# Generating an X509v3 certificate from a Microsoft Certificate Services Enterprise Root or Subordinate CA for your Linux application

The reason you might be doing this is because Google developers have decided that Chrome will consider any certificate that doesn't support the 509v3 extensions (including SubjectAlternativeName standards) as invalid.

**NOTE**: Certificate Services created or upgraded from Windows Server 2003 & 2008 could likely still be sending SHA-1 signed certificates, and those too will be considered invalid by Chrome. You should upgrade Certificate Services if your implementation is still using SHA-1 signing. However, that upgrade process is far more detailed, and is not covered in this process.

1. The first step of this procedure would be to verify that you have already trusted the CA Certificate on all the machines that you intend to use this certificate from. For the purposes of this document, I will assume that you understand how to do that via Group Policy or some other means.

2. When creating a certificate from a Linux system, you can use the openssl command to generate the CSR while logged into the linux system as the root user (or "sudo"ed as root)

   **NOTE**: This procedure assumes that openssl is installed. Google "install openssl centos 6" if you are unsure on how install openssl in a Centos 6 server

   -
     ```
     cd /var/tmp
     mkdir /var/tmp/mycert
     cd /var/tmp/mycert
     openssl req -new -newkey rsa:2048 -nodes -keyout
     someapp.somedomain.key -out someapp.somedomain.csr
     cat someapp.somedomain.csr
     ```

3. In order for Microsoft Certificate services to support Subject Alternative Names (509 v3 certificate requirement), you must run the following commands on the Microsoft Certificate Services server:

   **NOTE**: If you have performed this step on the Microsoft Certificate Services server at some earlier date, it is not required a second time

   -
     ```
     certutil -setreg policy\EditFlags
     +EDITF_ATTRIBUTESUBJECTALTNAME2
     ```

   -
     ```
     net stop certsvc
     ```

- `net start certsvc`

  Kudos for helping me find how to enable this functionality to: https://ibrahimnore.wordpress.com/2012/10/21/generating-certificates-with-subject-alternative-names-sans-from-internal-certificate-authority-microsoft-ca/

4. Browse to the Microsoft Certificate services website from Chrome or another browser (your hostname need to be used in place of the example): http://certificateservices.somedomain.com/certsrv/certrqus.asp

   and click on "Advanced Certificate Request".

5. Copy the contents of the somedomain.csr file from the linux server (outputted in step 2 from the "cat" command)into the "Saved Request" text box
6. Change the "Certificate Template" to "Web server"
7. In the "Additional Attributes" Text box, use the following format, and identify ALL the hostnames that this server can respond on for this certificate (if the hostname differs from the website name)
   Examples include web clusters (use as many "dns=foo&" as necessary for each server, including the friendlyname users will browse to):

   ```
   san:dns=someapp.somedomain.com&dns=someapp1.somedomain.com&dns=someapp2.somedomain.com

   -or-

   san:dns=someapp.somedomain.com&dns=someapp*.somedomain.com
   ```

8. Click "Submit" to generate the certificate
9. Click the Radio Button for the "Base-64" version of the certificate
10. Click the hyperlink to download the Certificate and name the file "someapp.somedomain.crt"
11. SCP the file to the unix server into the /var/tmp/mycert directory (using some utility similar to WinSCP)
12. Copy the certificates to the appropriate folder depending on your apache setup:
    in my apache install, the certificate goes into:

    - `cp /var/tmp/mycert/someapp.somedomain.crt /etc/pki/tls/certs/`
    - `cp /var/tmp/mycert/someapp.somedomain.key /etc/pki/tls/private/`

    *** This information is based on /etc/httpd/conf.d/ssl.conf and the following configuration values:  SSLCertificateFile & SSLCertificateKeyFile.  You can place the files anywhere, but this is the suggested location from the ssl.conf file on my installation of CentOS 6.

13. Verify and or update the owner of these 2 files:
    - `ls -larth /etc/pki/certs/`
    - `ls -larth /etc/pki/private/`

    **NOTE**: If the files show "root:root" (the owner & group of the file) you will need to change the owner and group to apache so that the apache user can access these files
    - `chown apache:apache /etc/pki/certs/someapp.somedomain.com.crt`
    - `chown apache:apache /etc/pki/private/someapp.somedomain.com.key`

14. Backup the ssl.conf file:
    - `cp /etc/httpd/conf.d/ssl.conf /etc/httpd/conf.d/ssl.conf.backup`

15. Now modify the /etc/httpd/conf.d/ssl.conf to use these new files (use vi, nano, or your favorite editor) updating these lines as appropriate

    SSLCertificateFile /etc/pki/tls/certs/someapp.somedomain.com.crt
    SSLCertificateKeyFile /etc/pki/tls/private/someapp.somedomain.com.key

16. Restart the httpd service:
    - `service httpd restart`

    If you encounter an error starting the httpd service after making the changes, it likely means that there is a permissions issue for the apache user or an incorrect filename used in the ssl.conf file above. Validate the filename matches the files you saved in previous steps to the /etc/pki/tls/private/ and /etc/pki/tls/certs/ folder.

    Permissions/ownership can be changed like so:

    - `chown apache:apache /etc/pki/tls/private/someapp.somedomain.com.key`
    - `chown apache:apache /etc/pki/tls/certs/someapp.somedomain.com.crt`

You can revert the changes made by restoring the ssl.conf file from the backup made, and restarting the httpd service as shown above, if needed

17. Browse the website to verify that Chrome now considers the certificate valid

18. Clean up the /var/tmp/mycert folder now that your certificate is installed and working correctly
    - `rm -rf /var/tmp/mycert/`