**Loging-log file location= %BASE\_DIR%/%SERVER\_URI%/log/**

[**http://blogs.forgerock.org/petermajor/**](http://blogs.forgerock.org/petermajor/)

|  |  |
| --- | --- |
|  | |
| [Zurück](javascript:history.back()) | How to create a self-signed SSL Certificate ... ...  which can be used for testing purposes or internal usage |
| Overview  The following is an extremely simplified view of how SSL is implemented and what part the certificate plays in the entire process.  Normal web traffic is sent unencrypted over the Internet. That is, anyone with access to the right tools can snoop all of that traffic. Obviously, this can lead to problems, especially where security and privacy is necessary, such as in credit card data and bank transactions. The Secure Socket Layer is used to encrypt the data stream between the web server and the web client (the browser).  SSL makes use of what is known as **asymmetric cryptography**, commonly referred to as **public key cryptography (PKI)**. With public key cryptography, two keys are created, one public, one private. Anything encrypted with either key can only be decrypted with its corresponding key. Thus if a message or data stream were encrypted with the server's private key, it can be decrypted only using its corresponding public key, ensuring that the data only could have come from the server.  If SSL utilizes public key cryptography to encrypt the data stream traveling over the Internet, why is a certificate necessary? The technical answer to that question is that a certificate is not really necessary - the data is secure and cannot easily be decrypted by a third party. However, certificates do serve a crucial role in the communication process. The certificate, signed by a trusted Certificate Authority (CA), ensures that the certificate holder is really who he claims to be. Without a trusted signed certificate, your data may be encrypted, however, the party you are communicating with may not be whom you think. Without certificates, impersonation attacks would be much more common.  Step 1: Generate a Private Key  The **openssl** toolkit is used to generate an **RSA Private Key** and **CSR (Certificate Signing Request)**. It can also be used to generate self-signed certificates which can be used for testing purposes or internal usage.  The first step is to create your RSA Private Key. This key is a 1024 bit RSA key which is encrypted using Triple-DES and stored in a PEM format so that it is readable as ASCII text.  **openssl genrsa -des3 -out server.key 1024** Generating RSA private key, 1024 bit long modulus .........................................................++++++ ........++++++ e is 65537 (0x10001) Enter PEM pass phrase: Verifying password - Enter PEM pass phrase:  Step 2: Generate a CSR (Certificate Signing Request)  Once the private key is generated a Certificate Signing Request can be generated. The CSR is then used in one of two ways. Ideally, the CSR will be sent to a Certificate Authority, such as Thawte or Verisign who will verify the identity of the requestor and issue a signed certificate. **The second option is to self-sign the CSR, which will be demonstrated in the next section**.  During the generation of the CSR, you will be prompted for several pieces of information. These are the X.509 attributes of the certificate. One of the prompts will be for "Common Name (e.g., YOUR name)". It is important that this field be filled in with the fully qualified domain name of the server to be protected by SSL. If the website to be protected will be https://public.akadia.com, then enter public.akadia.com at this prompt. The command to generate the CSR is as follows:  **openssl req -new -key server.key -out server.csr** Country Name (2 letter code) [GB]:**CH** State or Province Name (full name) [Berkshire]:**Bern** Locality Name (eg, city) [Newbury]:**Oberdiessbach** Organization Name (eg, company) [My Company Ltd]:**Akadia AG** Organizational Unit Name (eg, section) []:**Information Technology** Common Name (eg, your name or your server's hostname) []:**public.akadia.com** Email Address []:**martin dot zahn at akadia dot ch** Please enter the following 'extra' attributes to be sent with your certificate request A challenge password []: An optional company name []:  Step 3: Remove Passphrase from Key  One unfortunate side-effect of the pass-phrased private key is **that Apache will ask for the pass-phrase each time the web server is started**. Obviously this is not necessarily convenient as someone will not always be around to type in the pass-phrase, such as after a reboot or crash. mod\_ssl includes the ability to use an external program in place of the built-in pass-phrase dialog, however, this is not necessarily the most secure option either. **It is possible to remove the Triple-DES encryption from the key**, thereby no longer needing to type in a pass-phrase. If the private key is no longer encrypted, it is critical that this file only be readable by the root user! If your system is ever compromised and a third party obtains your unencrypted private key, the corresponding certificate will need to be revoked. With that being said, use the following command to remove the pass-phrase from the key:  **cp server.key server.key.org openssl rsa -in server.key.org -out server.key**  The newly created server.key file has no more passphrase in it.  -rw-r--r-- 1 root root 745 Jun 29 12:19 server.csr -rw-r--r-- 1 root root 891 Jun 29 13:22 server.key -rw-r--r-- 1 root root 963 Jun 29 13:22 server.key.org  Step 4: Generating a Self-Signed Certificate  At this point you will need to generate a self-signed certificate because you either don't plan on having your certificate signed by a CA, or you wish to test your new SSL implementation while the CA is signing your certificate. This temporary certificate will generate an error in the client browser to the effect that the signing certificate authority is unknown and not trusted.  To generate a temporary certificate which is good for 365 days, issue the following command:  **openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt** Signature ok subject=/C=CH/ST=Bern/L=Oberdiessbach/O=Akadia AG/OU=Information Technology/CN=public.akadia.com/Email=martin dot zahn at akadia dot ch Getting Private key  Step 5: 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.  **cp server.crt /usr/local/apache/conf/ssl.crt cp server.key /usr/local/apache/conf/ssl.key**  Step 6: Configuring SSL Enabled Virtual Hosts  **SSLEngine on SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key SetEnvIf User-Agent ".\*MSIE.\*" nokeepalive ssl-unclean-shutdown CustomLog logs/ssl\_request\_log \    "%t %h %{SSL\_PROTOCOL}x %{SSL\_CIPHER}x \"%r\" %b"**  **Step 7: Restart Apache and Test**  **/etc/init.d/httpd stop /etc/init.d/httpd stop**  **https://public.akadia.com** | |

**Create self-signed SSL certificate**

Authentication part of the SSL communication works on a private-public key based trust system. In other words you have a key that is signed by another key from a 3rd party which rest of the world (or the people who needs to verify you) trust. This signed key is called a certificate.

It is however possible for you to sign your own certificate, but this certificate wont be trusted by others but it will provide the SSL encryption just like a paid SSL certificate.

Few scenarios why you would need/use such certificate are:

* Provide encrypted communication between two parties that are already trust each other (i.e. Web application on a intranet)
* Development or demo web site that needs to simulate the production environment with SSL

**Private Key**

First step in the process is to generate a private key, which will be your unique signature to sign certificates.

**$** openssl genrsa -aes256 -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:

We have created a 4096-bit long key with AES256 encryption and it has asked us for a password to protect the key with. Generated key will be saved to the file named server.key, You need to keep this key in a safe place as it will be needed in the future to extend the validity of the SSL certificate.

Also if someone get hold of this key, they will be able to generate duplicate certificates for your server.

**Certificate Sign Request**

Next step is to generate a Certificate Signing Request (CSR). CSR contains a signature of your server along with some data that are displayed to the public when some one look at this certificate (i.e Company name, address, etc.)

**$** openssl req -new -key server.key -out demo.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]:LK

State or Province Name (full name) [Some-State]:

Locality Name (eg, city) []:Pannipitiya

Organization Name (eg, company) [Internet Widgits Pty Ltd]:

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:sudaraka.org

Email Address []:info@sudaraka.org

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:

You will first need to enter the password for your key file which you created in the earlier step. Every thing after that are optional details except for the “Common Name” which is the exact domain name you are creating this certificate for. \* character can be used as a wild-card.

In a normal procedure where you get your SSL certificate signed by a 3rd party authority, you will be sending this CSR to the relevant authority to be signed. This is where the self-sign process forks away from that procedure.

**Certificate Signing**

You can use the same key we created above to sign the CSR, or you can create a new key by following the same instructions. Let's assume you created a second key called sign.key.

**$** openssl x509 -req -days 365 -in demo.csr -signkey sign.key -out demo.crt

Signature ok

subject=/C=LK/ST=Some-State/L=Pannipitiya/O=Internet Widgits Pty Ltd/CN=sudaraka.org/emailAddress=info@sudaraka.org

Getting Private key

Enter pass phrase for sign.key:

You now have everything that is needed to add the SSL certificate to Apache web server (demo.crt and sever.key)

**Usage**

In the Apache configuration files locate the virtual host you want to add this SSL certificate and add the following.

1. SSLEngine on
2. SSLCertificateFile "/path/to/demo.crt"
3. SSLCertificateKeyFile "/path/to/sign.key"

One possible issue here is since our sign.key is password protected, Apache will require that password every time it starts and it has to be entered interactively. This is not possible because in most systems Apache run as a daemon process. Solution is to remove the password from the sign.key that we use for Apache like below.

**$** openssl rsa -in sign.key -out sign.key

Enter pass phrase for sign.key:

writing RSA key

Now restart Apache and you should be good to go.

<https://login.example.com/OpenAM-11.0.0/federation/FSAuthDomainsEdit>

-XX:+CMSClassUnloadingEnabled

-XX:+CMSPermGenSweepingEnabled

-XX:+UseConcMarkSweepGC

-XX:MaxPermSize=1024M

http://login.example.com/OpenAM-11.0.0/config/defaultSummary.htm

|  |  |
| --- | --- |
| **Default User [amAdmin]** | |
| *\**Password |  |
| *\**Confirm Password |  |

|  |
| --- |
| **Default Policy Agent [UrlAccessAgent]** |