# Apache 2 supports SSL / TLS: Step-by-step instructions (continued Part II)

The third method of this signed certificate can be used in the Intranets network just like all other organizations use, or plan to use their own certificate authentication. In this case, the local CA certificate must be installed on all web browsers connected to the security of w

**Review Part I**
**First review Part II**

**Method 3: Certificate signed by an internal CA**

The third method of this signed certificate can be used in the Intranets network just like all other organizations use, or plan to use their own certificate authentication. In this case, the local CA certificate must be installed on all web browsers connected to the web server security.

To use this method, we need to create the local CA's public / private key just like the CA certificate and storage for new keys.

*Note : The* internal CA should be created on a distributed server, not connected at all to the network. The operating system only allows access to those who have passed the test and the machine itself is placed in a physical security level. CA's private key is the most significant element of the entire PKI system. If this key is passed, all other certificates signed by the CA will also be considered as approved.

We will use the OpenSSL library to install the environment step by step as listed below. Of course, if we already have an internal CA, we can skip this section and switch to creating certificate requests for the web server.

1. Prepare the directory structure for the new CA ($ SSLDIR environment variable should be added to the corresponding boot, such as **/ etc / profile or /etc/rc.local** ):

```
export SSLDIR = $ HOME / shift
mkdir $ SSLDIR
mkdir $ SSLDIR / certs
mkdir $ SSLDIR / crl
mkdir $ SSLDIR / newcerts
mkdir $ SSLDIR / private
mkdir $ SSLDIR / requests
touch $ SSLDIR / index.txt
echo "01"> $ SSLDIR / serial
```

chmod 700 $ SSLDIR

2. Create the main OpenSSL configuration file **$ SSLDIR / openssl.cnf** with the following content (optimized for use with SSL web server):

```
# ==========================================
# OpenSSL configuration file
# ==========================================

RANDFILE = $ ENV :: SSLDIR / .rnd

[ca]
default_ca = CA_default

[CA_default]
dir = $ ENV :: SSLDIR
certs = $ dir / certs
new_certs_dir = $ dir / newcerts
crl_dir = $ dir / crl
database = $ dir / index.txt
private_key = $ dir / private / ca.key
certificate = $ dir / ca.crt
serial = $ dir / serial
crl = $ dir / crl.pem
RANDFILE = $ dir / private / .rand
default_days = 365
default_crl_days = 30
default_md = sha1
preserve = no
policy = policy_anything
name_opt = ca_default
cert_opt = ca_default

[policy_anything]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[req]
default_bits = 1024
default_md = sha1
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
x509_extensions = v3_ca
```

```
string_mask = nombstr

[req_distinguished_name]
countryName = Country Name (2 letter code)
countryName_min = 2country
Name_max = 2
stateOrProvinceName = State or Province Name (full name)
localityName = Locality Name (eg, city)
0.organizationName = Organization Name (eg, company)
organizationalUnitName = Organizational Unit Name (eg, section)
commonName = Common Name (eg, YOUR name)
commonName_max = 64
emailAddress = Email Addressemail
Address_max = 64

[usr_cert]
basicConstraints = CA: FALSE
# nsCaRevocationUrl = https: //url-to-exposed-clr-list/crl.pem

[ssl_server]
basicConstraints = CA: FALSE
nsCertType = server
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, nsSGC, msSGC
nsComment = "OpenSSL Certificate for SSL Web Server"

[ssl_client]
basicConstraints = CA: FALSE
nsCertType = client
keyUsage = digitalSignature, keyEnciphermen
textendedKeyUsage = clientAuth
nsComment = "OpenSSL Certificate for SSL Client"

[v3_req]
basicConstraints = CA: FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[v3_ca]
basicConstraints = critical, CA: true, pathlen: 0
nsCertType = sslCA
keyUsage = cRLSign, keyCertSign
extendedKeyUsage = serverAuth, clientAuth
nsComment = "OpenSSL CA Certificate"

[crl_ext]
basicConstraints = CA: FALSE
keyUsage = digitalSignature, keyEncipherment
nsComment = "OpenSSL generated CRL"
```

3. Now create the CA's public / private key pair and the self-signed CA certificate:

openssl req
-config $ SSLDIR / openssl.cnf
-new -x509 -days 3652
-sha1
-newkey rsa: 1024
-keyout $ SSLDIR / private / ca.key
-out $ SSLDIR / ca.crt
-subj '/ O = Seccure / OU = Seccure Root CA'

4. It should be emphasized that CA's private key (ca.key) can be protected by a hard-to-imagine passphrase and should have a longer time than normal certificates (such as 10). -30 years or more).

CA 'ca.crt' certificate should be published on the Intranet website and installed in any web browser that may need to be used. An example of the original CA certificate installed on Internet Explorer is shown in Figure 2:

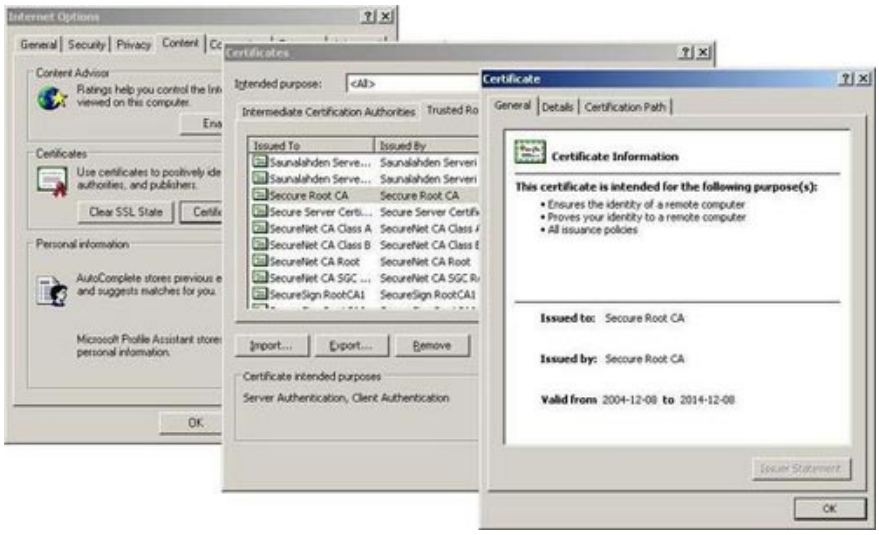


**Figure 2.** Example of installing the original CA certificate on Internet Explorer .

From this point, we can now use the internal CA to sign or revoke the certificate. To create a web server certificate, we should follow these steps:

1. Create the private / public key pair of the web server ( *server.key* ), and the certificate request ( *request.pem* ). The introduction should be implemented on the web server.

openssl req
-new
-sha1
-newkey rsa: 1024
-nodes
-keyout server.key

-out request.pem
-subj '/ O = Seccure / OU = Seccure Labs / CN = www.seccure.lab'

2. Copy the above certificate request (request.pem) into the $ SSLDIR / requests directory on the CA workstation (using removable media, such as USB drives)

3. Sign the certificate requirements as follows (only for executing on the CA workstation):

openssl ca
-config $ SSLDIR / openssl.cnf
-policy policy_anything
-extensions ssl_server
-out $ SSLDIR / requests / signed.pem
-infiles $ SSLDIR / requests / request.pem

4. The result of the above command is a signed certificate (signed.pem) placed in the **$ SSLDIR / newcerts directory** , and in the file **$ SSLDIR / signed.pem** . It includes a version of the TXT and PEM versions of the certificate. Because Apache wants to have a pure PEM format, we need to convert it as follows:

openssl x509
-in $ SSLDIR / requests / signed.pem
-out $ SSLDIR / requests / server.crt

5. Copy the signed PEM encoding certificate ( *server.crt* ) back to the web server machine.

Now the web server certificate is ready to use.

With the internal certificate authority, if the web server certificate is obtained, the CA is responsible for revoking the certificate. Then notify the user and the application program knows this certificate is no longer used.

To retrieve, we need to find the number sequence of the certificate we want in the $ SSLDIR / index.txt file. The following can be recalled:

openssl ca
-config $ SSLDIR / openssl.cnf
-revoke $ SSLDIR / newcerts / .pem

To create a CRL file (Certificate Revocation List), we can use the following statements:

openssl ca -config $ SSLDIR / openssl.cnf -gencrl -crlexts crl_ext -md sha1 -out $ SSLDIR / crl.pem

The above file should be published on CA's website and distributed to users. When distributing CRLs we should also use the Online Certificate Status Protocol (OCSP). For more information about OCSP, you can look in RFC 2560.

*Note that* some paths (including Firefox) only accept DER encoded CRLs, so before installing crl.pem file we must convert as follows:

openssl crl
-in $ SSLDIR / crl.pem
-out $ SSLDIR / revoke_certs.crl
-outform DER

Also note that in order for the web browser to check if the web server certificate is revoked, the 'Check for server certificate revocation' option should be checked in the Settings section of MS Internet Explorer. It is shown in Figure 3 and 4 as follows:

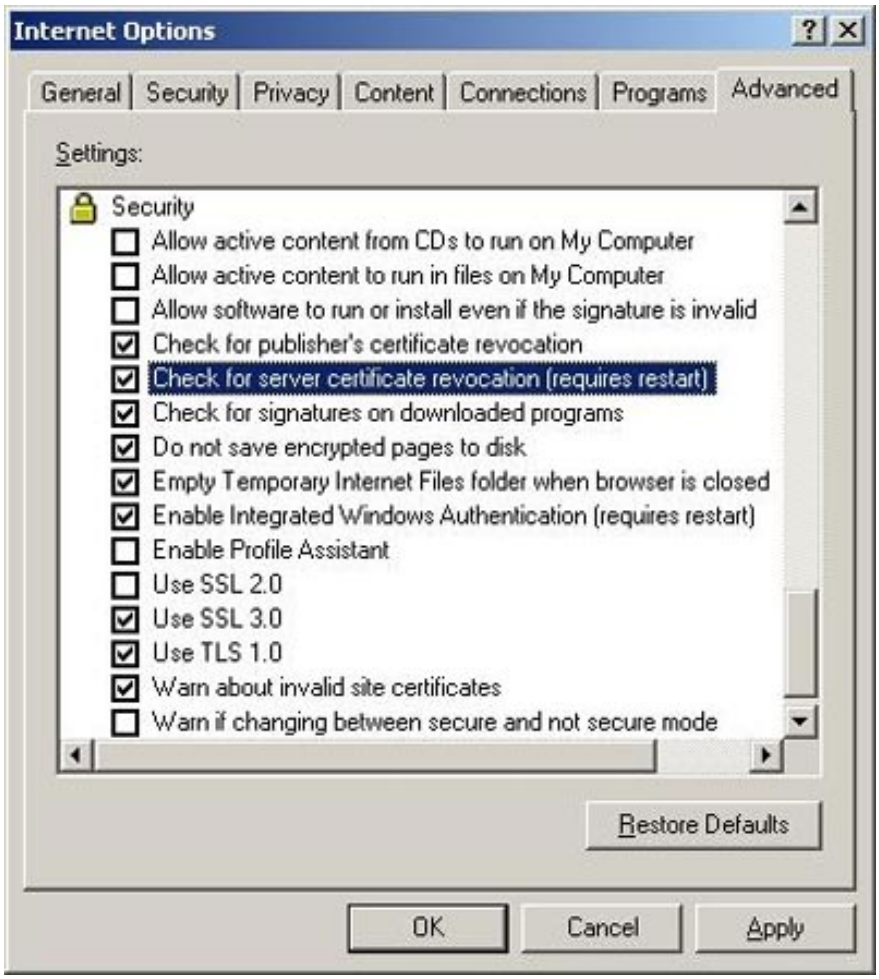


**Figure 3.** Configure Internet Explorer to check certificate revocation.


**Figure 4.** Internet Explorer's reply to a revoked certificate.

**Install the certificate**

At this point we can proceed to install the web server's private key (server.key) and certificate (server.crt) into the Apache environment:

install -m 600 -o root -g sys server.key /usr/local/apache2/conf/ssl.key/
install -m 644 -o root -g sys server.crt /usr/local/apache2/conf/ssl.crt/

We should also make sure that the instructions in Apache's configuration file are pointing to the above file (in httpd.conf).

SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server.key

The final step is to restart Apache for the changes to work:

/ usr / local / apache2 / bin / apachectl stop
/ usr / local / apache2 / bin / apachectl startssl

We can now check to see if the SSL website is accepted from the web browser. And whether web browsers can authenticate the web server successfully. This time there are no warnings as shown in Figure 5:


**Figure 5.** Secure connection with an appropriate certificate.

**Summary of Part II**

This section has shown you how to configure *mod_ssl* , create and use the web server's X.509v3 certificate. Next, in Part III and also at the end of this series, we will discuss client authentication through certificates as well as common errors and common types of attacks that can threaten the security of SSL communication.

***Please welcome to read Part III***