# Prerequisites

The installation of the Apache HTTPd server (HTTPd) begins by downloading several files for installation:

* Download HTTPd from [http://www.apache.org](http://apache.org)
* Download openssl from <http://www.openssl.org>
* Download mod\_proxy\_html from <http://apache.webthing.com/mod_proxy_html/>

All of these file will need to place in the /data/bits directory and then unzipped.

# Setup OpenSSL

Change into the directory that contains the openssl bits. Make sure you are running as “root”. From that directory run the following commands:

* $ ./config --prefix=/usr/local --openssldir=/usr/local/openssl
* $ make
* $ make test
* $ make install {Run as root or permissions are an issue}

The prefix points to the /usr/local directory and the openssldir points to where you want the bits installed. If you leave openssldir off, you will get it all install in /usr/local, and that will make thing confusing. Make sure you provide the openssldir flag.

The “make” builds the openssl, the “make test” ensures it is good, and “make install” creates the service in the openssldir directory.

Now openssl is installed and part of the OS path.

# Setup of HTTPs

Once the bits have been downloaded to the server from Apache.org, change your user ID to “proxy”. Unzip the downloaded file and change into the created directory for the bits and run the following commands:

* $ ./configure --prefix=/data/httpd-2.2.21 --enable-proxy --enable-rewrite --enable-ssl –enable-headers
* $ make
* $ make test
* $ make install

The “configure” command build the specifics for HTTPd: prefix sets the directory where HTTPd will be installed, the enable flags tell the installer to include the listed modules. The first make builds the source files to be made and installed, the make test checks for errors. The make install builds the server into the prefix directory, includeing all of the added modules. If you encounter errors you can correct them and then run “make clean” to build your server from scratch.

Now the server is installed and ready to run. The server can be started using the following command: [serverRoot]/bin/apachectl start.

[serverRoot]/bin/apachectl command that you should know:

* stop
* start
* restart
* graceful (restarts gracefully, not dumping any current connections)
* graceful-stop (stops gracefully, let all the existing connections finish)
* -l (list all the loaded modules)
* -t (Perform a syntax check on the conf files

If you run [serverRoot]/bin/apachectl –l you can make sure that mod\_proxy and mod\_ssl got loaded.

## Configuring the HTTPd server

Now change into [serverRoot]/conf and edit the httpd.conf file.

* Find ServerName and uncomment the line, and change the servers name to the correct host.domainname.
* Find Listen and make sure it is set to 8090
* The following needs to be added to the server:

<VirtualHost 50.57.152.213:9090>

ServerAdmin sales@syntegritynet.com

DocumentRoot "/data/httpd-2.2.21/htdocs"

ServerName qa001.easyidam.com

ErrorLog "/data/httpd-2.2.21/logs/<port>-error\_log"

LogLevel debug(What ever you think is needed, debug is too high for production)

TransferLog "/data/httpd-2.2.21/logs/<port>-access\_log"

# Turn off ALL Clients need to have a certificate for access for this port

SSLVerifyClient none

</VirtualHost>

Now the basic Apache HTTPd server can be checked and restarted. (-t checks the syntax, then start)

## Configuring the Reverse Proxy

You will need to download libxml2\_devel to the server with “yum install libxml2\_devel”. Now unzip mod\_proxy\_html in the /data/bits directory.

Switch to user “proxy”, and run these command from the /data/httpd-2.2.21/bin directory:

This command builds the mod\_xml2enc libraries and moducles.

$./apxs -aic -I/usr/include/libxml2 /data/bits/mod\_proxy\_html/mod\_xml2enc.c

This command builds the mod\_proxy\_html libraries and modules.

$./apxs -aic -I/usr/include/libxml2 -I/data/bits/mod\_proxy\_html /data/bits/mod\_proxy\_html/mod\_proxy\_html.c

Now change into [serverRoot]/conf and edit the httpd.conf file. Just above the <VirtualHost> entry add:

LoadFile /usr/lib64/libxml2.so

LoadModule proxy\_html\_module modules/mod\_proxy\_html.so

LoadModule xml2enc\_module modules/mod\_xml2enc.so

LoadModule header\_module modules/mod\_headers.so

The within the <VirtualHost> entry add:

# Setting up the Reverse Proxy of OpenAM

<IfModule proxy\_module>

ProxyHTMLEnable On

ProxyRequests off (Protects the server from hackers)

ProxyPass /openam <http://qa001.easyidam.com:9090/openam> (Which directory to pass and to which server)

ProxyHTMLURLMap http://qa001.easyidam.com:9090/openam /openam (What to change HTML from – to)

<Location /openam>

ProxyPassReverse <http://qa001.easyidam.com:9090/openam> (Pass all openam on HTTPd to Tomcat OpenAM)

SetOutputFilter proxy-html

ProxyHTMLURLMap / /openam/

ProxyHTMLURLMap /openam /openam

RequestHeader unset Accept-Encoding

</Location>

</IfModule>

For every request to the server qa001.easyidam.com:8090/openam, the HTTPd will forward to qa001.easyidam.com:9090/openam, and rewrite all the returning HTML to HTTPd and not OpenAM running on Tomcat.

## Configuring SSL

To create the certificate authority, you begin by running the following command:

$ openssl genrsa -des3 -out certauth.key 1024

This command will prompt you to protect this key with a password. This is important because you will need to use this later. This command will generate a certificate authority key using 1024 as its key size. Most browsers don’t support larger key sizes, so 1024 was what I used. You could make a larger key by simply changing 1024 with 4096, 2048, …

Once the base key has been created, you need to build the CA certificate:

$ openssl req -new -x509 -days 1826 -key certauth.key -out certauth.crt

You will be prompt by this command:

Country: US

State: Virginia

City: Fairfax

Company: Syntegrity Networks, Inc.

Organization: SaaS

Common Name: SaaS CA

The “days” flag needs to be set to a number of days in years, 1 year 365, 2 years 730, etc… I chose 5 years 1826, You will note that I add one day to this because of Leap Year.

To generate a server key:

$ openssl genrsa -des3 -out qa001-server.key 1024

You will be prompt to enter a password, which you want to keep, since you needed it to start the server and load the certificate.

Create a request for a certificate authority to sign:

$ openssl req -new -key qa001-server.key -out qa001-server.csr

It is important to give the server key the name of the server as it is in DNS.

Country: US

State: Virginia

City: Fairfax

Company: Syntegrity Networks, Inc.

Organization: SaaS

Common Name: qa001.easyidam.com

The “common name” will be different for every server and the server’s common name, must not be the same as the certificate authority’s common name.

This is how you sign the server certificate.

$ openssl x509 -req -days 1826 -in qa001-server.csr -CA certauth.crt -CAkey certauth.key -set\_serial 01 -out qa001-server.crt

The cert will last 5 years, it is the first certificate signed by the CA (This is important, each time you sign a server certificate, you need to increment serial number by one. If you don’t, you will get error messages when connecting to servers. Each certificate must have its own common name and its own serial number.

Both the certificate authority and the server certificates need to be in PEM form. It is a two step process to convert a “crt” to a “pem”.

$ openssl x509 -in qa001-server.crt -out qa001-server.der -outform DER

$ openssl x509 -in qa001-server.der -inform DER -out qa001-server.pem -outform PEM

To check or view each cert you can run the following commands:

$ openssl rsa -noout -text -in anyoneofthe.key

$ openssl x509 -noout -text –in anyoneofthe.crt

These commands will give you a large number that will help compare the server key with the server cert. It is simpler to run the command below and just check the certificate serial number.

$ openssl x509 -noout -modulus -in qa001.server.crt | openssl md5

$ openssl rsa -noout -modulus -in qa001.server.key | openssl md5

The output will look like this:

[proxy@qa001 ssl.files]$ openssl x509 -noout -modulus -in qa001.server.crt | openssl md5

(stdin)= fd3de5c5c081d3def142f09b03676c63

[proxy@qa001 ssl.files]$ openssl rsa -noout -modulus -in qa001.server.key | openssl md5

Enter pass phrase for qa001.server.key:

(stdin)= fd3de5c5c081d3def142f09b03676c63

Here is what the “httpd-ssl.conf” should looks like:

<IfModule mod\_ssl.c>

Listen 4430

</IfModule>

AddType application/x-x509-ca-cert .crt

AddType application/x-pkcs7-crl .crl

SSLPassPhraseDialog exec:/data/httpd-2.2.21/conf/ssl.files/passphrase

SSLSessionCache "shmcb:/data/httpd-2.2.21/logs/ssl\_scache(512000)"

SSLSessionCacheTimeout 300

SSLMutex "file:/data/httpd-2.2.21/logs/ssl\_mutex"

<VirtualHost 50.57.152.214:4430>

# General setup for the virtual host

DocumentRoot "/data/httpd-2.2.21/htdocs/secure"

ServerName qa001.easyidam.com

ServerAdmin sales@syntegritynet.com

ErrorLog "/data/httpd-2.2.21/logs/ssl-error\_log"

LogLevel debug

TransferLog "/data/httpd-2.2.21/logs/ssl-access\_log"

# SSL Engine Switch:

SSLEngine on

# SSL Cipher Suite:

SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# Server Certificate:

SSLCertificateFile "/data/httpd-2.2.21/conf/ssl.files/qa001.server.pem"

# Server Private Key:

SSLCertificateKeyFile "/data/httpd-2.2.21/conf/ssl.files/qa001.server.key"

# Certificate Authority (CA):

SSLCACertificatePath "/data/httpd-2.2.21/conf/ssl.files"

SSLCACertificateFile "/data/httpd-2.2.21/conf/ssl.files/certauth.pem"

# Client Authentication (Type):

SSLVerifyClient none

SSLOptions +FakeBasicAuth +ExportCertData

# Per-Server Logging:

CustomLog "/data/httpd-2.2.21/logs/ssl\_request\_log" \

"%t %h %{SSL\_PROTOCOL}x %{SSL\_CIPHER}x \"%r\" %b"

</VirtualHost>

The passphrase file needs to look like this:

#!/bin/sh

echo "password you created with the server key"