INSTALL APACHE2:

```
$ sudo apt-get install apache2
```

HTTPS Configuration:

The **mod_*ssl*** module is available in ***apache2-common*** package. To enable the ***mod_ssl*** module:

```
$ sudo a2enmod ssl
Module ssl installed; run /etc/init.d/apache2 force-reload to enable.

$ sudo /etc/init.d/apache2 force-reload
 * Reloading web server config apache2
```

Certificates:

To set up your secure server, use public key cryptography to create a public and private key pair. You need a key and a certificate to operate your secure server, which means that you can either generate a **self-signed certificate** or purchase a **CA-signed certificate**.

1. You can create your **own self-signed certificate**. Note, however, that self-signed certificates should not be used in most production environments. Self-signed certificates are not automatically accepted by a user's browser. Users are prompted by the browser to accept the certificate and create the secure connection.

2. You can send your certificate request (including your public key), proof of your company's identity, and payment to a **Certificate Authority** (CA). The CA verifies the certificate request and your identity, and then sends back a certificate for your secure server.

Once you have a self-signed certificate or a signed certificate from the CA of your choice, you need to **install** it on your secure server.

(1A) Create a self-signed certificate

1A.1 Generate a Private Key

To generate the *Certificate Signing Request* (*CSR)*, you should create your own key. In order to create the ***server key*** run the following command:

```
$ sudo openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.....................++++++
........++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
```

Then create a certificate signing request with it.

1A.2 Generate a CSR (Certificate Signing Request)

The following command will prompt for a series of things (country, state or province, etc.). Make sure that "**Common Name (eg, YOUR name)**" **matches the registered fully qualified domain name** of your box (or **your IP address** if you don't have one). If the website to be protected will be *https://www.server.com*, then enter *www.server.com* at this prompt. The default values for the questions ([AU], Internet Widgits Pty Ltd, etc.) are stored in ***/etc/ssl/openssl.cnf***.

```
$ sudo openssl req -new -key server.key -out server.csr
Enter pass phrase for server.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]:it
State or Province Name (full name) [Some-State]:italy
Locality Name (eg, city) []:rende
Organization Name (eg, company) [Internet Widgits Pty Ltd]:mycompany
Organizational Unit Name (eg, section) []:myunit
Common Name (eg, YOUR name) []:localhost
Email Address []:myemail

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

Now you are ready to sign the certificate signing request (see next step).

1A.3 Generating a Self-Signed Certificate

```
$ sudo openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
Signature ok
subject=/C=it/ST=italy/L=rende/O=mycompany/OU=myunit/CN=localhost/emailAddress=myemail
Getting Private key
Enter pass phrase for server.key:
```

1A.4 Skipt to step (2) but before...

Please check...you should have:
server.crt: The self-signed server certificate.
server.csr: Server certificate signing request.
server.key: The private server key

(1B) Generate your own CA (Certificate Authority)

Complete this section if you want to make a CA (Certificate Authority) and sign a server certificate with it. The **Common Name** (**CN**) of the **CA** and the **Server certificates must NOT match** or else a naming collision will occur and you'll get errors later on. In this step, you'll provide the CA entries. In this example, I just added "ca" to the CA's CN field, to distinguish it from the Server's CN field.

CA:
Common Name (CN): www.somesite.edu CA
Organization (O): Somesite
Organizational Unit (OU): Development
Server:
Common Name (CN): www.somesite.edu
Organization (O): Somesite
Organizational Unit (OU): Development

If you don't have a fully qualified domain name, you should use the IP that you'll be using to access your SSL site for Common Name (CN).

1B.1 Generate a CA private key

```
$ sudo openssl genrsa -des3 -out ca.key 4096
Generating RSA private key, 4096 bit long modulus
..................................................................................++
..............................................................................................................++
e is 65537 (0x10001)
Enter pass phrase for ca.key:
Verifying - Enter pass phrase for ca.key:
```

1B.2 Generate a request for signing (csr)

```
$ sudo openssl req -new -x509 -days 365 -key ca.key -out ca.crt
Enter pass phrase for ca.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]:it
State or Province Name (full name) [Some-State]:italy
Locality Name (eg, city) []:cs
Organization Name (eg, company) [Internet Widgits Pty Ltd]:companyca
Organizational Unit Name (eg, section) []:unitca
Common Name (eg, YOUR name) []:localhostca
Email Address []:camail
```

1B.3 Generate a server key

```
$ sudo openssl genrsa -des3 -out server.key 4096

Generating RSA private key, 4096 bit long modulus
...........................................................................................................................................................++
.............................................++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:_
```

1B.4 Generate a request for signing (csr)

After generating a server key, generate a request that you want it signed (the .csr file) by a Certificate Authority (the one you just created in Step above.) Think carefully when inputting a Common Name (CN) as you generate the .csr file below. This should match the DNS name, or the IP address you specify in your Apache configuration. If they don't match, client browsers will get a "domain mismatch" message when going to your https web server. If you're doing this for home use, and you don't have a static IP or DNS name, you might not even want worry about the message (but you sure will need to worry if this is a production/public server). For example, you could match it to an internal and static IP you use behind your router, so that you'll never get the "domain mismatch" message if you're accessing the computer on your home LAN, but will always get that message when accessing it elsewhere.

```
$ sudo openssl req -new -key server.key -out server.csr
```

```
Enter pass phrase for server.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]:it
State or Province Name (full name) [Some-State]:italy
Locality Name (eg, city) []:rende
Organization Name (eg, company) [Internet Widgits Pty Ltd]:mycomp
Organizational Unit Name (eg, section) []:myunit
Common Name (eg, YOUR name) []:localhost
Email Address []:mymail

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

1B.5 Sign the certificate signing request (csr) with the self-created (CA)

Note that 365 days is used here. After a year you'll need to do this again. Note also that I set the serial number of the signed server certificate to "01". Each time you do this, especially if you do this before a previously-signed certificate expires, you'll need to change the serial key to something else -- otherwise everyone who's visited your site with a cached version of your certificate will get a browser warning message to the effect that your certificate signing authority has screwed up -- they've signed a new key/request, but kept the old serial number. There are a couple ways to rectify that. crl's (certificate revocation list) is one method, but beyond the scope of the document. Another method is for all clients which have stored the CA certificate to go into their settings and delete the old one manually. But for the purposes of this document, we'll just avoid the problem. (If you're a sysadmin of a production system and your server.key is compromised, you'll certainly need to worry.)

The command below does a number of things. It takes your signing request (csr) and makes a one-year valid signed server certificate (crt) out of it. In doing so, we need to tell it which Certificate Authority (CA) to use, which CA key to use, and which Server key to sign. We set the serial number to 01, and output the signed key in the file named server.crt. If you do this again after people have visited your site and trusted your CA (storing it in their browser), you might want to use 02 for the next serial number, and so on. You might create some scheme to make the serial number more "official" in appearance or makeup but keep in mind that it is fully exposed to the public in their web browsers, so it offers no additional security in itself.

```
$ sudo openssl x509 -req -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial
01 -out server.crt
```

```
Signature ok
subject=/C=it/ST=italy/L=rende/O=mycomp/OU=myunit/CN=localhost/emailAddress=mymail
Getting CA Private Key
Enter pass phrase for ca.key:
```

1B.6 Skipt to step (2) but before...

Please check...you should have:
server.crt: The self-signed server certificate.
server.csr: Server certificate signing request.
server.key: The private server key
ca.crt: The Certificate Authority's own certificate.
ca.key: The key which the CA uses to sign server signing requests.

The CA files are important to keep if you want to sign additional server certificates and preserve the same CA. You can reuse these so long as they remain secure, and haven't expired.

(2) Setting up SSL

Preliminaries

If you have a registered DNS name, be sure that you properly set it up. On the Gnome console: System->Administration->Networking:General. Your host/domain name here should match the one you'll be using in later steps. You can also edit /etc/hosts directly if you're comfortable with that route.

Installing the Private Key and Certificate

When Apache with ***mod_ssl*** is installed, it creates several directories in the Apache *config* directory. The location of this directory will differ depending on how Apache was compiled.
This step suggests putting certificate-related files in this location: ***/etc/apache2/ssl***.
If the "ssl" directory **doesn't already exist** there, go ahead and **mkdir it** now.

```
$ sudo mkdir /etc/apache2/ssl
```

Then copy the ***server.key*** and ***server.crt*** files into position:

```
$ sudo cp server.key /etc/apache2/ssl
$ sudo cp server.crt /etc/apache2/ssl
```

Enable ssl

You'll want to run the ***/usr/sbin/a2enmod*** script. If you look at this script, it's simply a general purpose utility to establish a symlink between a module in ***/etc/apache2/mods-available*** to ***/etc/apache2/mods-enabled*** (or give a message to the effect that a given module doesn't exist or that it's already symlinked for loading).

```
$ sudo a2enmod ssl
```

Create a stub SSL conf. file and establish a necessary symlink

The first command copies the default configuration file for port 80, to use it as a stub configuration file for 443. The second command establishes a symlink from the '*available*' *ssl* file to the '*enabled*' file. The symlinking methodology between those two directories is similar in philosophy to mods-available and mods-enabled (previous step). The general idea is that enabled files exist as symlinks created to their available counterparts.

```
$ sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-available/ssl

$ sudo ln -s /etc/apache2/sites-available/ssl /etc/apache2/sites-enabled/ssl
```

Set up the document roots

The default location for HTML pages with an initial install of Ubuntu is **/var/www** and there exists no separate place for ssl files. I prefer to serve up basic HTML pages in /var/www/html and SSL pages in /var/www-ssl/html.

```
cd /var/www
mkdir html
cd /var
mkdir www-ssl
cd www-ssl
mkdir html
```

Configure virtual hosts

su to the superuser and make a backup of the original Apache configuration file. Call it whatever you want. My practice is to add "_original" to any default configuration file before I make changes -- in case I need to revert. You should not make a backup of the following file in the sites-enabled directory, since both the original and backup will be loaded when you restart Apache. Also note that a symlink exists from /etc/apache2/sites-enabled/000-default to /etc/apache2/sites-available/default.

```
$ sudo su
$ cd /etc/apache2/sites-available
$ cp /etc/apache2/sites-available/default default_original
```

Now you need to declare the IP of your box (or FQDN/DNS name) and document roots you created in a previous step.

Configure default and ssl file

To **configure HTTP over port 80** (**edit /etc/apache2/sites-available/default**):

```
NameVirtualHost *:80

<VirtualHost *:80>
ServerName localhost
DocumentRoot /var/www/html/
```

(Note: Use your assigned IP or DNS name followed with ":80" if you have one for ServerName).

Similar procedure for **HTTPS over port 443** (**edit /etc/apache2/sites-available/ssl**):

```
NameVirtualHost *:443

<VirtualHost *:443>
ServerName localhost
DocumentRoot /var/www-ssl/html/
```

(Note: Again, use your assigned IP or a DNS name followed with ":443" if you have one for ServerName.)

Instruct Apache to listen to 443

Go to ***/etc/apache2/ports.conf*** and check whether it looks like (starting with Ubuntu 7.10) the ***ports.conf*** may already have an *IfModule* clause in it for the SSL portion. If you see this, you can just leave it as-is:

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

Turn on the SSL engine

For example, in the middle of /etc/apache2/sites-available/ssl file, after the commented area which says "# Possible values include: debug, info, notice, warn, error, crit..." add the following.

```
SSLEngine On
SSLCertificateFile /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
```

When starting and stopping Apache there may be a complaint such as "Could not determine the server's fully qualified domain name, using 127.0.1.1 for ServerName". You may encounter this if you don't have a DNS name for your server, and are just using an IP. If this applies to you, go into your /etc/hosts file and make the following changes.

```
127.0.0.1 localhost localhost.localdomain {your system name}
127.0.1.1 {your system name}
{static IP if you you have one} {fully qualified DNS host name if you have one}
```

If you don't have a fully qualified domain name (FQDN) for your box, you may need to make an additional tweak. In your ***/etc/apache2/apache2.conf*** file, you may want to add the following line at the very end of the file if Apache is still complaining about lacking a fully qualified domain name at startup: ***ServerName localhost***.

(3) Test it:

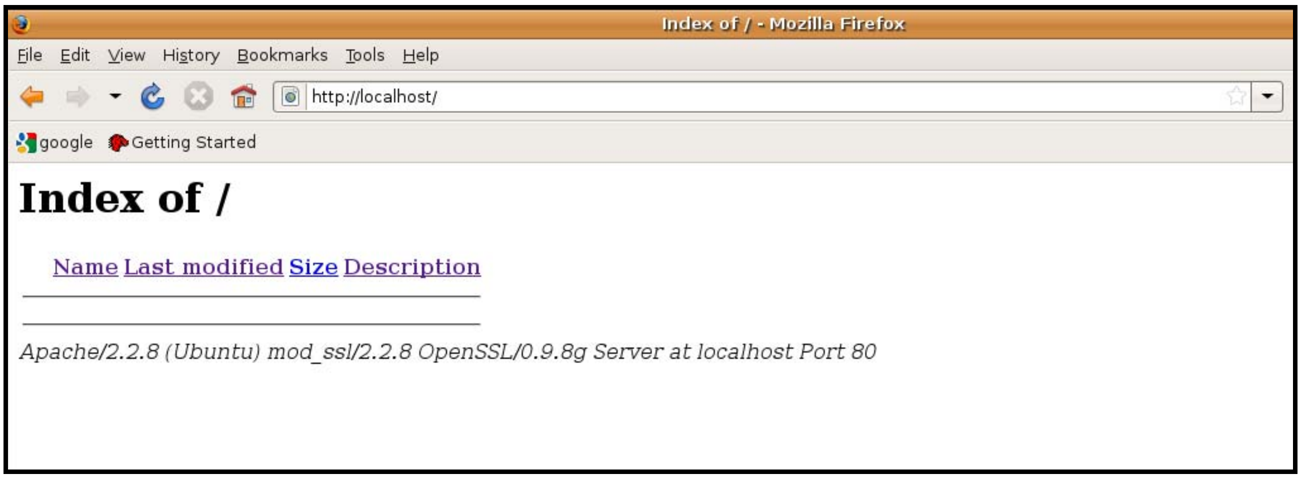
```
$ sudo /etc/init.d/apache2 start
Apache/2.2.8 mod_ssl/2.2.8 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide the pass phrases.

Server localhost:443 (RSA)
Enter pass phrase:

OK: Pass Phrase Dialog successful.
```

http://localhost/
https://localhost/

(1A) → (2) → http://localhost/



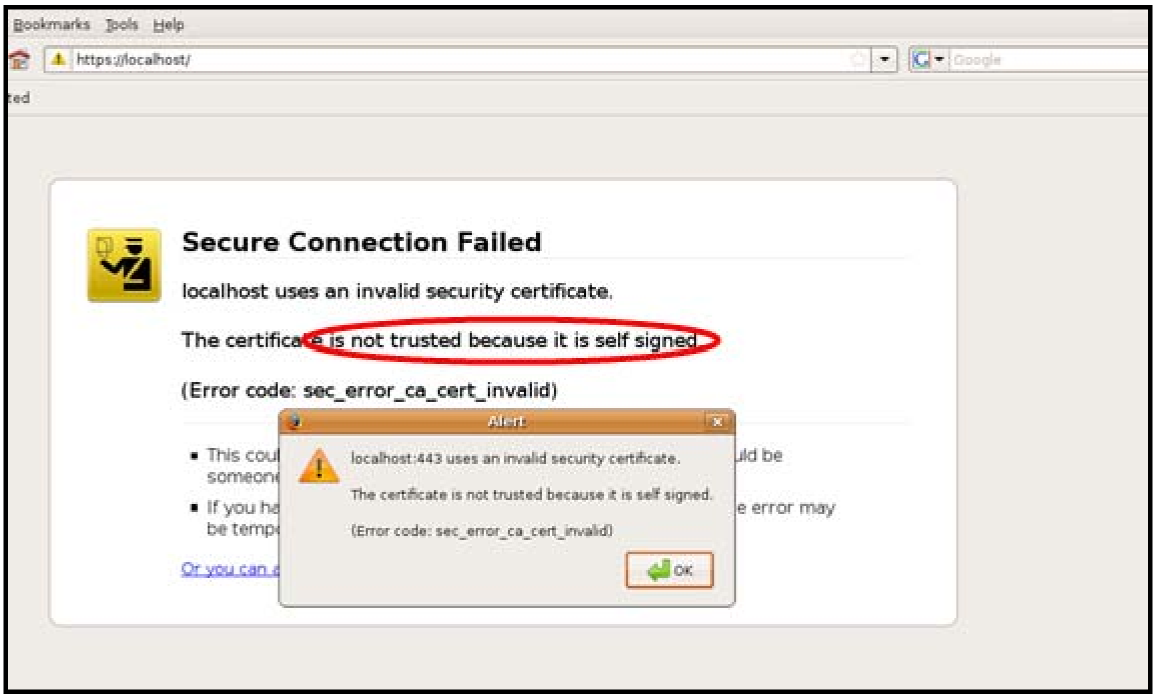
Index of / - Mozilla Firefox

Index of /

Name Last modified Size Description

Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g Server at localhost Port 80

(1A) → (2) → https://localhost/



Secure Connection Failed

localhost uses an invalid security certificate.

The certificate is not trusted because it is self signed.

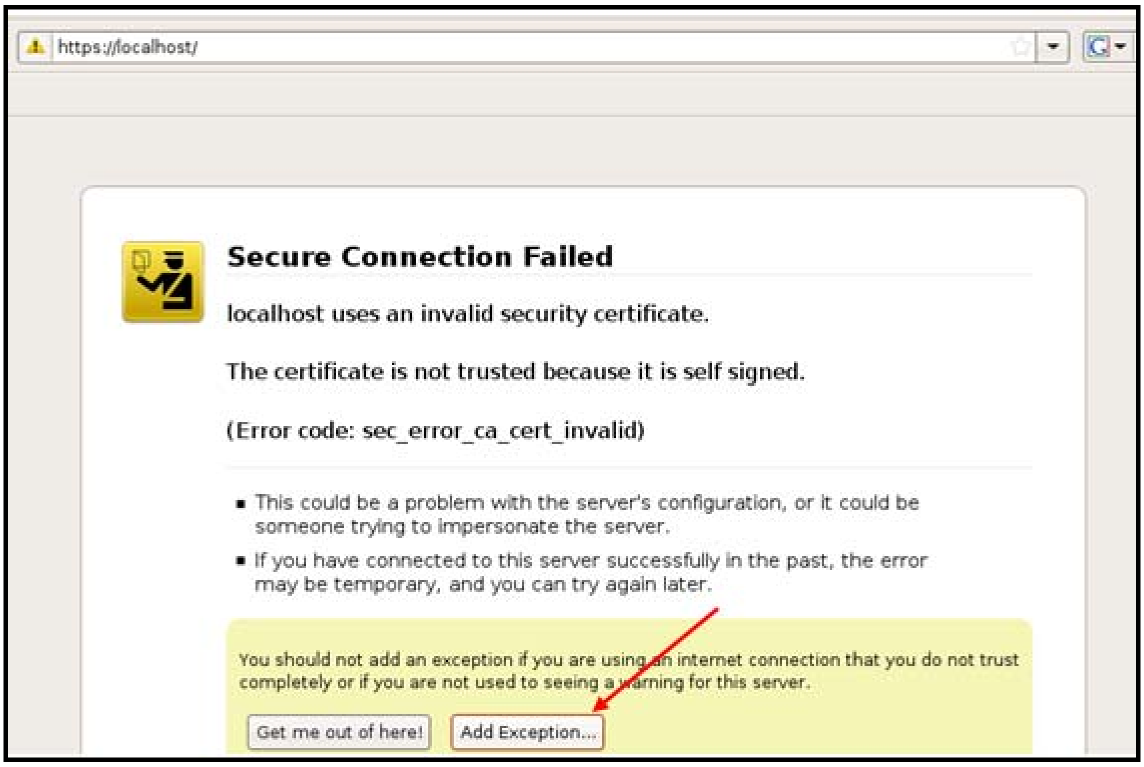
(Error code: sec_error_ca_cert_invalid)

Alert

localhost:443 uses an invalid security certificate.

The certificate is not trusted because it is self signed.

(Error code: sec_error_ca_cert_invalid)


https://localhost/

Secure Connection Failed

localhost uses an invalid security certificate.

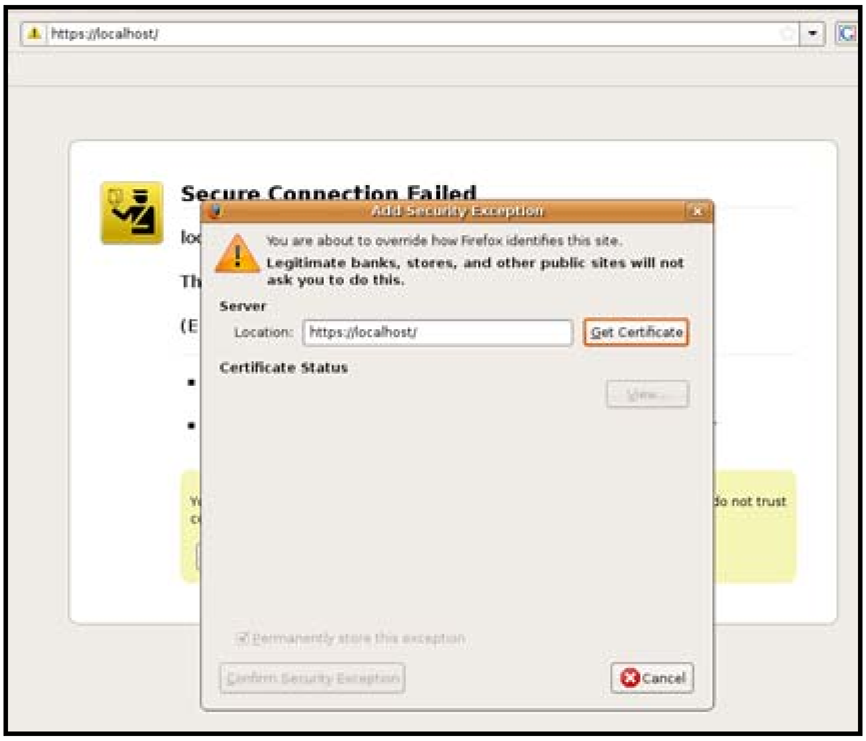
The certificate is not trusted because it is self signed.

(Error code: sec_error_ca_cert_invalid)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

You should not add an exception if you are using an internet connection that you do not trust completely or if you are not used to seeing a warning for this server.

If you add the Exception you can get the certificate:


https://localhost/

Secure Connection Failed

Add Security Exception

You are about to override how Firefox identifies this site.

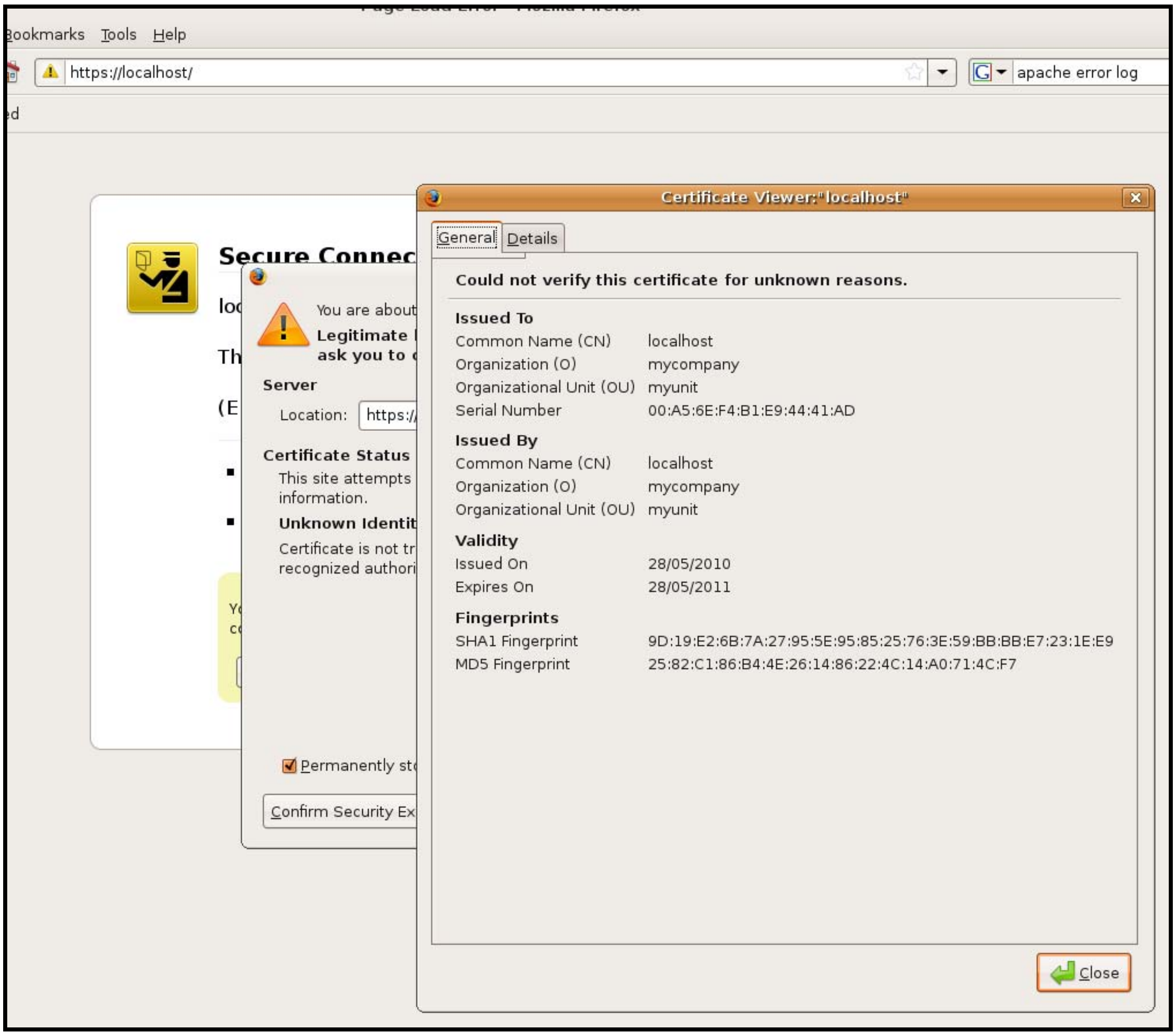
Legitimate banks, stores, and other public sites will not ask you to do this.

Server

Location: https://localhost/

Certificate Status

Then you can see properties and details of the certificate:



Could not verify this certificate for unknown reasons.

Issued To

Common Name (CN)	localhost
Organization (O)	mycompany
Organizational Unit (OU)	myunit
Serial Number	00:A5:6E:F4:B1:E9:44:41:AD

Issued By

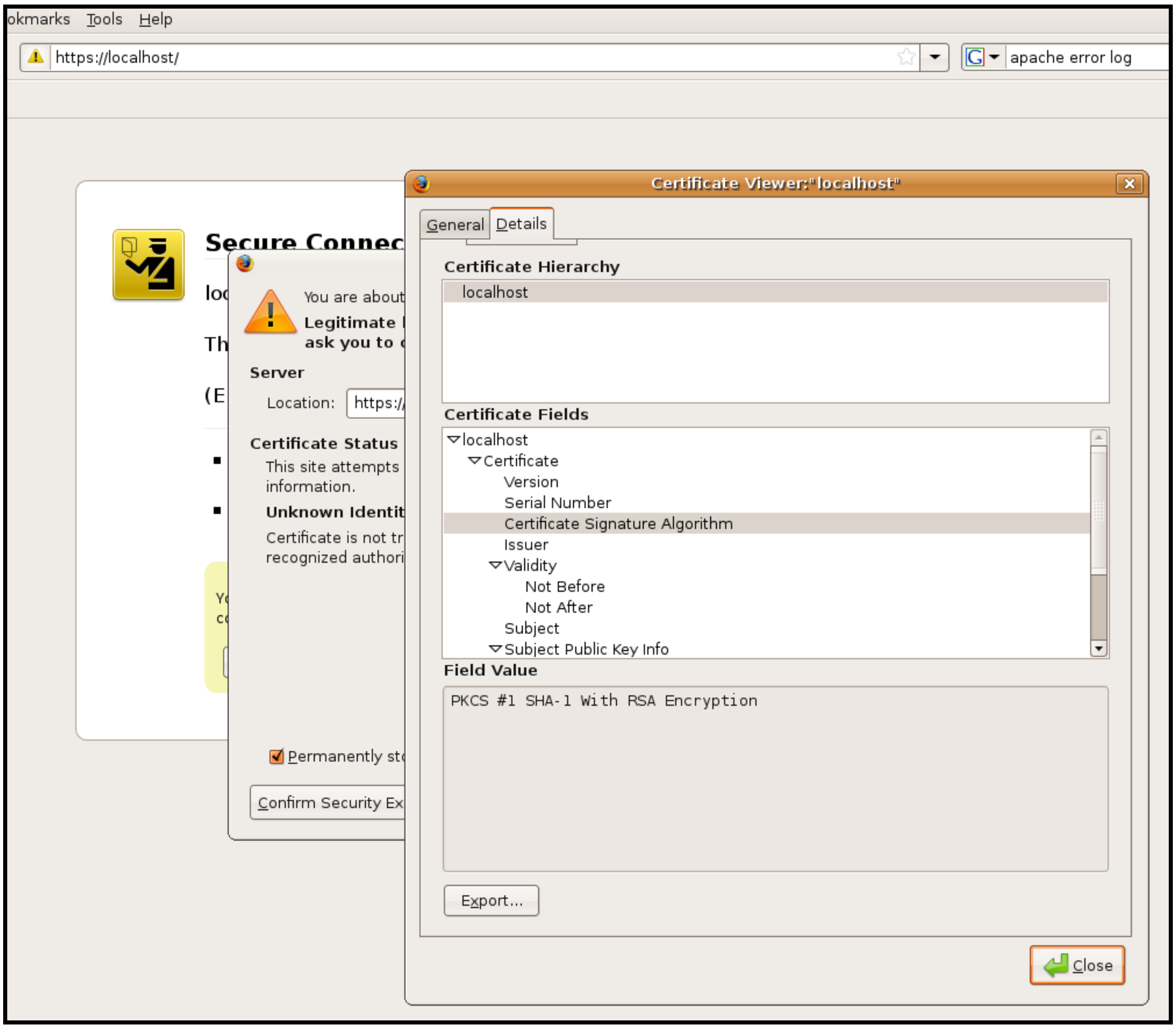
Common Name (CN)	localhost
Organization (O)	mycompany
Organizational Unit (OU)	myunit

Validity

Issued On	28/05/2010
Expires On	28/05/2011

Fingerprints

SHA1 Fingerprint	9D:19:E2:6B:7A:27:95:5E:95:85:25:76:3E:59:BB:BB:E7:23:1E:E9
MD5 Fingerprint	25:82:C1:86:B4:4E:26:14:86:22:4C:14:A0:71:4C:F7


okmarks Tools Help

https://localhost/

apache error log

Secure Connec

You are about

Legitimate

ask you to

Server

Location: https:/

Certificate Status

This site attempts information.

Unknown Identit

Certificate is not tr recognized authori

Permanently st

Confirm Security Ex

Certificate Viewer:"localhost"

General Details

Certificate Hierarchy

localhost

Certificate Fields

```
▽localhost
  ▽Certificate
      Version
      Serial Number
      Certificate Signature Algorithm
      Issuer
    ▽Validity
        Not Before
        Not After
      Subject
    ▽Subject Public Key Info
```

Field Value

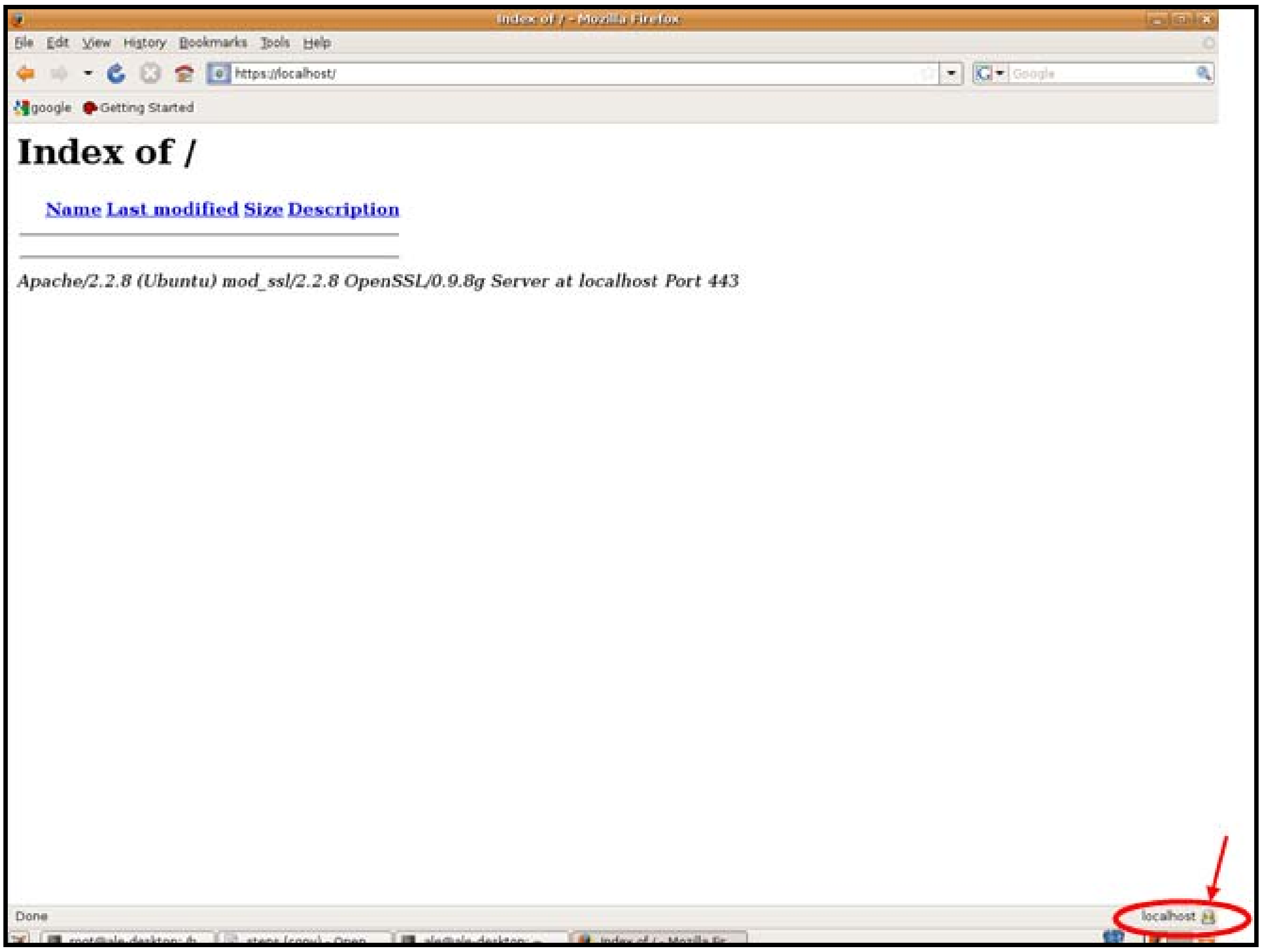
```
PKCS #1 SHA-1 With RSA Encryption
```

Export...

Close

You can also export the certificate (via the Export button).

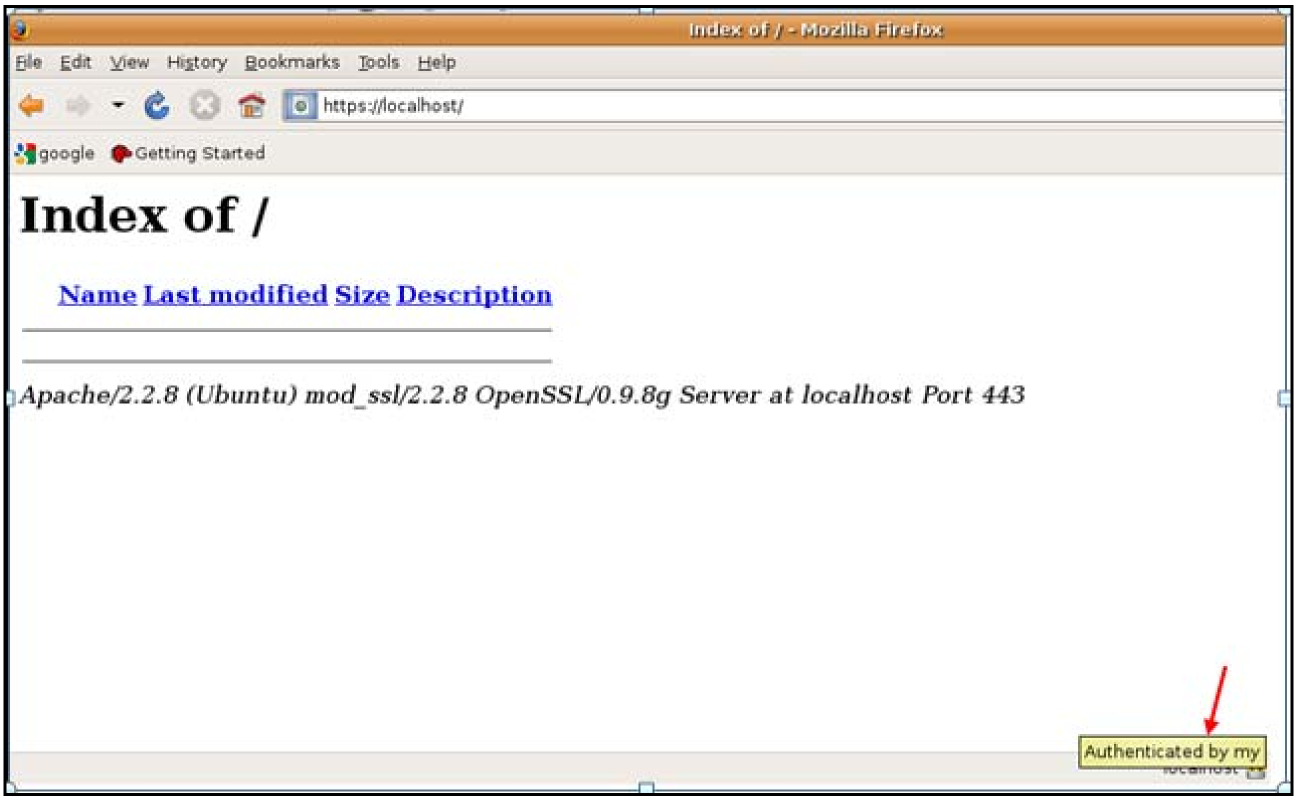
When accepting the certificate you are allowed to connect to https://localhost (you see the ***lock icon*** on the status bar of your browser).


Index of / - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://localhost/
google Getting Started
Done
localhost

Index of /

Name Last modified Size Description

Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g Server at localhost Port 443

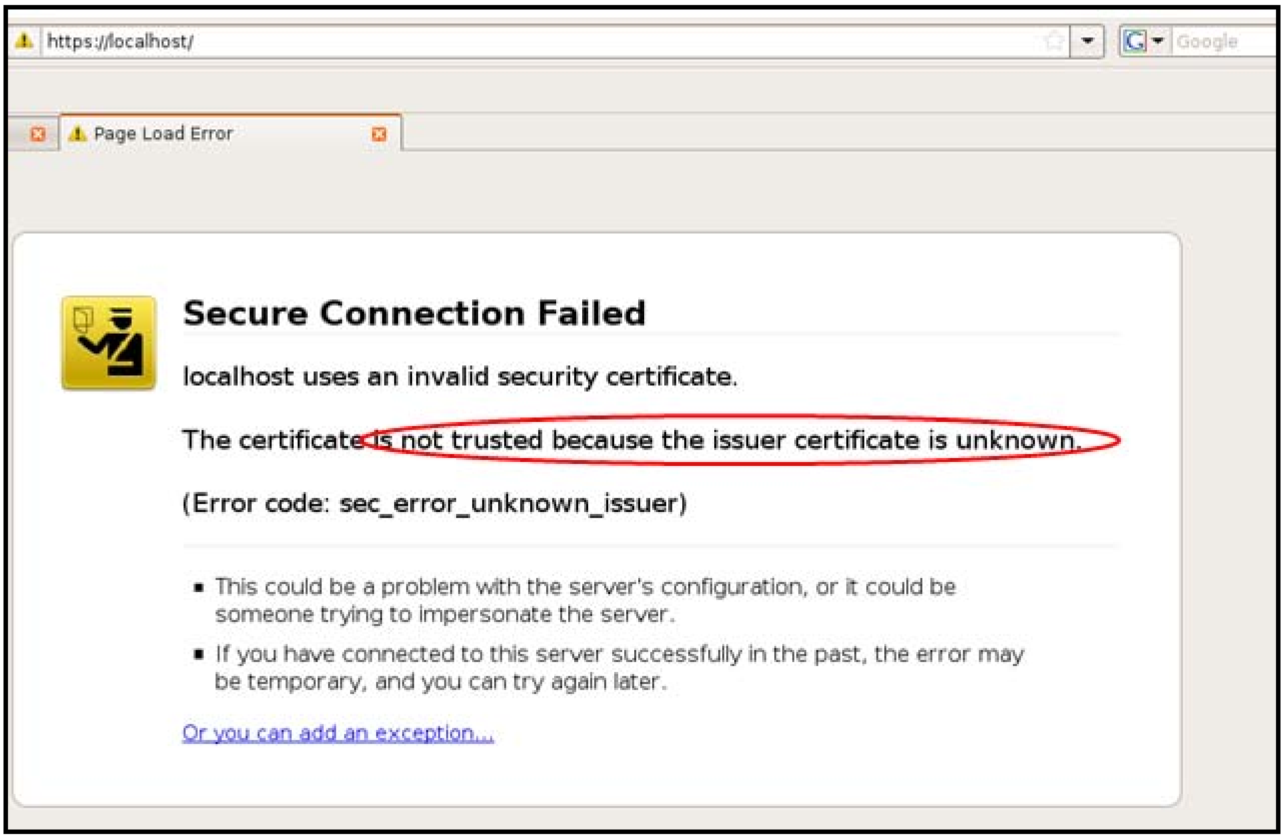

Index of / - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://localhost/
google Getting Started
Authenticated by my

Index of /

Name Last modified Size Description

Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g Server at localhost Port 443

(1B) → (2) → https://localhost/



https://localhost/

Page Load Error

Secure Connection Failed

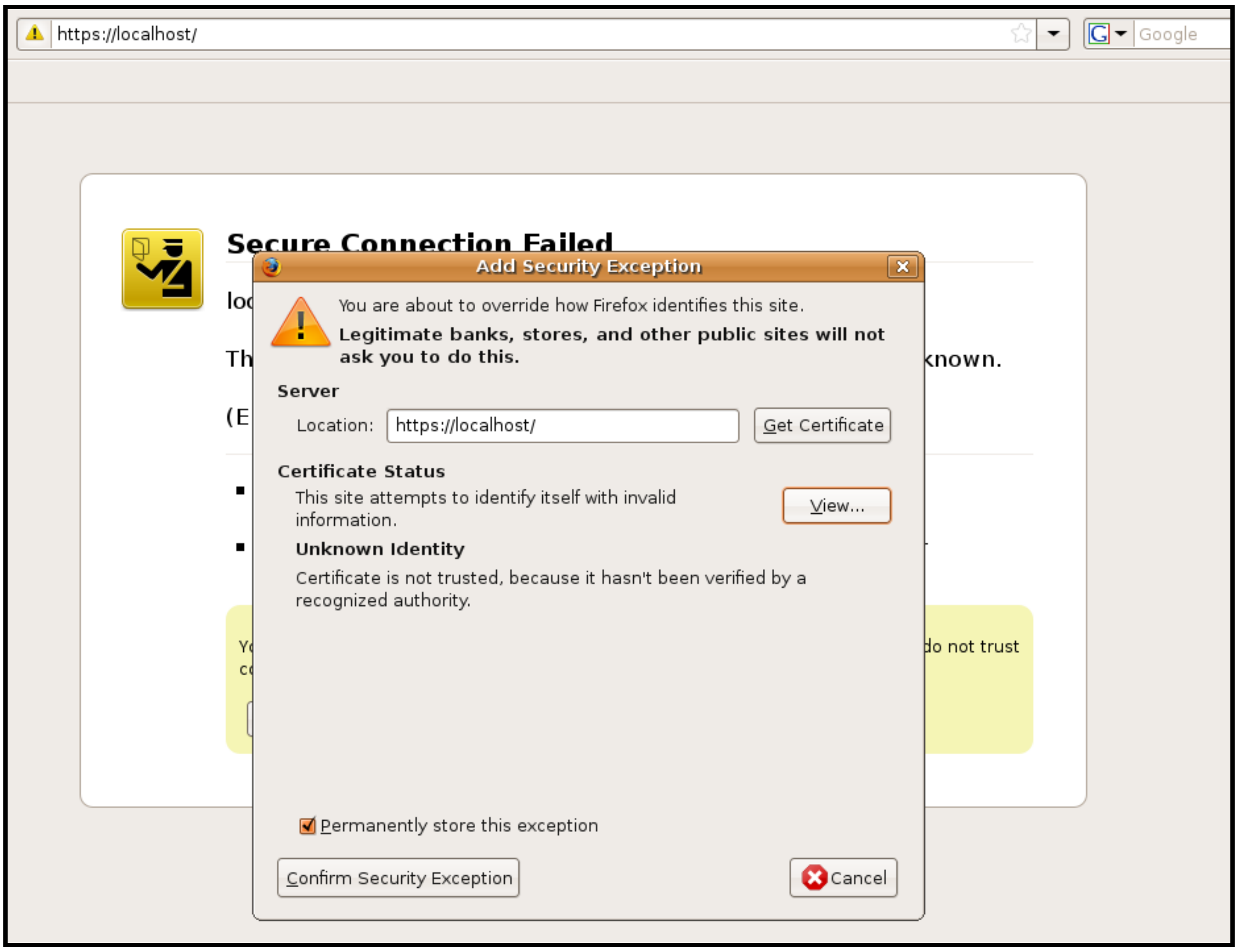
localhost uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.

(Error code: sec_error_unknown_issuer)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

Or you can add an exception...



https://localhost/

Secure Connection Failed

Add Security Exception

You are about to override how Firefox identifies this site.

Legitimate banks, stores, and other public sites will not ask you to do this.

Server

Location: https://localhost/ Get Certificate

Certificate Status

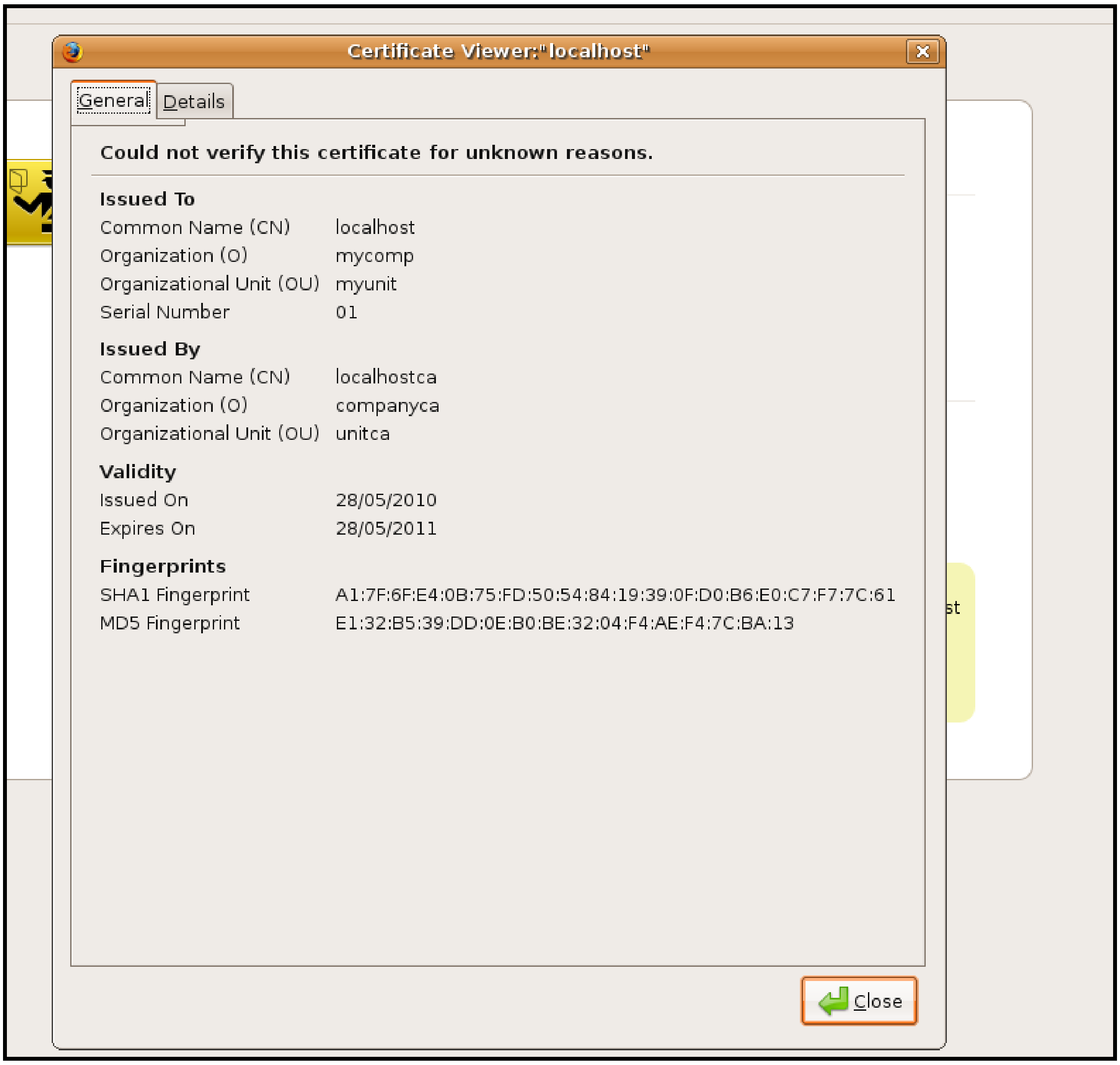
This site attempts to identify itself with invalid information. View...

Unknown Identity

Certificate is not trusted, because it hasn't been verified by a recognized authority.

☑ Permanently store this exception

Confirm Security Exception Cancel



Could not verify this certificate for unknown reasons.

Issued To

Common Name (CN)	localhost
Organization (O)	mycomp
Organizational Unit (OU)	myunit
Serial Number	01

Issued By

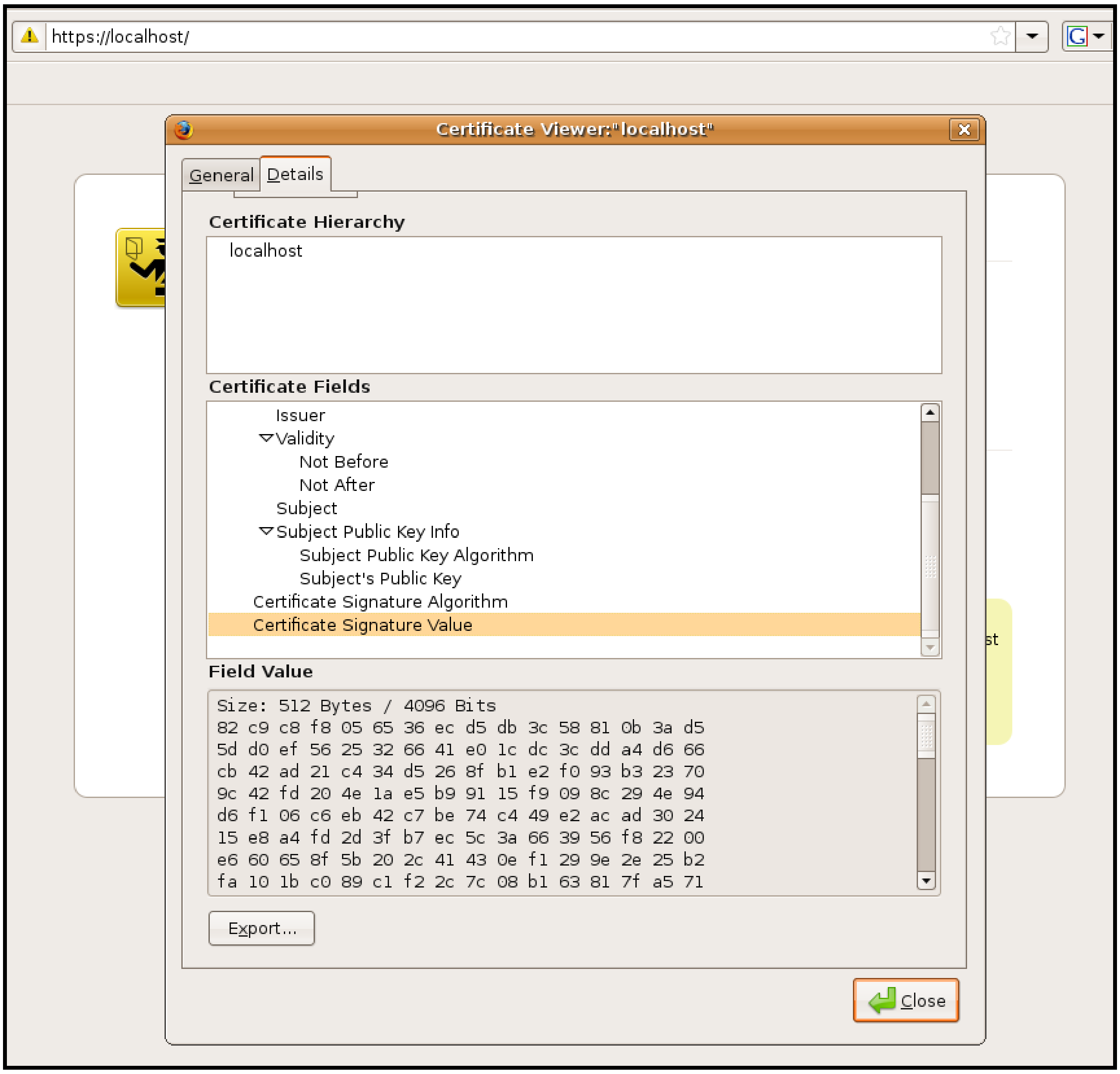
Common Name (CN)	localhostca
Organization (O)	companyca
Organizational Unit (OU)	unitca

Validity

Issued On	28/05/2010
Expires On	28/05/2011

Fingerprints

SHA1 Fingerprint	A1:7F:6F:E4:0B:75:FD:50:54:84:19:39:0F:D0:B6:E0:C7:F7:7C:61
MD5 Fingerprint	E1:32:B5:39:DD:0E:B0:BE:32:04:F4:AE:F4:7C:BA:13



Certificate Hierarchy

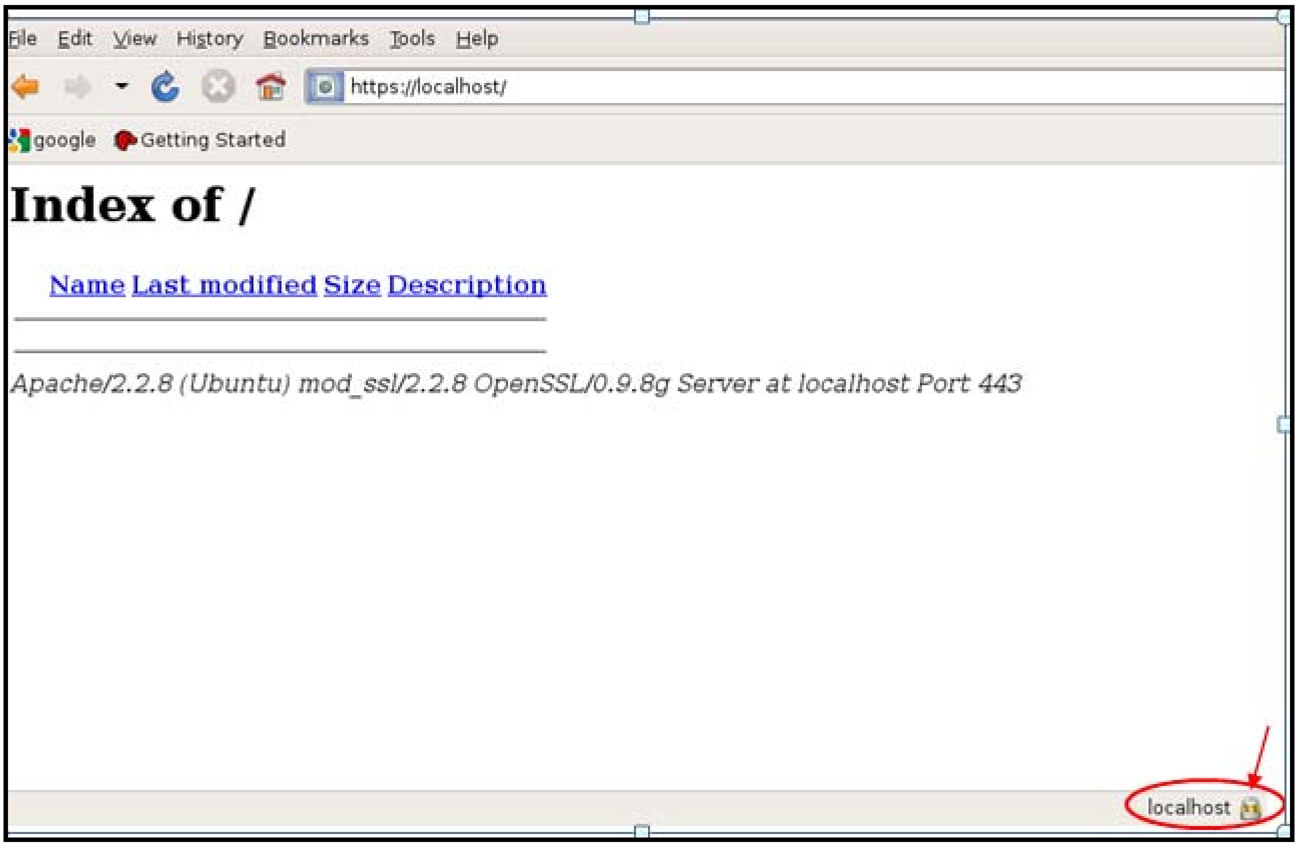
localhost

Certificate Fields

```
      Issuer
    ▽Validity
        Not Before
        Not After
      Subject
    ▽Subject Public Key Info
        Subject Public Key Algorithm
        Subject's Public Key
    Certificate Signature Algorithm
    Certificate Signature Value
```

Field Value

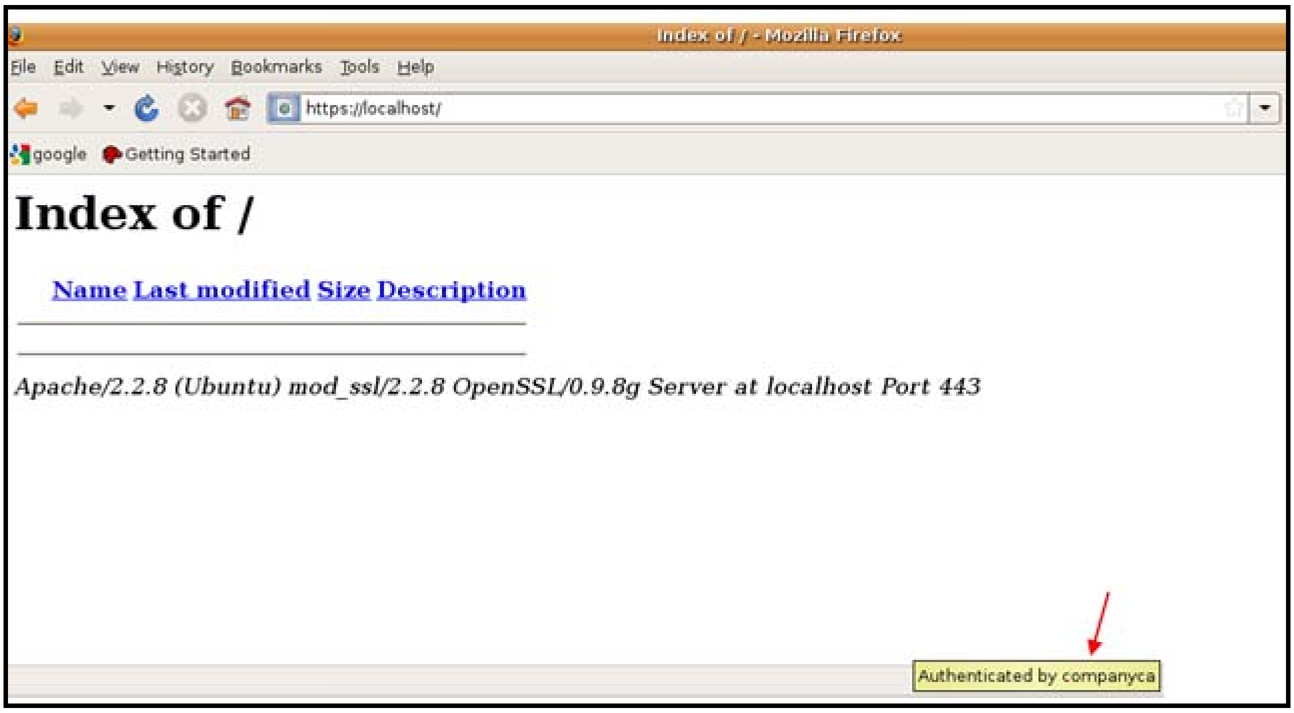
```
Size: 512 Bytes / 4096 Bits
82 c9 c8 f8 05 65 36 ec d5 db 3c 58 81 0b 3a d5
5d d0 ef 56 25 32 66 41 e0 1c dc 3c dd a4 d6 66
cb 42 ad 21 c4 34 d5 26 8f b1 e2 f0 93 b3 23 70
9c 42 fd 20 4e 1a e5 b9 91 15 f9 09 8c 29 4e 94
d6 f1 06 c6 eb 42 c7 be 74 c4 49 e2 ac ad 30 24
15 e8 a4 fd 2d 3f b7 ec 5c 3a 66 39 56 f8 22 00
e6 60 65 8f 5b 20 2c 41 43 0e f1 29 9e 2e 25 b2
fa 10 1b c0 89 c1 f2 2c 7c 08 b1 63 81 7f a5 71
```



Index of /

Name Last modified Size Description

Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g Server at localhost Port 443

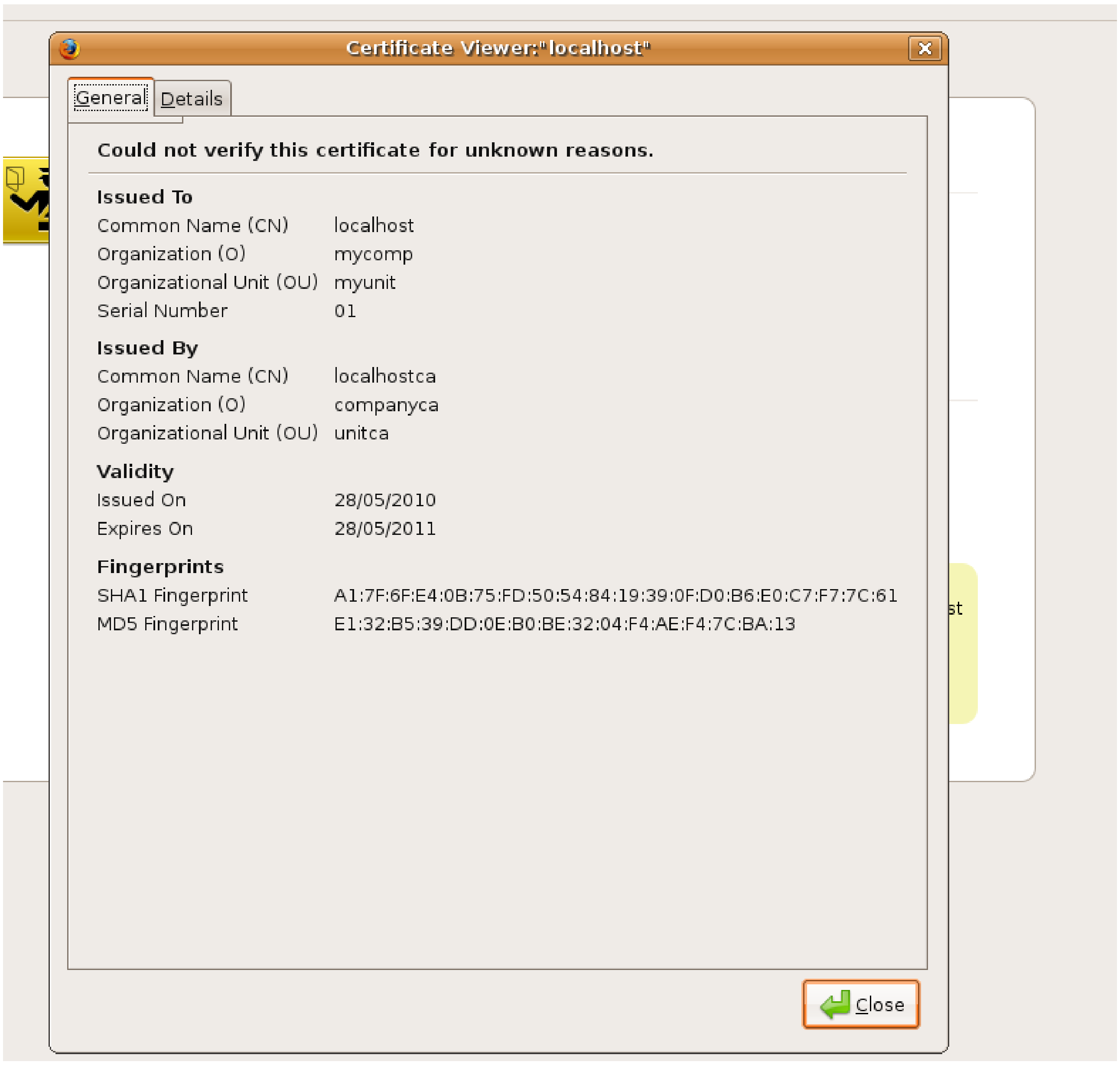

Index of / - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://localhost/
google
Getting Started

Index of /

Name Last modified Size Description

Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g Server at localhost Port 443

Authenticated by companyca



Could not verify this certificate for unknown reasons.

Issued To

Common Name (CN)	localhost
Organization (O)	mycomp
Organizational Unit (OU)	myunit
Serial Number	01

Issued By

Common Name (CN)	localhostca
Organization (O)	companyca
Organizational Unit (OU)	unitca

Validity

Issued On	28/05/2010
Expires On	28/05/2011

Fingerprints

SHA1 Fingerprint	A1:7F:6F:E4:0B:75:FD:50:54:84:19:39:0F:D0:B6:E0:C7:F7:7C:61
MD5 Fingerprint	E1:32:B5:39:DD:0E:B0:BE:32:04:F4:AE:F4:7C:BA:13