Apache Security with SSL Using Ubuntu

Some SSL background

- Invented by Netscape for secure commerce.
- Only available using Netscape and Netscape Commerce Server.
- Originally only one signing authority, RSA Data Security.
- Eric A. Young created SSLeay, an Open Source SSL implementation.
- OpenSSL project extends SSLeay for public use.
- RSA spun certificate services division to Verisign in 1995.
- Netscape and Microsoft decided to support multiple CA's.
- 1996 the IETF Transport Layer Security (TLS) task force was created. They published RFCs to support an open stream encryption standard.
- TLS is based on SSL version 3.0 with additions. TLS and SSL are just semantics.

What SSL Provides

- Secure communication between client and server.
- SSL protocol works on top of the TCP/IP layer and below the Application layer.
- Provides for authentication using certificates, multiple encryption cipher choices, methods to exchange session keys, and integrity checking.
- Server authentication almost always takes place. Client authentication is optional.
- Once authentication and handshaking are done then data is transmitted using the strongest mutually available cipher over TCP/IP.
- Weaker ciphers have resulted in some potential SSL security holes.

Apache+mod_ssl – What is it?

Together Apache and mod_ssl create a system of security with digital certificates that allows you to offer secure, encrypted connections to your web server.

mod_ssl is an Apache module that adds “secure sockets layer” (ssl) and “transport layer security” (tls) between a web server and it's clients (web browsers).

Apache-ssl – What is it?

The original Apache with SSL software. mod_ssl is a “split” from the apache-ssl project.

Aimed at stability and security with less features.

You can install both Apache-ssl and Apache +mod_ssl via FreeBSD ports, packages, or from source.

What are we going to use?

We'll use Apache Web server version 2.2 with mod_ssl version 2.8.31.

Apache currently runs about 50% of all web sites on the Internet:

mod_ssl is the most popular method for using SSL with apache at this time.

And, the name?

What does “apache+mod_ssl” mean?

Any guesses?...

Apache = A Patchwork of programs

mod = Module (an Apache program)

SSL = Secure Socket Layer

Digital certificates and signatures

If you generate a local digital certificate you can pay a signing authority to verify your certificate and they'll send it back to you with their “signature”.

With the signing authority's signature your certificate will be accepted by clients (web browsers) without additional prompts.

A digitally signed certificate implies trust that you are who you say you are between your server and the clients who connect to it.

How a certificate request is done

To generate a signed digital certificate from a commercial CA for your site (using FreeBSD and openssl) you do the following:

- Generate your own public and private keys using openssl.
- Answer requested information for the CA you choose to use.
- Send your public key and information to the CA.
- The CA will verify you are who you say you are.
- The CA creates a signed, digital certificate with their private key, using your public key and additional information.
- The signed certificate is made available to you.
- You place the certificate file in the appropriate location.
- Apache will now use this for all https requests. If client browsers have the CA's public key, then a secure connection is made without additional prompting.

Issues with certificate requests

- Can you trust the Certificate Authority?
- Maybe you should sign your public key...
- Verisign bought Thawte. Verisign signs the majority of digital certificates. They are US-based.
- How does the CA know who you are?

All these are good reasons to insist on expiration dates in certificates.

Creating a signed certificate locally

- Today we will sign our own certificate using our own private key.
- This can still be useful:
  - Encrypts data.
  - Deals with man-in-the middle attacks after the initial connection and certificate acceptance.
  - It doesn't cost anything!

Installing SSL support cont.

Another form to install mod_ssl is to compile Apache with mod_ssl together from source.

You can download the code from:

- http://www.apache.org/
- http://www.modssl.org/

And, you can specify *many* options that you cannot do, or that are more difficult to do, using the package install or build from port methods.

Configure a digital certificate

Do the following steps:

- mkdir /usr/local/etc/apache/mycert
- cd /usr/local/etc/apache/mycert
- openssl genrsa -des3 -out server.key 2048
- openssl rsa -in server.key -out server.pem
- openssl req -new -key server.key -out \ server.csr (*answer the series of questions*)
- openssl x509 -req -days 60 -in server.csr \ -signkey server.key -out server.crt

OpenSSL is installed with mod_ssl if it's not already on your system.

Configure a certificate cont.

Explanation

```
openssl genrsa -des3 -out server.key 2048
```

generates a 2048 bit RSA key using the OpenSSL libraries. The key is encoded with the des3 (triple des) algorithm.

This key is private.

Configure a certificate cont.

Explanation

```
openssl rsa -in server.key -out server.pem
```

This removes the passphrase from the private key and places the private key in server.pem for future use.

We'll show why this is useful a bit later.

Configure a certificate cont.

Explanation

```
openssl req -new -key server.key -out server.csr
```

This generates a “csr” (Certificate Signing Request) so that you can have the key signed, or to generate a self-signed certificate.

```
openssl x509 -req -days 365 -in server.csr -signkey \
   server.key -out server.crt
```

This generates a certificate that's good for 365 days. You can make this shorter or longer if you wish.

Remove the password

If we use the server.key default file then each time Apache starts you'll be prompted for the passphrase of your private key.

To remove the passphase we'll use the file server.pem in place of the current server.key file. This is the same as server.key, but it's not encoded with a passphrase.

Making the connection

OK, so you have a server.crt (server certificate) file and a server.key file (with our without a passphrase). Now what happens when someone actually connects to your ssl-enabled server?

From http://www.iiitmk.ac.in/~courses/itm108/2004-winter/presentation/ssloverv.ppt

- 10 Steps to an SSL session
  - Client wants document from secure server: https://some.server/document.html

Making the connection cont.

- 10 Steps to an SSL session continued...
  - Checks if certificate was issued by trusted CA.
  - Client compares information in the the certificate with site's public key and domain name.
  - Client tells the server what Cipher suites it has available.
  - The server picks the strongest mutually available cipher suite and notifies the client.
  - The client then generates a session key, encrypts it using the server's public key and sends it to the server

Making the connection cont.

- 10 Steps to an SSL session continued...
  - The server receives the encrypted session key and decrypts it using its private key.
  - The client and the server use the session key to encrypt and decrypt the data they send to each other.

DANE

RFC 6698 has the specification.

- DNS hierarchy is a single tree rather than the SSL root hierarchy.
- DNS can be secured with DNSSEC
- Entries can be put in the zone file to specify the signatures to trust for particular RRs.
- Currently Firefox supports it via a plugin, Chrome has the code written but not committed.

Solving problems

If you cannot connect to the server check the following:

- Check if firewalling software is running and blocking access to port 443.
- Verify that Apache is listening for connections on port 443 using

```
netstat -an | grep LISTEN
```

- To see certificate and/or configuration file errors look in: ==>

Solving problems cont.

See errors in:

- /var/log/messages (tail -f /var/log/messages)
- /var/log/httpd-error.log
- /var/log/ssl_engine_log

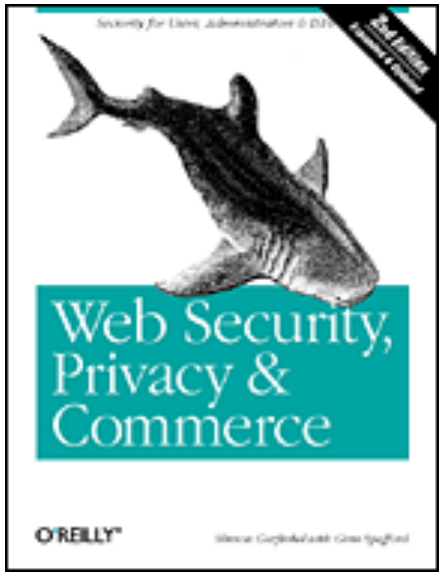
And, as always, you can use:

http://www.google.com/

to look for other people having the same problem.

Understanding SSL: Some resources

- Original Open Source version by Eric Young:
  http://www2.psy.uq.edu.au/~ftp/Crypto/Welcome.html
- Nice published resource:
  Web Security, Privacy & Commerce, 2nd. Ed.
  O'Reilly Press: http://www.oreilly.com/catalog/websec2/index.html
- Apache+mod_ssl:
  http://www.modssl.org/
- Apache-ssl:
  http://www.apache-ssl.org/
- The OpenSSL Project:
  http://www.openssl.org/



Conclusion

The installation of Apache with mod_ssl permits you to run a “secure” web server.

If you run webmail a secure server is *essential* for your security and your client's security.

Apache with mod_ssl=https. This is an extra load on your server. If you have many webmail clients you may need to plan accordingly.

We'll take a look at some of the signing authorities in your web browser now.

Without a signed certificate there is a fundamental problem of trust when connecting to a server.

Exercises

And, now let's install Apache with mod_ssl and generate our own local certificate that we'll sign using our own private key...