# Apache 2.0 with SSL / TLS protocol: Step by step instructions (continued Part I)

Before running Apache for the first time, we also need to provide the initial configuration and join some sample web content. At the very least, we need to follow these steps (as Root):

**Review the beginning of Part I**

**Configure SSL / TLS**

Before running Apache for the first time, we also need to provide the initial configuration and join some sample web content. At the very least, we need to follow these steps (as Root):

**1. Create some sample web content that will be met via SSL / TLS** :

```
umask 022 mkdir /www echo " Test works." > /www/index.html chown -R root:sys /ww
```

**2. Replace the default Apache configuration file** (usually found in /usr/local/apache2/conf/httpd.conf ) with a new one, using the following content (optimizing security and performance ):

```
# ================================================ # Basic settings # =========
```

**3. Note:** You should change some values ??in the above configuration file. Such as the name of the web server, the e-mail address of the administrator. etc .

**4. Re-structure the web server's private key directory** , certificates, and certificate revocation list (CRLs).

```
umask 022 mkdir /usr/local/apache2/conf/ssl.key mkdir /usr/local/apache2/conf/ss
```

**5. Create a ' *self-signed* ' service** (it will only be used for testing purposes - your real certificates should come from an appropriate CA like *Verisign* ):

```
openssl req -new -x509 -days 30 -keyout /usr/local/apache2/conf/ssl.key/server.k
```

**Check the installation**

At this point, we can start Apache supporting SSL / TLS as follows:

```
/usr/local/apache2/bin/apachectl startssl Apache/2.0.52 mod_ssl/2.0.52 (Pass Phra
```

After the server starts, we can try to connect to it by pointing to the web browser with a URL that looks like this: https: //name.of.the.web.server (in their case I am https: //www.seccure.lab )

In a few minutes, we will see a warning saying there is a problem with verifying the web server validation we want to access. The illustration in Figure 3 is an example from MS Internet Explorer 6.0.

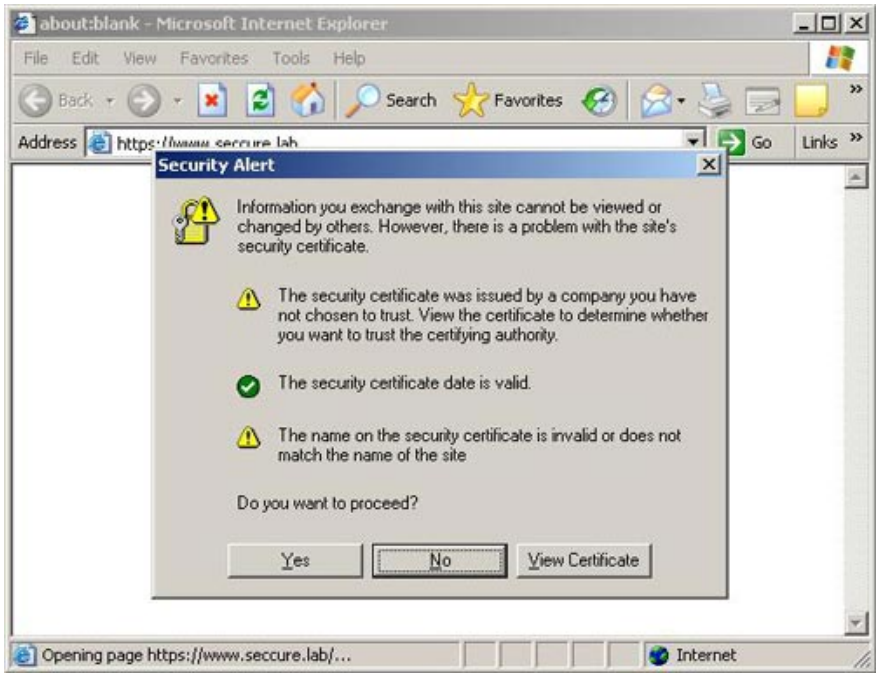


**Figure 3** .Prior warning of IE 6.0.

The appearance of the above warning properly perfectly! We should receive this warning for two reasons:

1. The web browser does not know the *Certificate Authority* , provided by the web server certificate (and cannot know, because we are using a *selfsigned certificate* ).
2. CN ( *Common Name* ) - The generic name given by the unconnected certificate of the website name - at the time it is a read -*only certificate* ( *Text-only Certificate)* , and it will be the full domain name of the web server ( eg: www.seccure.lab )

After executing with Internet Explorer, we should see the following web content, as shown in Figure 4:

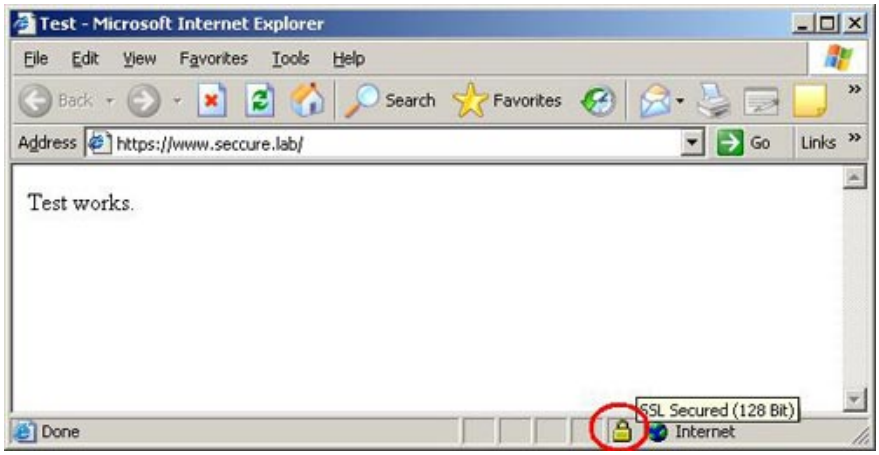


**Figure 4.** Website template work of SSL.

One thing to note, there's a golden key at the end of the web browser. That means the SSL connection has been successfully established. The 128-bit value indicates that the symmetric key is used to encrypt the transaction with a 128-bit length. And it is strong enough (at least at this time) to protect network traffic from unauthorized intrusion.

If we double-click the lock icon, we will see the properties of the website certificate, as shown in *Figure 5* :

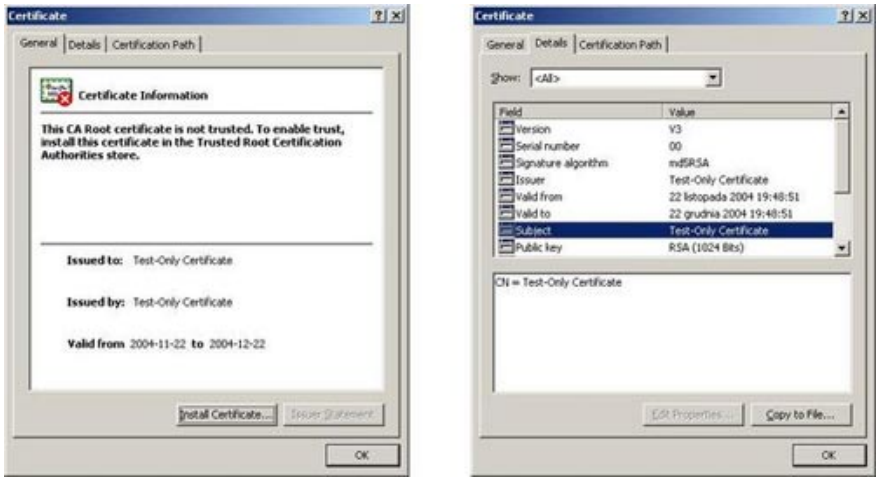


**Figure 5.** Details of the self -*signed certificate* ( *sekf-signed certificate* )

**Disentangle**

If for some reason we can't access the website, there's a useful diagnostic function ' *s_client* ', located in the OpenSSL library. It can be used to troubleshoot SSL / TLS connections. The following example shows how to use this function:

```
/usr/bin/openssl s_client -connect localhost:443 CONNECTED(00000003) depth=0 /CN=
```

The *s_client* function has many useful options. Such as turning *on / off* ( *on / off* ) a specific protocol ( -ssl2, -ssl3, -tls1 ). Then restart Apache and check the log files ( / usr / local / apache2 / logs / ) for more information.

We can also use *Ethereal* or *ssldump* . We can passively see SSL Handshake notifications, and try to find out the reason for the error without these tools.

A small screen executing this on Ethereal is depicted in *Figure 6* .

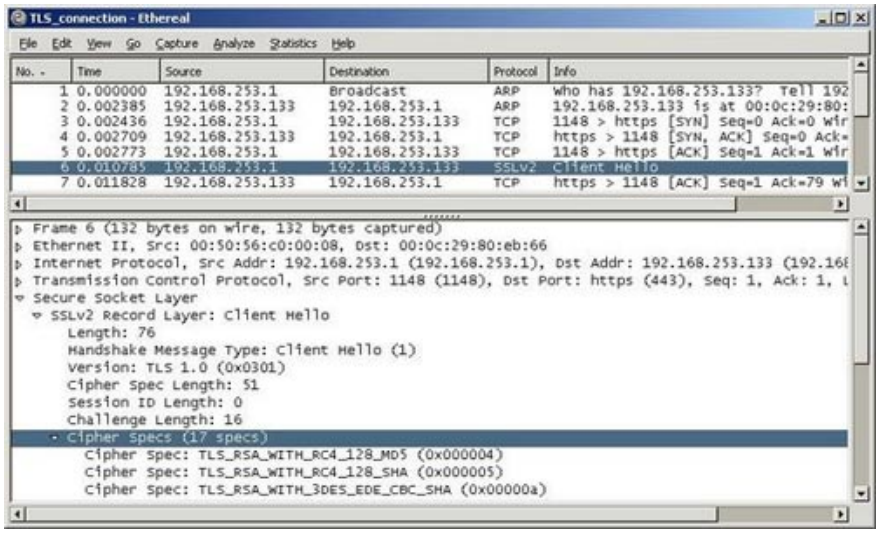


**Figure 6.** Ethereal screen with SSL Handshake method.

**Conclusion section I**

Installing and running Apache 2 secure with the SSL protocol and a sample certificate has ended in part one of this series. In part two, you will be introduced to the security settings and implementation for mod_ssl, as well as the process of creating an appropriate web server certificate.

**See Part II**