Linux como sistema operativo y GNU/Debian7 como distribución. Los paquetes necesarios y que deben estar instalados son:

- apache2
- apache2-common
- ssl-cert
- openssl:

3. SSL - Secure Sockets Layer

Secure Sockets Layer (SSL) proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía. Habitualmente, solo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar; la autenticación mutua requiere un despliegue de infraestructura de claves públicas (o PKI) para los clientes. Los protocolos permiten a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir escuchas, la falsificación de la identidad del remitente y mantener la integridad del mensaje.

4. Gestión de certificados

Trabajar con SSL nos permite que todos los datos que se transfieren entre el cliente y el servidor vayan cifrados. Antes de comenzar el trabajo, debemos entender algunos términos:

- **RSA Private Keys**: Archivo digital que podemos usar para descifrar mensajes que nos envian. Tiene una parte pública (que distribuimos con nuestro certificado), que permite a la gente cifrar los mensajes que nos envia. Este mecanismo de clave asimétrica nos asegura que los mensajes cifrados con la clave pública (que distribuimos a mucha gente) sólo pueden ser descifrados con la clave privada (que sólo conocemos nosotros).
- **Certificate Signing Request (CSR)**: Es un archivo digital que contiene nuestra clave pública y nuestro nombre. Nos sirve para ser enviado al CA para que nos firme y acredite.
- **Certification Authority (CA)**: Entidad de confianza encargada de firmar certificados (CSR).
- **Certificate (CRT)**: Una vez la CA a firmado el CSR, obtenemos un CRT. Este archivo contiene nuestra clave pública, nuestro nombre, el nombre del CA, y está firmado digitalmente por la CA. De esta forma otras entidades pueden verificar esta firma para comprobar la veracidad del certificado. Es decir, si obtenemos un certificado que esta firmado por una CA que nosotros consideramos de confianza, podemos confiar también en la autenticidad del certificado.

El procedimiento de creación de certificados se puede resumir en la figura 1.

Figura 1: Generación de certificados.

4.1. Creando nuestro CA

Existen numerosas **CAs**, que previo pago pueden firmar nuestro **CSR**. Estas son muldialmente conocidasde manera que cualquier cliente podrá conectarse con confianza a nuestro servidor. VeriSign es un ejemplo de **CA**.

Para nuestro propósito, crearemos nuestro propia **CA**, sin embargo para que nuestro certificado tenga validez en la red debieramos enviar nuestro **CSR** a un **CA** válido.

Para crear los certificados necesitamos tener instalado el paquete **openssl**, el que nos provee de un script para hacer nuestro trabajo: */usr/lib/ssl/misc/CA.sh*

El siguiente ejemplo muestra los pasos como crear nuestro **CA**. Debe ejecutar el script como superusuario. La ruta puede variar pero por orden es mejor hacerlo como en el ejemplo.

```
root@debian:/etc/apache2# /usr/lib/ssl/misc/CA.sh -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 2048 bit RSA private key
.....................................................+++
.....................+++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase: <Ingresar una clave>
Verifying - Enter PEM pass phrase: <Repetir la 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]:CL
State or Province Name (full name) [Some-State]:Talca
Locality Name (eg, city) []:Talca
Organization Name (eg, company) [...]:Universidad de Talca
Organizational Unit Name (...) []:Escuela de Ingeniería en Bioinformática
Common Name (e.g. server FQDN or YOUR name) []:Autoridad Certificadora EIBI
Email Address []:avaldes@utalca.cl

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: <Enter>
An optional company name []: <Enter>
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/./cakey.pem:
Check that the request matches the signature
```

```
Signature ok
Certificate Details:
        Serial Number:
            a1:1e:7e:ba:8d:88:e3:bf
        Validity
            Not Before: Apr  5 13:14:12 2013 GMT
            Not After : Apr  4 13:14:12 2016 GMT
        Subject:
        countryName               = CL
        stateOrProvinceName       = Talca
        organizationName          = Universidad de Talca
        organizationalUnitName    = Escuela de Ingeniería en Bioinformática
        commonName                = Autoridad Certificadora EIBI
        emailAddress              = avaldes@utalca.cl
        X509v3 extensions:
        X509v3 Subject Key Identifier:
          E5:A6:FA:38:63:74:B1:9A:38:C2:E9:97:D2:32:B1:11:DA:73:FA:6C
        X509v3 Authority Key Identifier:
          keyid:E5:A6:FA:38:63:74:B1:9A:38:C2:E9:97:D2:32:B1:11:DA:73:FA:6C

            X509v3 Basic Constraints:
                CA:TRUE
Certificate is to be certified until Apr  4 13:14:12 2016 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
```

Al ejecutar el script, nos hará las siguientes preguntas:

- Nombre del certificado de la **CA** o que pulsemos <Enter> para crearlo. En nuestro caso pulsamos <Enter> para crear uno nuevo.
- Una <pass phrase>, que nos vuelve a preguntar para confirmarla. Esta será la clave para acceder a la clave privada (la clave privada se guarda cifrada).
- Cierta información para añadir al certificado, código del país, provincia, localidad, etc.

Una vez terminada la ejecución del script podemos ver en el directorio actual que aparece una nueva carpeta, *demoCA*, la que tiene la siguiente estructura:

```
root@debian:/etc/apache2/demoCA# ls -l
total 40
-rw-r--r-- 1 root root 4848 abr  5 10:14 cacert.pem
-rw-r--r-- 1 root root 1143 abr  5 10:14 careq.pem
drwxr-xr-x 2 root root 4096 abr  5 10:10 certs
drwxr-xr-x 2 root root 4096 abr  5 10:10 crl
-rw-r--r-- 1 root root  216 abr  5 10:14 index.txt
-rw-r--r-- 1 root root   21 abr  5 10:14 index.txt.attr
-rw-r--r-- 1 root root    0 abr  5 10:10 index.txt.old
drwxr-xr-x 2 root root 4096 abr  5 10:14 newcerts
drwxr-xr-x 2 root root 4096 abr  5 10:10 private
-rw-r--r-- 1 root root   17 abr  5 10:14 serial
```

4.2. Creando nuestro CSR

Ahora crearemos nuestro **CSR**, el cual debería firmar un autentico **CA**, pero en este caso lo firmaremos nosotros con nuestra propia **CA** que generamos anteriormente.

Primero, creamos la clave para nuestro servidor Apache, la clave será **triple-DES** y en formato **PEM**. Crearemos una carpeta para alojar los archivos, la llamaremos *cert*. Al ejecutar el script se nos pedirá ingresar una *pass phrase* y luego confirmarla.

```
root@debian:~/cert# openssl genrsa -des3 -out midominio.cl.key 1024
Generating RSA private key, 1024 bit long modulus
..................++++++
.....++++++
e is 65537 (0x10001)
Enter pass phrase for midominio.cl.key:
Verifying - Enter pass phrase for midominio.cl.key:

root@debian:~/cert# ls -l
total 4
-rw-r--r-- 1 root root 963 abr  5 11:42 midominio.cl.key
```

Una vez creada la clave del servidor, generamos el **CSR** usando la clave generada anteriormente.

```
root@debian:~/cert# openssl req -new -days 365 -key midominio.cl.key
  -out midominio.cl.csr
Enter pass phrase for midominio.cl.key:
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]:CL
State or Province Name (full name) [Some-State]:Talca
Locality Name (eg, city) []:Talca
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Particular
Organizational Unit Name (eg, section) []:Particular
Common Name (e.g. server FQDN or YOUR name) []:midominio.cl
Email Address []:avaldes@utalca.cl

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@debian:~/cert# ls -l
total 8
-rw-r--r-- 1 root root 708 abr  5 11:49 midominio.cl.csr
-rw-r--r-- 1 root root 963 abr  5 11:45 midominio.cl.key
```

Lo primero que nos pedirá será la *pass phrase* que pusimos a la clave en el punto anterior. Luego nos pedirá los datos que queremos agregar a nuestro certificado. Aquí, cuando nos consulte por el *Common Name* debemos poner el nombre completo del dominio del servidor, en nuestro caso colocaremos *midominio.cl*

4.3. Creando nuestro CRT

El último paso es firmar el **CSR** para conseguir el **CRT**. Para esto volvemos a usar el script *CA.sh*. Necesitamos modificar un poco el archivo de configuracion */usr/lib/ssl/openssl.cnf* y cambiar la línea:

```
#dir = ./demoCA           # Where everything is kept
dir = /etc/apache/demoCA  # Where everything is kept
```

Además, creamos un enlace simbólico a nuestro **CSR**, ya que el script tiene algunos datos configurados en duro.

```
root@debian:~/cert# ln -s midominio.cl.csr newreq.pem
root@debian:~/cert# ls -l
total 8
-rw-r--r-- 1 root root 708 abr  5 11:49 midominio.cl.csr
-rw-r--r-- 1 root root 963 abr  5 11:45 midominio.cl.key
lrwxrwxrwx 1 root root  16 abr  8 21:12 newreq.pem -> midominio.cl.csr
```

Ahora ejecutamos el escript para firmar el certificado:

```
root@debian:~/cert# /usr/lib/ssl/misc/CA.sh -signreq
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for /etc/apache2/demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            a1:1e:7e:ba:8d:88:e3:c0
        Validity
            Not Before: Apr  9 00:16:07 2013 GMT
            Not After : Apr  9 00:16:07 2014 GMT
        Subject:
            countryName               = CL
            stateOrProvinceName       = Talca
            localityName              = Talca
            organizationName          = Particular
            organizationalUnitName    = Particular
            commonName                = midominio.cl
            emailAddress              = avaldes@utalca.cl
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
```

```
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                CE:FD:2B:A2:87:DD:B5:DC:FE:3B:13:27:AE:46:66:C6:AE:99:0B
            X509v3 Authority Key Identifier:
                keyid:E5:A6:FA:38:63:74:B1:9A:38:C2:E9:97:D2:32:B1:11:DA

Certificate is to be certified until Apr  9 00:16:07 2014 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            a1:1e:7e:ba:8d:88:e3:c0
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=CL, ST=Talca, O=Universidad de Talca,
          OU=Escuela de Ingeniería en Bioinformática,
          CN=Autoridad Certificadora EIBI/emailAddress=avaldes@utalca.cl
        Validity
            Not Before: Apr  9 00:16:07 2013 GMT
            Not After : Apr  9 00:16:07 2014 GMT
        Subject: C=CL, ST=Talca, L=Talca, O=Particular, OU=Particular,
          CN=midominio.cl/emailAddress=avaldes@utalca.cl
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:c1:38:92:21:c4:3a:fe:f8:df:c0:90:e5:77:e0:
                    dd:11:7f:a3:8c:10:03:02:0d:49:7b:4f:04:17:f2:
                    a4:6f:3b:58:03:a6:3c:ad:93:3b:4e:cc:ae:0c:2f:
                    88:d6:7a:71:f4:e1:c6:a5:f8:25:77:7c:40:1f:f0:
                    5d:17:97:9f:40:a1:68:d9:d6:b1:40:dd:59:81:7a:
                    dc:a3:4d:32:b2:cd:e3:73:50:82:ea:70:e2:54:d6:
                    51:98:2a:d7:da:50:0e:a6:d3:6f:e4:c4:f7:01:41:
                    13:5c:1d:ec:77:5a:d7:72:14:1d:41:56:08:e7:dd:
                    9f:3f:f4:05:c2:ec:0f:11:b3
                Exponent: 65537 (0x10001)
        X509v3 extensions:
```

```
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
              CE:FD:2B:A2:87:DD:B5:DC:FE:3B:13:27:AE:46:66:C6:AE:99:0B
            X509v3 Authority Key Identifier:
              keyid:E5:A6:FA:38:63:74:B1:9A:38:C2:E9:97:D2:32:B1:11:DA

    Signature Algorithm: sha1WithRSAEncryption
         96:8a:87:09:aa:b9:b8:64:38:ba:99:01:89:ef:0f:20:98:4d:
         82:6b:cc:e3:ca:2e:b4:bf:17:3a:6c:84:89:9f:33:a6:51:2f:
         9c:01:71:bc:5e:fa:a8:8e:61:56:09:96:ee:e0:fa:16:03:13:
         02:5a:00:40:ec:f7:c9:57:7c:57:30:91:f3:34:54:de:84:6d:
         c0:07:95:74:0e:35:cf:8a:53:af:ad:ca:4a:1a:fb:7e:52:13:
         40:9b:f9:d9:b9:66:98:af:84:82:a9:cf:0e:f7:c6:9b:98:eb:
         f8:47:37:a0:65:ef:aa:c8:f4:9f:c5:70:e4:76:4f:b4:f0:48:
         a4:56:99:c6:b1:ea:34:b7:45:2d:d0:ec:a7:13:8a:b2:64:f8:
         c2:4d:72:75:93:a9:ee:19:5f:92:7f:ef:34:5f:d4:03:30:5c:
         14:ad:06:84:8d:c8:46:1b:36:54:c8:eb:51:d0:d2:60:a1:45:
         2f:bc:bb:f5:17:97:5c:38:0d:02:a3:9e:bf:c0:71:96:6f:3c:
         51:59:72:b9:c7:6c:1a:fd:e4:1f:51:d8:40:2a:be:17:b5:20:
         5a:61:55:06:2c:66:6a:14:6f:f6:2e:28:ad:dd:8e:cc:63:85:
         93:32:0e:bb:d6:54:c4:fd:ab:4c:82:9f:1c:78:4c:60:42:27:
         8e:51:42:b3
-----BEGIN CERTIFICATE-----
MIIDyTCCArGgAwIBAgIJAKEefrqNiOPAMA0GCSqGSIb3DQEBBQUAMIG9MQswCQYD
VQQGEwJDTDEOMAwGA1UECAwFVGFsY2ExHTAbBgNVBAoMFFVuaXZlcnNpZGFkIGRl
IFRhbGNhMTYwNAYDVQQLDC1Fc2N1ZWxhIGRlIEluZ2VuaWVyw4PCrWEgZW4gQmlv
aW5mb3Jtw4PCoXRpY2ExJTAjBgNVBAMMHEF1dG9yaWRhZCBDZXJ0aWZpY2Fkb3Jh
IEVJQkkxIDAeBgkqhkiG9w0BCQEWEWF2YWxkZXNAdXRhbGNhLmNsMB4XDTEzMDQw
OTAwMTYwN1oXDTE0MDQwOTAwMTYwN1owgZAxCzAJBgNVBAYTAkNMMQ4wDAYDVQQI
DAVUYWxjYTEOMAwGA1UEBwwFVGFsY2ExEzARBgNVBAoMClBhcnRpY3VsYXIxEzAR
BgNVBAsMClBhcnRpY3VsYXIxFTATBgNVBAMMDG1pZG9taW5pby5jbDEgMB4GCSqG
SIb3DQEJARYRYXZhbGRlc0B1dGFsY2EuY2wwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAME4kiHEOv7438CQ5Xfg3RF/o4wQAwINSXtPBBfypG87WAOmPK2TO07M
rgwviNZ6cfThxqX4JXd8QB/wXReXn0ChaNnWsUDdWYF63KNNMrLN43NQgupw4lTW
UZgq19pQDqbTb+TE9wFBE1wd7Hda13IUHUFWCOfdnz/0BcLsDxGzAgMBAAGjezB5
MAkGA1UdEwQCMAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZXJhdGVkIENl
cnRpZmljYXRlMB0GA1UdDgQWBBTO/Suih9213P47EyeuRmbGrpkLrzAfBgNVHSME
GDAWgBTlpvo4Y3SxmjjC6ZfSMrER2nP6bDANBgkqhkiG9w0BAQUFAAOCAQEAloqH
Caq5uGQ4upkBie8PIJhNgmvM48outL8XOmyEiZ8zplEvnAFxvF76qI5hVgmW7uD6
```

```
FgMTAloAQOz3yVd8VzCR8zRU3oRtwAeVdA41z4pTr63KShr7flITQJv52blmmK+E
gqnPDvfGm5jr+Ec3oGXvqsj0n8Vw5HZPtPBIpFaZxrHqNLdFLdDspxOKsmT4wk1y
dZOp7hlfkn/vNF/UAzBcFK0GhI3IRhs2VMjrUdDSYKFFL7y79ReXXDgNAqOev8Bx
lm88UVlyucdsGv3kH1HYQCq+F7UgWmFVBixmahRv9i4ord2OzGOFkzIOu9ZUxP2r
TIKfHHhMYEInjlFCsw==
-----END CERTIFICATE-----
Signed certificate is in newcert.pem

root@debian:~/cert# ls -l
total 16
-rw-r--r-- 1 root root  708 abr  5 11:49 midominio.cl.csr
-rw-r--r-- 1 root root  963 abr  5 11:45 midominio.cl.key
-rw-r--r-- 1 root root 4098 abr  8 21:16 newcert.pem
lrwxrwxrwx 1 root root   16 abr  8 21:12 newreq.pem -> midominio.cl.csr
```

Nos pedirá la *pass phrase* que usamos para crear el certificado del **CA**. Nos mostrará información de nuestro certificado y nos preguntará si queremos firmar el certificado. Luego nos preguntará si queremos hacer *commit* de los certificados firmados, es decir, si queremos confirmar la operación. Al finalizar, habrá generado el archivo *newcert.pem*, el que renombraremos como *midominio.cl.crt*

Por último, debemos mover nuestros archivos a otros directorios, en los que apache contiene los certificados (por un tema de orden).

```
root@debian:~/cert# mv newcert.pem midominio.cl.crt
root@debian:~/cert# mkdir /etc/apache2/ssl.key
root@debian:~/cert# mkdir /etc/apache2/ssl.crt
root@debian:~/cert# mv midominio.cl.key /etc/apache2/ssl.key/
root@debian:~/cert# mv midominio.cl.crt /etc/apache2/ssl.crt/
```

5. Apache

El servidor HTTP Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 (RFC 2616) y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo.

Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor parcheado).

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido.

5.1. Preparando el entorno de pruebas

Habilitamos el módulo con el soporte de Apache2 para SSL y comprobamos que servicios corren localmente. El que nos interesa que aparezca el el puerto 443/TCP, el puerto por defecto para este servicio.

```
root@debian:/etc/apache2/mods-available# a2enmod ssl

root@debian:~# nmap localhost

Starting Nmap 6.00 ( http://nmap.org ) at 2013-04-08 21:47 CLST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000025s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

Como vamos a trabajar con un dominio que no existe, es solo de pruebas, modificaremos algunos archivos para que el browser envie la solicitud de página localmente.

En */etc/host.conf* agregamos en la primera linea la orden para que se busque primero localmente información del dominio *midominio.cl*

```
order hosts, bind
multi on
```

Y en */etc/hosts* agregamos la referencia:

```
127.0.1.1       debian midominio.cl
```

Modificamos un poco la página inicial del sitio por defecto alojada en */var/www/index.html* tanto para el servicio normal como para el encriptado.

```
<html>
<body>
  <h1>midominio.cl!</h1>
</body>
</html>
```

Modificamos los archivos de configuración por defecto */etc/apache2/sites-available/default*

```
...
ServerName  midominio.cl
DocumentRoot /var/www
...
```

Y */etc/apache2/sites-available/default-ssl*, indicando la ruta de los certificados antes generados.

```
...
ServerName  midominio.cl
DocumentRoot /var/www
SSLCertificateFile    /etc/apache2/ssl.crt/midominio.cl.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/midominio.cl.key
...
```

Habilitamos el sitio *default-ssl*

```
root@debian:/etc/apache2/sites-available# a2ensite default-ssl
```

Ahora comprobamos visitando el sitio en ambos servicios.

Figura 2: Sitio sin HTTPS

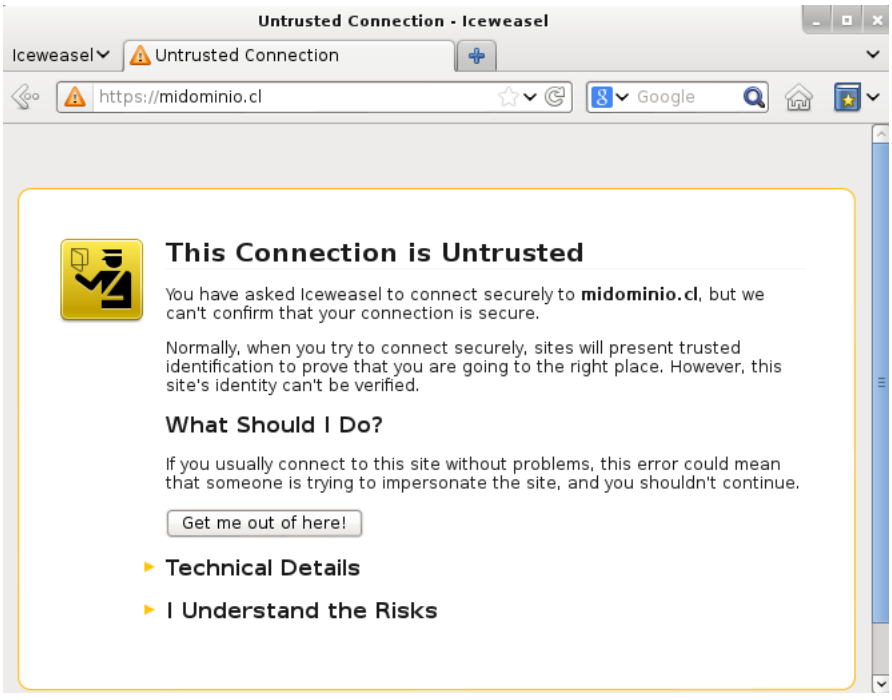

Figura 3: Con HTTPS indicando información del certificado

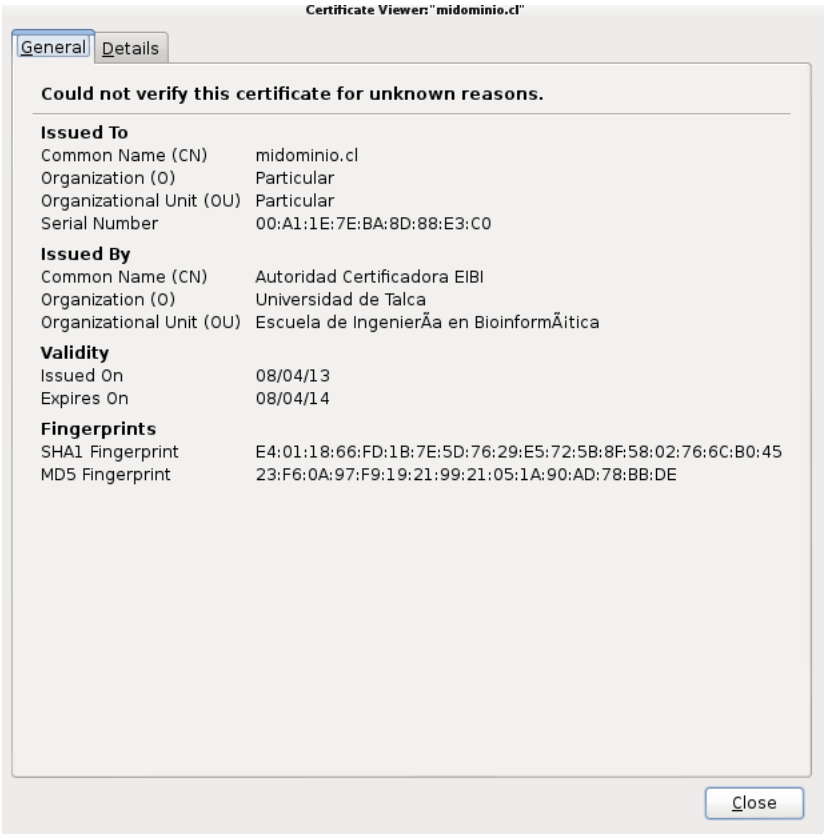

Figura 4: Información del certificado

Figura 5: Sitio con HTTPS