

# Installing and Securing the Apache Webserver with SSL

*Dale Coddington* 2000-03-27

## I. Introduction

The purpose of this paper is to describe, in detail, how to install and configure the Apache Web Server with SSL (Secure Socket Layer) support. Today more than ever, sensitive data can no longer be trusted to be sent across the Internet in plain text. SSL adds a layer of encryption to data being sent across an http stream ensuring the that data is transmitted with a large degree of protection from prying eyes. SSL is not the be all and end all of securing data; SSL only protects the data in transit. Once the data has reached its destination it no longer benefits from SSL's encrypted tunnel and could fall prey to prying eyes. This is only too common an occurrence in today's world of e-commerce.

There are commercial packages available that combine the features of Apache with SSL which may also include additional features, including support. This paper will focus on implementing a system that is just as secure as the commercial counter-parts, with no additional cost involved. Using commercial packages is just no fun when you can build it yourself.

## II. Prerequisites

When initially installing your system and creating partitions, create an extra partition called /chroot to run Apache in. How big this partition will be is entirely up to the reader. For a basic web server 40 megs will suffice. We will discuss the purpose of this /chroot partition later.

If you are using these instructions on a system that has already had Apache installed you may choose to add an extra disk to create the /chroot partition on, or elect to not use a /chroot partition, creating the /chroot directory on the root filesystem, or in some other location.

It is also assumed that the reader has already taken measures to the secure the machine they will be installing Apache on. This includes, but is not limited to, removing all

unnecessary SUIDS, upgrading daemons, and disabling non-essential system services. It should go without saying that if you are setting up a machine to serve web pages then that is the only service that should be available on that particular machine. It is also assumed the machine is properly running TCP/IP and has an IP address assigned. Setting up an SSL enabled web server for localhost is rather useless.

### III. Testbed

These instructions have been tested with the following:

- o Slackware 4.x distribution using gcc 2.7.2.3 and Perl v5.005\_02
- o Solaris 7 on Sparc using gcc v2.8.1 and Perl v5.005\_03

### IV. Gathering the Required Software

Unfortunately, Apache does not come with SSL as an included feature, so we must first gather all the pieces of software that will allow us to be able to provide encrypted web based transactions. The following pieces of software will be needed to complete the installation:

- o Apache Web Server - <http://www.apache.org/dist/>  
Obviously we will need to get the web server itself. As of this writing the current version is Apache 1.3.11. The Apache Web Server is still one of the most popular and widely used servers on the internet today.
- o mod\_ssl - <http://www.modssl.org>  
This is a module which provides strong cryptography for the Apache 1.3.x Web Server using the SSL v2 and v3 and TLS (Transport Layer Security) v1 protocols. This package is available under a BSD-style license which means you may download it and use it for free for both non-commercial and commercial purposes. Determining which version of mod\_ssl to grab is simple. The syntax `<mod_ssl-version>-<apache-version>` is used. In this case we will be using the latest version - 2.5.0-1.3.11.
- o mod\_perl - <http://perl.apache.org/dist/>  
This module links the Perl runtime library into the server and provides an object-oriented Perl interface to the server's C language API.
- o Open SSL - <http://www.openssl.org>  
The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets

Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols with full-strength cryptography world-wide. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.

- o RSAref - Do a search for "rsaref20.tar.Z" using your favorite search engine. Users in the US can get it [here](#), as part of the source distribution of PGP 2.6.2. Users outside of the US need not obtain RSAref, and can instead use the routines included as part of OpenSSL.

For the time being we will store all of these programs in /usr/local

Modules are 'plug-in software packages that can give Apache extra functionality. These may include mod\_perl as discussed above or another popular package, mod\_php which adds the popular PHP scripting language to Apache. Any other modules you wish to use will need to be obtained and activated as they normally are.

## V. Installing the Software

Before we actually begin the installation process we must first decide what kind of environment we wish to set the server up in. For the paranoid the server and software may be installed in a chroot'd environment. Chroot changes the root directory and executes a program there. This provides a "contained" environment, outside of your regular file structure, for the server to run. That way if an intruder was to somehow break a cgi script or in some other way gain system access via the web server they will be in a contained or "jailed" environment without access to the entire file system. From a security standpoint this would be desirable but from a system administration standpoint it can be bothersome. It will be necessary to install required libraries, Perl, and any other utilities your web server may need to function in the chroot'd environment. For the sake of security we will be installing the web server in a chroot'd environment. First we will install Apache in the normal fashion then we will transfer it to the /chroot partition that we created at installation time. The installation instructions are provided in a checklist format to facilitate your installation.

- o Extract all the necessary packages:

```
#gzip -d -c apache_1.3.11.tar.gz | tar xvf -
```

```
#gzip -d -c mod_ssl-2.5.0-1.3.11.tar.gz | tar xvf -
#gzip -d -c openssl-0.9.4.tar.gz | tar xvf -
#gzip -d -c mod_perl-1.21.tar.gz | tar xvf -
```

- o Extract and build rsaref

```
#mkdir rsaref
#cd rsaref
#gzip -d -c ../rsaref20.tar.Z | tar xvf -
#tar xvf rsaref.tar
#cp -rp install/unix temp
#cd temp
#make
#mv rsaref.a librsaref.a
#cd ../../
```

- o Build OpenSSL

```
#cd openssl-0.9.4
#perl util/perlpath.pl /usr/bin/perl (Path to Perl)
#./config -L`pwd`/../../rsaref/temp/
#make
#make test
#cd ..
```

- o Apply mod\_perl to the Apache source tree

```
#cd mod_perl-1.21
#perl Makefile.PL APACHE_PREFIX=/usr/local/apache \
    APACHE_SRC=../apache_1.3.11/src \
    USE_APACI=1
```

You will be prompted with the following:

```
Configure mod_perl with ../apache_1.3.11/src ? [y]
```

Hit 'enter' for the default of yes

Next the Makefile will prompt if it should build httpd. Say 'n' for no.

```
#make
#make install
#cd ..
```

- o Apply mod\_ssl patches to Apache source tree

```
#cd mod_ssl-2.5.0-1.3.11
#./configure --with-apache=../apache_1.3.11 \
    --prefix=/usr/local/apache \
    --with-ssl=../openssl-0.9.4 \
    --with-rsa=../rsaref/temp \
    --activate-module=src/modules/perl/libperl.a
#cd ..
```

- o Compile the Apache source:

```
#cd apache_1.3.11
```

Before we actually build Apache, lets add another layer of security. Edit the following file to obscure the version number and name of the server we are using. This will thwart reconnaissance from attackers or otherwise nosey individuals.

```
#<your favorite text editor> src/include/httpd.h
```

Look for the following line (approx. 454) and change the server name/version to something of your liking.

```
define SERVER_BASEVERSION "Apache/1.3.11"
```

Now we can compile Apache

```
#make
```

You can now generate a 'test certificate' or optionally install an actual certificate.

```
#make certificate
```

Follow the instructions to install a certificate. This script is not forgiving of mistakes so carefully.

```
#make install
```

This will place Apache in the `/usr/local/apache` directory. Since we will be moving it to our "chroot jail" this is fine.

- o Testing the web server (non-SSL enabled) - To invoke the web server:

```
/usr/local/apache/bin/apachectl start
```

Once the web server is running you may either use lynx or another browser to hit port 80 on your machine to ensure web pages are being served.

To stop the server:

```
/usr/local/apache/bin/apachectl stop
```

- o Testing the web server (SSL enabled) - To invoke the web server with SSL:

```
/usr/local/apache/bin/apachectl startssl
```

Once the server is running use Netscape or another SSL aware browser to ensure the server is running properly by loading `https://your.ip.here`

To stop the server:

```
/usr/local/apache/bin/apachectl stop
```

## VI. Apache Configuration Files

We will now examine the main Apache configuration files. One thing to keep in mind is changes made to the configuration files will not take effect until the web server is restarted. The configuration files are located in `/usr/local/apache/conf`

- o `httpd.conf` -

This is Apache's main configuration file. You can set such variables as number of `httpd` processes to start initially, number of maximum client connections, ports to listen on, and more. This file is heavily commented and thus very easy to make changes to.

- o `access.conf` -

This configuration file is the default file for the `AccessConfig` directive

in httpd.conf. To simplify things it is recommended you place all of your server directives into the http.conf file and leave this one empty.

- o srm.conf -

This configuration file is the default file for the ResourceConfig directive in httpd.conf. Once again this file can be left empty and all directives can be placed in the httpd.conf configuration file.

To restart the web server so new changes will take effect:

```
#!/usr/local/apache/bin/apachectl restart
```

## VII. Setting Up Apache in the chroot environment

We will now begin the process of preparing our chroot environment and moving the Apache installation and all required files into it. This part of the installation is optional. As stated earlier it is better to add an extra layer of protection and chroot the web server.

- o Create the /chroot directory

```
#mkdir /chroot
```

- o Create the rest of the necessary sub-directories

```
#mkdir /chroot/dev
```

```
#mkdir /chroot/lib
```

```
#mkdir /chroot/etc
```

```
#mkdir /chroot/bin
```

```
#mkdir /chroot/usr
```

```
#mkdir /chroot/usr/local
```

- o Create a /dev/null for our chroot

```
#mknod -m 666 /chroot/dev/null c 1 3
```

- o Copy the Apache distribution over to the /chroot directory

```
#cp -rp /usr/local/apache/ /chroot/usr/local
```

- o Copy required bins (this will vary)

```
#cp /bin/sh /chroot/bin
```

- To determine which libraries will be necessary (this will vary depending on which modules were compiled in)

```
#ldd /usr/local/apache/bin/httpd
```

- Copy required libraries to chroot

```
#cp /lib/libm.* /chroot/lib/  
#cp /lib/libgdbm.* /chroot/lib  
#cp /lib/libdb.* /chroot/lib  
#cp /lib/libdl.* /chroot/lib  
#cp /lib/libc.* /chroot/lib
```

- Copy extra libraries for network functions

```
#cp /lib/libnss* /chroot/lib
```

- Copy the required /etc files to the chroot

```
#cp /etc/passwd /chroot/etc  
#cp /etc/shadow /chroot/etc  
#cp /etc/group /chroot/etc  
#cp /etc/resolv.conf /chroot/etc  
#cp /etc/hosts /chroot/etc  
#cp /etc/localtime /chroot/etc  
#cp /etc/localtime /chroot/etc  
#cp /etc/ld.so.* /chroot/etc
```

- Test the chroot'd Apache

```
#chroot /chroot /usr/local/apache/bin/apachectl start
```

- Bring the server back down

```
#chroot /chroot /usr/local/apache/bin/apachectl stop
```

Everything should be up and running at this point. If not, make sure you have all the required libraries in /chroot/lib. If that doesn't solve the problem you may

want to try invoking httpd with strace. The output from strace may be useful in determining if you are missing any required libraries or binaries.

Now it's time to do a little post-install cleanup and tightening down. By default apache runs as user nobody and group nobody. While this may seem trivial, if you have other software running with that same user and group and apache is compromised, the attacker may have access to those programs as well.

- The passwd, shadow, and host files contained in the /chroot/etc directory contain a lot of entries that do not pertain to httpd.

- Create a user named httpd with a UID of 80
- Create a group named httpd with a GID of 80
- The following commands will overwrite the old files in /chroot/etc

```
#echo "httpd:x:80:100:,,,:/home/httpd:/bin/false" > /chroot/etc/passwd
#echo "httpd:*LK*:11010:0:99999:7:::" > /chroot/etc/shadow
#echo "httpd:x:80:" > /chroot/etc/group
```

- Tighten down file perms

```
#chmod 600 /chroot/etc/passwd shadow group
```

- Now we need to modify Apache's configuration to reflect these changes. Open /chroot/usr/local/apache/conf/httpd.conf with a text editor. Look for the lines with the user and group to run Apache as (approx. line 263) and change them to httpd/httpd.
- Finally if all is in working order we can delete the original Apache installation we did and remove the other source files we used to compile Apache

```
#rm -rf /usr/local/apache
#rm -rf /usr/local/mod_ssl-2.5.0-1.3.11/
#rm -rf /usr/local/mod_perl-1.21/
#rm -rf /usr/local/openssl-0.9.4/
#rm -rf /usr/local/rsaref
```

\*Note - It may be a good idea to keep your original Apache\_1.3.11 directory in place should you decide to later add more modules. Just remember if you do this to move the newly built distribution over to the chroot jail.

- o Set up Apache to invoke automatically at boot time. With a text editor open /etc/rc.d/rc.local and add the following lines

```
echo "Starting Apache-SSL"  
/usr/sbin/chroot/apache/bin/apachectl startssl
```

If all goes well you are now running Apache with SSL support in a chroot'd jail for extra security. These directions are for a simple web server; the more modules you add, the more tweaking will be necessary. One thing to keep in mind is it is a good idea to only keep essential binaries in the chroot jail. SUID binaries in the chroot should also be avoided at all cost.

## Relevant Links

[Apache Webserver website](#)

*Apache.org*

[The Perl on Apache website](#)

*Apache.org*

[The ModSSL website](#)

*Ralf Engelschall*

[The OpenSSL website](#)

*Ralf Engelschall*

[The MIT PGP Distribution Page](#)

*MIT*

[Privacy Statement](#)

Copyright 2006, SecurityFocus