**Purpose:**

Creating a secure login process involves three steps:

1. configure the SSL keystore password and save
2. add SSL (secure sockets layer transport) and https modules to start.d (recommended) or to start.ini
3. modify start.d (or start.ini) for ssl listen port and Keystore password

This process for the **first step** follows the X.509 documentation distributed by UBC IT ([how to obtain, deploy and verify an X.509 certificate](https://confluence.it.ubc.ca/display/ITSecurity/how+to+obtain%2C+deploy+and+verify+an+X.509+certificate) ) created by [Luca Filipozzi](https://confluence.it.ubc.ca/display/~lucaf), and last modified by [Aaron Heck](https://confluence.it.ubc.ca/display/~aheck) on [Aug 12, 2020](https://confluence.it.ubc.ca/pages/diffpagesbyversion.action?pageId=74423506&selectedPageVersions=59&selectedPageVersions=60)

This process implements a reverse proxy Apache server to isolate the applications from the user.

**Background**

This guide demonstrates how to obtain, deploy and verify an X.509 certificate. While written specifically in the context of a [domain-validation single-address certificate](https://www.gandi.net/ssl/standard#single) for www.obfuscate.xyzfrom Entrust, the OpenSSL commands are valid for other vendors (e.g.: Gandi), for other certificate types (e.g.: wildcard), and for other validation modes (e.g.: organization-validation or extended-validation).

When following this guide, replace www.obfuscate.xyzwith the common name for your website. Similarly, if purchasing a certificate from another vendor, replace the URLs for the bundle of intermediate CA certificates and the root certificates, as necessary.

When the single-domain certificate signing request is for «domain», Entrust adds www.«domain» as a subject alternative name to the certificate. It is recommended to always set the common name to «domain» even if the website is intended to be hosted at www.«domain» (despite this guide using www.obfuscate.xyzthroughout this guide).

When the wildcard certificate signing request is for \*.«domain», Entrust adds «domain» as a subject alternative name to the certificate.

**Requirements**

To generate the required key, config, and signing request, you will need access to a system with the openssl command line tool installed on it. **This does not need to be the server or system on which the certificate will reside.** In fact, in most cases, it's best to generate the required files on a controlled, private system. The openssl command line tool is provided with OpenSSL or LibreSSL, which are installed by default on macOS, nearly all Linux/Unix/BSD systems, and available for download for Windows.

This guide focuses on openssl as it is consistent and platform agnostic. There are any number of other tools which you can use to generate keys and certificate request files for use with the UBC centralized certificate management service.

| **Step** | **Major Activity** | **References, Forms and Details** |
| --- | --- | --- |
| 1 | Read the document 335-F01 to understand the basic concepts of webserver security | 335-F01 |
| 2 | **Replace all references throughout this process to** [**www.obfuscate.xyz**](http://www.obfuscate.xyz/) **with the actual website name**.  Save the modified document as a working copy with a different name. | Other than replacing [www.obfuscate.xyz](http://www.obfuscate.xyz/), do not reformat any strings.  This will facilitate a copy-paste from the working copy process to the server commands. |
| 3 | Generate the configuration file (www.obfuscate.xyz.cfg )  [ req ]  distinguished\_name = distinguished\_name  default\_md = sha256  prompt = no    [ distinguished\_name ]  C = CA  ST = British Columbia  L = Vancouver  O = The University of British Columbia  OU = CISO Office  CN = www.obfuscate.xyz | The cost is absorbed by UBC, and thus is effectively free to the user.  Provide the configuration file exactly as listed, subject to the replacement string.  Typical file is 335-F20 |
| 4 | On a convenient server, generate the **private key**. Using the following command, generate www.obfuscate.xyz.key  containing a P-256 ECDSA private key:  openssl ecparam -out www.obfuscate.xyz.key -name prime256v1 -genkey | Do not reformat the command string or it may not work.  **Public-key cryptography**, or **asymmetric cryptography**, is a cryptographic system that uses pairs of keys: ***public keys***, which may be disseminated widely, and ***private keys***, which are known only to the owner.  Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security  ECDSA keys and certificates are recommended by Cybersecurity |
| 5 | On a convenient server, generate the certificate signing request (CSR)  openssl req -config www.obfuscate.xyz.cfg -new -key www.obfuscate.xyz.key -out [www.obfuscate.xyz.csr](http://www.obfuscate.xyz.csr/) | Other than replacing www.obfuscate.xyz, do not reformat the command string or it may not work.  A CSR is necessary for issuing a TLS certificate. The CSR request is normally made by the server administrator on the server where is the domain hosted. |
| 6 | Verify the certificate integrity (private key and csr)  openssl pkey -pubout -**in** www.obfuscate.xyz.key | openssl dgst -sha1 -binary | xxd -p  openssl req -noout -pubkey -**in** www.obfuscate.xyz.csr | openssl dgst -sha1 -binary | xxd -p | Other than replacing www.obfuscate.xyz, do not reformat the string.  For each test, the response should be the same each time. For example:  021d551fb04d52b18464ddb26ba95c79d94a3c3d |
| 7 | Send the \*.cfg and \*.csr to the UBC CISO (Chief Information Security Officer). | The CISO administrator sends the CSR (certificate signing request) to the Certification Authority, which contains the public key, the name and usually the location. This CSR is then signed by the CA to prove ownership of the associated **private key**. |
| 8 | UBC Cybersecurity will provide the root.crt, intermediate.crt and end-entity certificate issued by CA (UBC: Entrust), or CA will provide download instructions.  CA (Apache server, pem format)  ServerCertificate.crt and ChainBundle2.crt  or  CA (other server, non-pem)  Root.crt, Intermediate.crt and  ServerCertificate.crt | There are two types of [certificate authorities (CAs)](https://support.dnsimple.com/articles/what-is-certificate-authority): **root CAs** and **intermediate CAs**. For an SSL certificate to be trusted, that certificate must have been **issued by a CA that’s included in the trusted store of the device that’s connecting.**  PEM certificates may have the following file extensions: \*.pem, \*.crt, \*.cer.  The \*.pem is a base64 encoded certificate placed between the headers  -----BEGIN CERTIFICATE-----  and  -----END CERTIFICATE-----  It is the most widespread certificate format, which is mostly used by Linux-based servers, like Apache, Nginx, and by the majority of webhosting control panels (cPanel, Plesk, DirectAdmin, WebMin, etc.). |
| 9 | **Rename the certificates**  mv ServerCertificate.crt [www.obfuscate.xyz.pem](http://www.obfuscate.xyz.pem/)  mv ChainBundle2.crt [www.obfuscate.xyz-cacerts.pem](http://www.obfuscate.xyz-cacerts.pem/) | Renaming the certificates is optional, but good practice as it consolidates the naming. |
| 10 | If not already provided by the CA with ChainedBundle2.crt, obtain the bundle of intermediate CA certificates  wget syntax  wget -q -O - https://entrust.com/root-certificates/entrust\_l1f.cer | tr -d '\r' > www.obfuscate.xyz-cacerts.pem | Alternatively use curl syntax  curl -sL https://entrust.com/root-certificates/entrust\_l1f.cer | tr -d '\r' > www.obfuscate.xyz-cacerts.pem |
| 11 | Verify the received server pem certificate integrity (public key)  openssl x509 -noout -pubkey -**in** www.obfuscate.xyz.pem | openssl dgst -sha1 -binary | xxd -p | For this test, the response should be the same as the two previous tests in step 6. For example:  021d551fb04d52b18464ddb26ba95c79d94a3c3d |
| 12 | Verify the certificate's trust chain, test 1  openssl x509 -noout -subject -issuer -fingerprint -in www.obfuscate.xyz.pem | Typical result of test 1  subject= /C=CA/ST=British Columbia/L=Vancouver/O=The University of British Columbia/CN=www.obfuscate.xyz  issuer= /C=US/O=Entrust, Inc./OU=See www.entrust.net/legal-terms/OU=(c) 2012 Entrust, Inc. - for authorized use only/CN=Entrust Certification Authority - L1K  SHA1 Fingerprint=  C2:1A:1D:78:9C:04:94:AF:3B:CE:FC:FB:89:B8:4B:72:12:CC:0E:62 |
| 13 | Verify the certificate's trust chain, test 2  cat www.obfuscate.xyz-cacerts.pem | gawk 'BEGIN {RS="-----END CERTIFICATE-----\n";ORS=RS} NR==1 {print}' | openssl x509 -noout -subject -issuer –fingerprint | If verifying \*‑cacerts.pem   * cannot use ‘fingerprint’ * two strings end in L1F and EC1   If verifying ChainedBundle2.crt   * cannot use ‘fingerprint’ * two strings end in EC1 and EC1   Sample result of test 2  subject= /C=US/O=Entrust, Inc./OU=See www.entrust.net/legal-terms/OU=(c) 2012 Entrust, Inc. - for authorized use only/CN=Entrust Certification Authority - L1K  issuer= /C=US/O=Entrust, Inc./OU=See www.entrust.net/legal-terms/OU=(c) 2009 Entrust, Inc. - for authorized use only/CN=Entrust Root Certification Authority - G2 |
| 14 | Generate the chained certificate (server, intermediate certificates and public key)  cat www.obfuscate.xyz.pem <(echo) www.obfuscate.xyz-cacerts.pem > www.obfuscate.xyz-chained.pem | If the certificate wasn’t issued by a trusted CA, the connecting device (eg. a web browser) checks to see if the certificate of the issuing CA was issued by a trusted CA. It continues checking until either a trusted CA is found (at which point a trusted, secure connection will be established), or no trusted CA can be found (at which point the device will usually display an error). |
| 15 | Generate the PKCS12 archive (for Microsoft IIS or Apache Tomcat or Jetty)  Create a random password:  openssl rand -base64 36 > www.obfuscate.xyz-pfxpass.txt  And create the archive:  openssl pkcs12 -export -inkey www.obfuscate.xyz.key -in www.obfuscate.xyz-chained.pem -out www.obfuscate.xyz-chained.pfx -passout file:www.obfuscate.xyz-pfxpass.txt -name www.obfuscate.xyz | A PKCS12 (Public-Key Cryptography Standards) defines an archive file format for storing server certificates, intermediate certificate if any, and private key into a single encryptable file  PKCS#12/PFX is a file in binary format that contains the server certificate, optional chain certificates, the corresponding private key and is protected by a password.  The file extensions are: \*.p12 and \*.pfx.  Usually, PFX certificates are used on Windows machines, and are essential for transferring the certificate from one Windows server to another. |
| 16 | As root, move all ssl certificates for Apache to secure location  mv {cert\_location}/www.obfuscate.xyz\* *etc*pki/tls/certs | This step moves all the certificate and passphrase files to the secure area. |
| 17 | Move to secure area and configure files in final locations  cd etc/pki/tls/certs  chown -R root:root [www.obfuscate.xyz](http://www.obfuscate.xyz/)\*  chmod 644 [www.obfuscate.xyz](http://www.obfuscate.xyz/)\*  mv \*pfx\* \*key\*../private/  mv \*jksp\* ../private/  cd ../private/  chmod 440 [www.obfuscate.xyz](http://www.obfuscate.xyz/)\*  chmod 440 www.obfuscate.xyz.key  chown apache:apache www.obfuscate.xyz.key | At the conclusion:  certs folder contents  \*.cfg, \*.csr - not strictly required  \*.pem server certificate  \*-cacerts.pem trust chain certificates  \*-chained.pem server + trust chain certificates  \*.p7b chained certificates in p7b format (not needed)  As well, in the certs folder there are two important prior links to ca-trust folder:  ca-bundle.crt->/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.crt  ca-bundle.trust.crt->/etc/pki/ca­trust/extracted/openssl/ca-bundle.trust.crt  private folder contents  \*.key private key  \*-chained.pfx certificates in pkcs12 format  \*-pfxpass.txt passphrase for the pfx file  Following two passphrases in private folder only required if JKS format keystore:  \*-keypass.txt passphrase for the keystore  \*-jkspass.txt passphrase for the jks format keystore |
| 18 | Verify path  For Debian and Debian-based distributions (e.g.: Ubuntu) and Redhat7+  openssl verify -CApath /etc/ssl/certs -untrusted www.obfuscate.xyz-cacerts.pem [www.obfuscate.xyz.pem](http://www.obfuscate.xyz.pem/)  For RedHat6- and Redhat-based distributions (e.g.: CentOS6-):  openssl verify -CAfile /etc/ssl/certs/ca-bundle.trust.crt -untrusted www.obfuscate.xyz-cacerts.pem www.obfuscate.xyz.pem | Result of test  www.obfuscate.xyz.pem: OK |
| 19 | Verify the certificate's deployment with OpenSSL  For Debian and Debian-based distributions (e.g.: Ubuntu) and Redhat7+  echo "GET /" | openssl s\_client -CApath /etc/ssl/certs -connect www.obfuscate.xyz:443 -servername [www.obfuscate.xyz](http://www.obfuscate.xyz/)  For RedHat6- and Redhat-based distributions (e.g.: CentOS6-):  echo "GET /" | openssl s\_client -CAfile /etc/ssl/certs/ca-bundle.trust.crt -connect www.obfuscate.xyz:443 -servername www.obfuscate.xyz | Result truncated:      ...      Verify return code: 0 (ok)  ---  DONE |
| 20 | As superuser point at file locations in the Virtual Host file of the conf override folder conf.d:  cd /etc/httpd/conf.d  vim 20-ssl-vhost.conf  Set the following five parameters and save:  ServerName www.obfuscate.xyz  SSLCertificateFile /etc/pki/tls/certs/www.obfuscate.xyz.pem  SSLCertificateKeyFile /etc/pki/tls/private/www.obfuscate.xyz.key  SSLCertificateChainFile /etc/pki/tls/certs/www.obfuscate.xyz-cacerts.pem  SSLCACertificateFile /etc/pki/tls/certs/www.obfuscate.xyz-chained.pem | Folder conf.d is chosen because it is easier to keep track of changes and you do not need to touch httpd.conf, which might change during an update.  Typical 20-ssl-vhost.conf is at 335-F30 |
| 21 | Check Apache configuration  apachectl configtest |  |
| 22 | Update Openssl and reboot:  dnf update  reboot | Update can take a few minutes  Reboot can take a minute or so |
| 23 | Logon and configure the apache daemon:  systemctl enable httpd  systemctl start httpd  systemctl status httpd | Should see message  "Running, listening on: port 443, port 80"  Typical Apache and Jetty status reports at 335-F50. |
| 24 | Copy the \*.pfx file into the etc location as keystore  cp /etc/pku/tls/private/www.obfuscate.xyz-chained.pfx $JETTY\_BASE/etc/keystore | Keystore is protected by pfxpass passphrase:  /etc/pku/tls/private/www.obfuscate.xyz-pfxpass.txt |
| 25 | chown jetty:jetty $JETTY\_BASE/etc/keystore | Ensure ownership by jetty |
| 26 | Confirm keystore is correct:  openssl pkcs12  -in $JETTY\_BASE/etc/keystore \  -out result.txt /  -nodes |  |
| 27 | Place the following environment variables in /etc/profile.d/geo\_stuff.sh:  GEOWEBCACHE\_CACHE\_DIR=/data/gwc  GEOSERVER\_DATA\_DIR=/data/gs\_dir  Geonetwork\_dir=/data/gn\_dir | These variables are for all users  File name geo\_stuff.sh is arbitrary  Locations for geowebcache, geoserver and geonetwork are samples |
| 28 | Place the following environment variables in ~/.bashrc:  JETTY\_HOME={home dir}  JETTY\_BASE={mybase}  PGHOST=localhost  PGUSER=postgres  PGPORT= 5432 | These variables are unique for each user (basically hides the locations slightly) |
| 29 | Confirm Jetty environment  cd $JETTY\_BASE  java –jar $JETTY\_HOME/start.jar \  --list-config | Typical result in 335-F50 |
| 30 | Open port 8080 as superuser:  firewall-cmd –state  firewall-cmd –get-active-zones  firewall-cmd –zone-=public \  --add-port=8080/tcp \  --permanent  firewall-cmd –reload  firewall-cmd –list-all  firewall-cmd –list-port | If port not already open  Use sudo for superuser  Alternative to –permanent is  firewall-cmd –runtime-to-permanent  Expected response to –list-port is 8080/tcp  If need to remove a port (or service):  firewall-cmd –zone=public \  --remove-port=8080/tcp (or -service=http)  firewall-cmd --reload |
| 31 | Check listen ports:  netstat -ntlp | grep LISTEN |  |
| 32 | Start jetty as a service (as superuser)  systemctl start jetty  systemctl status jetty | For a typical response refer to  330-F60 Jetty systemctl status.txt |