## Apache with SSL building from source

Apache with ssl support should be the basic platform for providing web services...

There are several different implementations to choose from, some commercial (stronghold) and some open source (apache+ssl, apache+modssl). We've chosen to work with apache+modssl.

You can use the FreeBSD ports copy of apache, or build your own.

Much of how you install and configure Apache will depend on how the server will be used.

Will the server host lots of user websites, or just a few web-sites?

Is the machine to be a dedicated webserver ?

Is the webserver an interface to other applications?

## Apache install exercise

```
# cd /usr/local
# fetch ftp://noc.ws.afnog.org/pub/FreeBSD/ports/distfiles/httpd-2.2.2.tar.gz
# tar xvzf httpd-2.2.2.tar.gz
# cd httpd-2.2.2
# ./configure --enable-ssl --with-ssl=/usr/src/crypto/openssl/ssl/
# make
#make install
```

## Certificate management

We shall now setup a certificate for our server in order to begin serving https requests.

For this purpose we shall use the CA.sh or CA.pl scripts that come with openssl.

First we need to create our own Certificate Authority

This we do by running

1. cd /usr/src/crypto/openssl/apps
2. chmod 0755 CA.sh  
rm -rf demoCA
3. ./CA.sh -newca

You will be prompted for a pass phrase which will be used to protect your new certificate for the authority.

Once this is done,We will then generate a signing request which we will then sign with our new Certificate Authority.

./CA.sh -newreq to generate a new request for signing and  
CA.sh -sign in order to sign the request. please use afnog06 for a passphrase

## Setting up apache to use your certificate

This is done by editing  
/usr/local/apache2/conf/extra/httpd-ssl.conf and changing  
SSLCertificateFile  
SSLCertificateKeyFile  
to point to our new certificate and its key.

In normal production environments, we would normally only generate a signing requestthat would then be signed by a recognised certificate authority.

1. mkdir /usr/local/apache2/conf/ssl.key
2. cp newcert.pem /usr/local/apache2/conf/ssl.key/server.pem
3. cp newreq.pem /usr/local/apache2/conf/ssl.key/key.pem
4. vi /usr/local/apache2/conf/extra/httpd-ssl.conf

Change the lines mentioned above to:

```
SSLCertificateFile /usr/local/apache2/conf/ssl.key/server.pem
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/key.pem
```

Now we need to enable our SSL configuration:

vi /usr/local/apache2/conf/httpd.conf

Uncomment the line that looks like

```
Include conf/extra/httpd-ssl.conf
```

## Starting up and testing apache

We now need to test whether our apache configuration is working.

We start apache using:

/usr/local/apache2/bin/apachectl start

**We then check that it is running by using:**

**ps -auxw |grep httpd**

**You should see output similar to**

```
root      432  0.0  0.3  5296   744  ??  Ss   11:15AM   0:01.40
               /usr/local/apache2/bin/httpd -k start
daemon       515  0.0  0.7  5372  1716  ??  I    11:15AM   0:00.01
               /usr/local/apache2/bin/httpd -k start
daemon       515  0.0  0.7  5372  1716  ??  I    11:15AM   0:00.01
               /usr/local/apache2/bin/httpd -k start
daemon       515  0.0  0.7  5372  1716  ??  I    11:15AM   0:00.01
```

**We can also check that there is actually a listening process on both port 80**
**for normal http and 443 for ssl by running**

```
netstat -an |grep LIST
tcp4   0   0  *.110         *.*          LISTEN
tcp4   0   0  *.25          *.*          LISTEN
tcp4   0   0  *.21          *.*          LISTEN
tcp4   0   0  *.5999        *.*          LISTEN
tcp4   0   0  *.80          *.*          LISTEN
tcp4   0   0  *.443         *.*          LISTEN
tcp4   0   0  *.22          *.*          LISTEN
```

## Testing our webserver

You can now check your apache webserver using both a web browser and the openssl s_client

```
[inst@noc inst]$ openssl
OpenSSL> s_client -host localhost -port 443
```

This will output what happens when you make an ssl connection and show us details of the certificates installed. We can also do this easier by starting up X windows and then connecting to our local webserver
using https://localhost

## Removing the pass phrase from the certificate

The RSA private key inside your server.key file is stored in encrypted format for security reasons. The pass-phrase is needed to be able to read and parse this file thus everytime you want to start the apache server you would need to enter your passphrase. This is not useful if you want to remotely reboot your server or if you're likely to get power issues so once you've secured your server , you can then remove the passphrase so you can automatically start apache at boot.

1. cd /usr/local/apache2/conf/ssl.key
2. cp key.pem key.pem.orig
3. openssl rsa -in key.pem.orig -out key.pem

It is also advisable to change the permissions on the key to make it readable only by root since it is no longer encrypted. You should now be able to restart apache without having to enter a pass phrase.

## Virtual Hosts

The term virtual hosts refers to the practice of maintaining more than one web server on the same machine as differentiated by their apparent host name. For example as an ISP you may want to host 2 different web hosting clients www.example1.com and www.client2.com on the same apache server. Apache makes doing this very easy using name based virtual hosts.

## Configuring virtual hosts

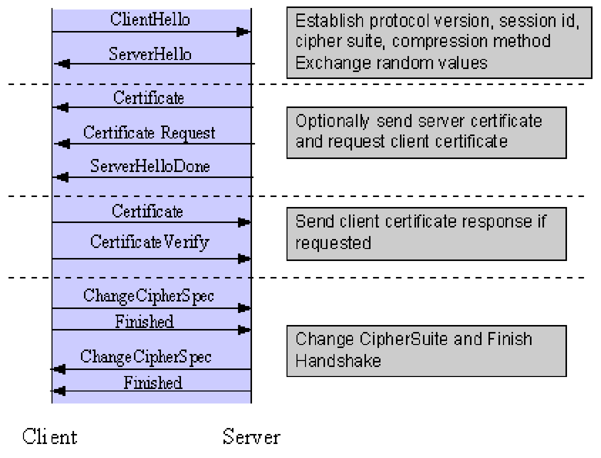
To configure our apache server to use virtual hosts we must first of all edit our httpd.conf and uncomment the line that reads:
Include conf/extra/httpd-vhosts.conf

We then proceed to edit our virtual host configuration file in
/usr/local/apache2/conf/extra/httpd-vhosts.conf

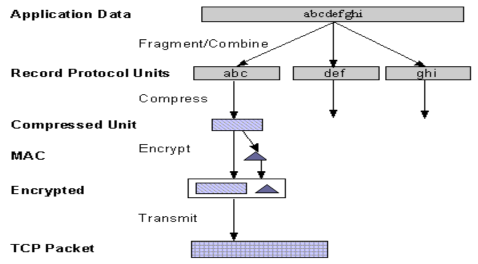
## vhost configuration

The Most important directives to consider when configuring your apache server for virtual hosting are the following:
NameVirtualHost
ServerName
ServerAlias
DocumentRoot
VirtualHost

Name based virtual hosts with apache will not work with SSL. You need to use ip based virtual host because name based hosts are implemented at the application layer while TLS is at the transport layer.
Documentation can be found at
http://httpd.apache.org/docs
http://www.modssl.org
http://httpd.apache.org/docs/vhosts/


ClientHello
ServerHello
Establish protocol version, session id, cipher suite, compression method Exchange random values
Certificate
Certificate Request
ServerHelloDone
Optionally send server certificate and request client certificate
Certificate
Certificate Verify
Send client certificate response if requested
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
Change CipherSuite and Finish Handshake
Client
Server

The SSL Record Protocol is used to transfer application and SSL Control data between the client and server, possibly fragmenting this data into smaller units, or combining multiple higher level protocol data messages into single units. It may compress, attach digest signatures, and encrypt these units before transmitting them using the underlying reliable transport protocol.



One common use of SSL is to secure Web HTTP communication between a browser and a webserver. We also commonly use it to secure other applications like pop3 and imap.

# Handshake Sequence Protocol

**The handshake sequence uses three protocols:**

*** The SSL Handshake Protocol for performing the client and server SSL session establishment.**
*** The SSL Change Cipher Spec Protocol for actually establishing agreement on the Cipher Suite for the session.**
*** The SSL Alert Protocol for conveying SSL error messages between client and server.**

**These protocols, as well as application protocol data, are encapsulated in the SSL Record Protocol. An encapsulated protocol is transferred as data by the lower layer protocol, which does not examine the data. The encapsulated protocol has no knowledge of the underlying protocol.**
**The encapsulation of SSL control protocols by the record protocol means that if an active session is renegotiated the control protocols will be transmitted securely. If there were no session before, then the Null cipher suite is used, which means there is no encryption and messages have no integrity digests until the session has been established.**