By punk0mi
Published: 2007-05-16 17:14

Secure Websites Using SSL And Certificates

This how-to will guide you through the entire process of setting up a secure website using SSL and digital certificates. This guide assumes that you have already a fully functional (and configured) server running Apache, BIND, and OpenSSL. Just as a side note, this guide was written based on a Fedora Core 6 distribution, but should be the same for most other distros out there.

Introduction

Today it is possible to create a secure website with relative ease by requiring a client to present a digitally signed certificate. A digitally signed certificate is simply a piece of information that contains data about the subject, public key, dates of validity, identification of the Certificate Authority (CA), and the digital signature. There are typically two ways to go about creating a secure website. First is by the use of a self-signed certificate. The second way is by using a Trusted Certificate signed by a CA. The choice is up to you, and this tutorial will show you how to do both. Go ahead and *su* into root and let's begin!

Step 1: Preconfigure OpenSSL

We can preconfigure OpenSSL to fill in the fields of the certificate by modifying its configuration file. Doing this will save us a few steps during the certificate creation process. First, navigate to the SSL directory and do a listing of files

```
# cd /etc/pki/tls
# ls -al
```

In the listing you should see the file *openssl.cnf*. Go ahead and open up this file in your favorite editor and find the beginning of the *[req_distinguished_name]* section. Modify the *countryName_default, stateOrProvinceName_default, 0.organizationName_default, 1.organizationName_default* values with the values that suit your needs. The section of the file you are looking for looks like this

```
[ req_distinguished_name ]
```

```
countryName                 = Country Name (2 letter code)
countryName_default         = US
countryName_min             = 2
countryName_max             = 2
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = Berkshire
localityName                = Locality Name (eg, city)
localityName_default        = Berkshire Hills
0.organizationName          = Organization Name (eg, company)
0.organizationName_default  = My Secure Website Company
# we can do this but it is not needed normally :-)
#1.organizationName         = Second Organization Name (eg, company)
#1.organizationName_default = MSWC
organizationalUnitName      = Organizational Unit Name (eg, section)
#organizationalUnitName_default = Your IT Department
commonName                  = Common Name (eg, your name or your server\'s hostname)
commonName_max              = 64
emailAddress                = Email Address
emailAddress_max            = 64
```

Once you have made the changes, save and quit the editor.

Step 2: Creating The Server CA

Now we must create the server's CA. You should still be in the `/etc/pki/tls` directory, if you are not sure just run a quick `pwd` to find out. We now need to make sure that we have no other certificates in the system:

```
# rm -rf ../../CA
```

Once that is done, we will use the shell scripts OpenSSL comes with to create a new CA:

```
# misc/CA -newca
```

A dialog will appear saying:

```
CA certificate filename (or enter to create)
```

Go ahead and press enter. At this point the program will begin generating the CA. During this process you will be asked to enter a PEM pass phrase, create one and write it down! Once you have entered the PEM and verified it you will be asked some questions about the Distinguished Name (DN). A lot of the questions will have the correct values in them by default since we edited the *openssl.cnf* file. So press enter until it asks for the *Common Name*. When asked for the common name you should put down the server's hostname. It should look like this:

```
#misc/CA -newca
   CA certificate filename (or enter to create)
   Making CA certificate ...
   Generating a 1024 bit RSA private key
   .............++++++
   ......................++++++
   writing new private key to '../../CA/private/./cakey.pem'
   Enter PEM pass phrase:    XXXXXXXXX
   Verifying - Enter PEM pass phrase:    XXXXXXXXX
   -----
  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) [US]:
State or Province Name (full name) [Berkshire]:
Locality Name (eg, city) [Berkshire Hills]:
```

```
Organization Name (eg, company) [My Secure Website Company]:
Organizational Unit Name (eg, section) [Your IT Department]:
Common Name (eg, your name or your server's hostname) []:hostname
Email Address []:admin-email@yourdomain.com
```

The CA for the server has been created and placed in the */etc/CA/private/cakey.pem* file.

Step 3: Creating A Server Certificate Request

Now that we have the CA established, we need to make a request to create the certificate. Once again, using the shell scripts included with OpenSSL we want to run this command:

```
# misc/CA -newreq
```

Once the command has been entered, the program will being generating the request. When asked for the PEM, go ahead and enter it and press enter. Once again you will be asked the same questions about the Distinguished Name (DN). Everything should be the same, except this time when asked for the common name you should enter the complete address of the site you wish to secure (not just the hostname as this will cause internet browsers to see a conflict!). Once you have entered the Common Name you will be asked to create a Challenge Password, create one and write it down. This process should look like this:

```
#misc/CA -newreq
   CA certificate filename (or enter to create)
   Making CA certificate ...
   Generating a 1024 bit RSA private key
   .............++++++
   ........................++++++
   writing new private key to 'newkey.pem'
   Enter PEM pass phrase:    XXXXXXXXX
   Verifying - Enter PEM pass phrase:    XXXXXXXXX
   -----
  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) [US]:
State or Province Name (full name) [Berkshire]:
Locality Name (eg, city) [Berkshire Hills]:
Organization Name (eg, company) [My Secure Website Company]:
Organizational Unit Name (eg, section) [Your IT Department]:
Common Name (eg, your name or your server's hostname) []:hostname.yourdomain.com
Email Address []:admin-email@yourdomain.com
   A challenge password []:
   An optional company name []:
   Request is in newreq.pem, private key is in newkey.pem
```

The certificate request has now been created and placed in the file */etc/pki/tls/newreq.pem* and the private key is in */etc/pki/tls/newkey.pem*.

Step 4: Signing The Certificate

Before you move on to the next step you need to consider which of the two routes you are going to take - the self-signed certificate or trusted third-party signed certificate. The choice is entirely up to you. If your website is not for commercial use, you can probably get away with using a self-signed certificate, however, most web browsers will not recognize it as a "trusted website" right away since it does not have the signature of a trusted CA. Alternatively to both the self-signed or purchased methods, I recommend looking into using **CACert.org**. CACert.org is a completely public, community driven, and FREE CA that is growing.

If you want to create a Self-Signed certificate go to Step 4A.

If you want to create a Trusted certificate thru a third party go to Step 4B.

Step 4A - The Self Signed Certificate

This is a fairly simple process. Using the shell scripts we can sign our own certificate with ease by running the following command

```
# misc/CA -sign
```

Once executed the program loads the configuration file from the *openssl.cnf* file. Once it opens the CA key information you will be prompted for the PEM pass phrase, go ahead and enter it. Once the pass phrase is entered the program then verifies the integrity and validity of the data. Once it is done checking, it will ask if you want to sign the certificate, enter "y" and continue. The whole result should look something like this:

```
# misc/CA -sign
   Using configuration from /etc/pki/tls/openssl.cnf
   Enter pass phrase for ../../CA/private/cakey.pem: XXXXXXXXX
   Check that the request matches the signature
   Signature ok
   Certificate Details:
   Serial Number: 1 (0x1)
   Validity
   Not Before: Dec 22 18:52:38 2006 GMT
   Not After : Dec 22 18:52:38 2007 GMT
   Subject:
   countryName               = US
   stateOrProvinceName       = Berkshire
   localityName              = Berkshire Hills
   organizationName          = My Secure Website Company
   organizationalUnitName    = Your IT Department
   commonName                = hostname.yourdomain.com
   emailAddress              = admin-email@yourdomain.com
   X509v3 extensions:
   X509v3 Basic Constraints:
   CA:FALSE
   Netscape Comment:
   OpenSSL Generated Certificate
   X509v3 Subject Key Identifier:
   CA:D9:9B:01:D2:9C:15:39:96:62:53:29:D6:6E:D8:B8:62:9D:A0:BD
   X509v3 Authority Key Identifier:
```

```
keyid:DB:B6:B7:15:40:C4:7B:14:AE:F6:CB:A9:DF:44:C3:1E:39:AE:E0:DD

Certificate is to be certified until Dec 22 18:52:38 2007 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 1 (0x1)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=Berkshire, O=My Secure Website Company, OU=Your IT Department,
CN=hostname/emailAddress=admin-email@yourdomain.com
Validity
Not Before: Dec 22 18:52:38 2006 GMT
Not After : Dec 22 18:52:38 2007 GMT
Subject: C=US, ST=Berkshire, L=Berkshire Hills, O=My Secure Website Company,
OU=Your IT Department, CN=hostname.yourdomain.com/emailAddress=admin-email@yourdomain.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:cc:91:db:ea:95:c6:d3:03:75:cd:74:b5:58:28:
b7:df:e5:33:4b:82:53:90:b0:98:5f:14:0b:d1:1a:
44:e4:41:0b:e8:59:f6:f9:d1:26:6a:d9:25:a5:ac:
8e:9d:f0:cd:65:f2:3a:13:b7:37:e2:82:3f:02:e4:
fd:f6:0b:eb:4d:27:97:2b:ab:74:07:8d:fa:0c:4b:
27:ea:c8:78:f6:1e:60:b9:fc:5f:30:0f:8f:02:4b:
d4:d1:b4:bc:a1:bb:d6:e9:dd:78:2e:76:28:21:b5:
5b:76:88:f7:cd:d3:40:26:07:33:2e:95:71:09:e1:
b4:5f:b2:95:99:fd:30:17:49
Exponent: 65537 (0x10001)
```

```
    X509v3 extensions:
    X509v3 Basic Constraints:
    CA:FALSE
    Netscape Comment:
    OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
    CA:D9:9B:01:D2:9C:15:39:96:62:53:29:D6:6E:D8:B8:62:9D:A0:BD
    X509v3 Authority Key Identifier:
    keyid:DB:B6:B7:15:40:C4:7B:14:AE:F6:CB:A9:DF:44:C3:1E:39:AE:E0:DD
    Signature Algorithm: sha1WithRSAEncryption
    69:a4:15:98:77:7e:fb:38:88:c3:a0:2e:d8:2b:24:51:50:2c:
    cd:dc:a7:56:e0:f5:f3:01:73:47:b2:67:69:4b:94:28:e9:86:
    51:fd:e0:2f:a0:30:8a:0d:0c:10:3d:5c:d5:4a:53:df:ea:0c:
    38:49:61:1b:92:61:fe:7f:96:bb:65:f4:f9:14:40:52:7d:aa:
    8c:0d:07:e1:06:aa:fb:bf:72:6f:7c:d1:68:6e:78:d6:30:7c:
    6b:f0:b9:ce:57:5b:f9:d2:cc:df:2c:42:f3:e0:42:cf:0b:5e:
    57:ff:72:ac:4a:2b:b9:ac:df:b7:23:80:ea:e4:37:01:b3:13:
    9f:73

-----BEGIN CERTIFICATE-----
MIIDIjCCAougAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgDELMAkGA1UEBhMCVVMx
ETAPBgNVBAgTCENvbG9yYWRvMQ0wCwYDVQQKEwRVQ0NTMRkwFwYDVQQLExBDb21w
dXRlciBTY2llbmNlMQ4wDAYDVQQDEwVDQWZjNjEkMCIGCSqGSIb3DQEJARYVY2FA
ZmM2LmNzbmV0LnVjY3MuZWR1MB4XDTA2MTIyMjE4NTIzOFoXDTA3MTIyMjE4NTIz
OFowga8xCzAJBgNVBAYTAlVTMREwDwYDVQQIEwhDb2xvcmFkbzEZMBcGA1UEBxMQ
Q29sb3JhZG8gU3ByaW5nczENMAsGA1UEChMEVUNDUzEZMBcGA1UECxMQQ29tcHV0
ZXIgU2NpZW5jZTEbMBkGA1UEAxMSZmM2LmNzbmV0LnVjY3MuZWR1MSswKQYJKoZI
hvcNAQkBFhx3ZWJtYXN0ZXJAZmM2LmNzbmV0LnVjY3MuZWR1MIGfMA0GCSqGSIb3
DQEBAQUAA4GNADCBiQKBgQDMkdvqlcbTA3XNdLVYKLff5TNLglOQsJhfFAvRGkTk
QQvoWfb50SZq2SWlrI6d8M1l8joTtzfigj8C5P32C+tNJ5crq3QHjfoMSyfqyHj2
HmC5/F8wD48CS9TRtLyhu9bp3XgudightVt2iPfN00AmBzMulXEJ4bRfspWZ/TAX
SQIDAQABo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdl
bmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUytmbAdKcFTmWYlMp1m7YuGKd
```

```
oL0wHwYDVR0jBBgwFoAU27a3FUDEexSu9sup30TDHjmu4N0wDQYJKoZIhvcNAQEF
BQADgYEAaaQVmHd++ziIw6Au2CskUVAszdynVuD18wFzR7JnaUuUKOmGUf3gL6Aw
ig0MED1c1UpT3+oMOElhG5Jh/n+Wu2X0+RRAUn2qjA0H4Qaq+79yb3zRaG541jB8
a/C5zldb+dLM3yxC8+BCzwteV/9yrEoruazftyOA6uQ3AbMTn3M=
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
```

The certificate has now been self-signed and placed in the `/etc/pki/tls` directory. Both the private key and the certificate need to be renamed to better identify what server they represent so do the following:

```
# mv newcert.pem hostnameCert.prm
# mv newkey.pem hostnameKey.pem
```

That is it! Continue on to step 5 to configure Apache to use the newly created information.

Step 4B - The Trusted Certificate

To get a server certificate created by some third-party company like Thawte, VeriSign, GeoTrust, or CACert.org, look for the certification request file that was created in step 3 (This file can be found in `/etc/pki/tls/newreq.pem`). Since each CA operates differently, you will have to research which way they want you to submit the data (also called the CSR by some websites)held within this file; however, most places have a simple on-line copy-and-paste procedure, or a file upload system. Once you have submitted the data held in the `newreq.pem` file you will be given back another batch of code similar to what you submitted. This snippet of code is your new signed/trusted certificate.

Once you have that code you will need to save it back to the `/etc/pki/tls` directory with a filename which makes it easily identifiable such as `hostnameCert.pem`

Once you have completed this step you are ready to move on to Step 5.

Step 5: Configuring Apache To Use Your Certificates

We must now configure the Apache server to utilize SSL. The first thing we will want to do is copy our newly made certificates to the proper location. Do the following:

```
# cd /etc/httpd/conf
# ls -al
```

In the listing, look for directories called *ssl.key* and *ssl.crt*. If they do not exist go ahead and make them:

```
# mkdir ssl.key ssl.crt
```

Once you have found or created the directories, it is time to copy our certificates over:

```
# cp /etc/pki/tls/hostnameCert.pem ssl.crt/server.crt
# cp /etc/pki/tls/hostnameKey.pem ssl.key/server.key
```

Once you have copied over the files, it is time to configure the Apache server to use them. Do this:

```
# cd /etc/httpd/conf.d
# ls -al
```

You should see a file called *ssl.conf*. Go ahead and open this file with your favorite editor. There are two thing that need to be changed in this file. They are around lines 112 and 119, it looks like this:

```
#   Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate.  If
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/httpd/conf/ssl.crt/localhost.crt



#   Server Private Key:
```

```
#  If the key is not combined with the certificate, use this
#  directive to point at the key file.  Keep in mind that if
#  you've both a RSA and a DSA private key you can configure
#  both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/localhost.key
```

Change the *SSLCertificateFile* path to the new path of */etc/httpd/conf/ssl.crt/server.crt*. Then, we need to change the *SSLCertificateKeyFile* to the new path of */etc/httpd/conf/ssl.key/server.key*. Once these changes are made save and quit the editor.

If we leave the configuration just as it is now, the SSL will work with a restart of the apache server. However, there is bit of a bump - each time we restart the httpd server it will ask us for the PEM passphrase. This is ok if you are constantly able to monitor the system, but for the sake of saving you that headache let's fix that problem. To fix this issue do the following:

```
# cd /etc/httpd/conf/ssl.key
# cp server.key server.key.orig
# openssl rsa -in server.key.orig -out server.key
```

You will be asked to enter the PEM pass phrase, so go ahead and input it and press enter. The program will finish and will have created the key file that will not require you to enter the PEM pass phrase on starting Apache. For security, we need to put security permissions on this file and restart the Apache server:

```
# chmod 700 server.key
# service httpd restart
```

If everything was done correctly, Apache will start with no problems. Now it is time to try out your new secure website. Open your browser and navigate to *https://hostname.yourdomain.com*.

If you decided to do a self-signed certificate, or one through CACert.org you may get a message about the certificate not being trusted. This is normal and you should just tell the browser to accept the certificate. Once your on the website you should be able to view your certificate and the details on it.

Congratulations - you have just set up up an SSL secure website!