Apache Security with SSL Using Linux

These materials are licensed under the Creative Commons *Attribution-Noncommercial 3.0 Unported* license (http://creativecommons.org/licenses/by-nc/3.0/)

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?

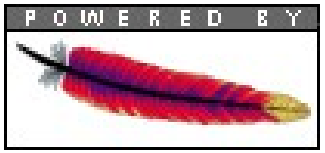
- Together Apache and mod_ssl create a system of security with digital certificates that allows you to offer secure, encrypted connections to your web server.
- mod_ssl is an Apache module that adds “secure sockets layer” (ssl) and “transport layer security” (tls) between a web server
and it's clients (web browsers)

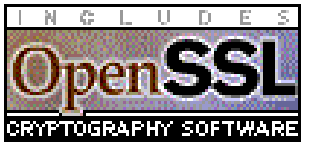
What are we going to use?

- We'll use Apache Web server version 2.2 with mod_ssl.
- Apache currently runs about 50% o 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 Linux 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 support for SSL with Apache

- Use apt-get install apache2
  - This includes SSL by default
- Turn on ssl
  - a2ensite default-ssl
  - a2enmod ssl
- Restart the web server
  - sudo service apache2 reload

Apache2 utilities

- a2enmod and a2dismod are available for enabling and disabling modules utilizing the above configuration system.

- a2ensite and a2dissite do essentially the same thing as the above tools, but for sites rather than modules.

Creating self-signed certificates

- When you install the ssl-cert package, a self-signed certificate will be automatically created using the hostname currently configured on your computer.
- You can recreate that certificate (e.g. after you have changed /etc/hosts or DNS to give the correct hostname) as user root with:

  make-ssl-cert generate-default-snakeoil

  --force-overwrite

Creating self-signed certificates

- To create more certificates with different host names, you can use:

  make-ssl-cert /usr/share/ssl-cert/ssleay.cnf

  /etc/ssl/private/cert-file.crt
- This will ask you for the hostname and place both SSL key and certificate in the file /etc/ssl/private/cert-file.crt .
- Use this file with the SSLCertificateFile directive in the Apache.The file /etc/ssl/private/cert-file.crt should only be readable by root.

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
  - Server sends its certificate to the client.

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
  - Server sends its certificate to the client.

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.

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
```

Solving problems cont.

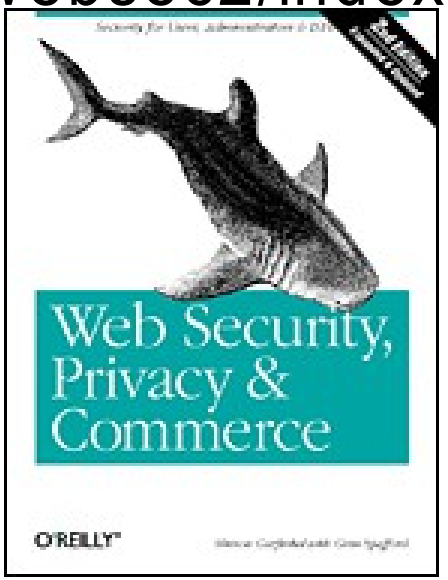
- See errors in:
  - /var/log/apache2/*.log
- And, as always, you can use:

  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/
- 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 usir our own private key...