SSL, TLS, mod_SSL,

And Other Cool Abv.

Orr Dunkelman

September 24, 2001

Topics of the Talk

1. Secure Socket Layer - What to do with it.

2. SSL & TLS - Reasons and History.

3. SSL (& TLS) - How does it work?

4. SSL - Where to get it?

5. SSL - What to do with it.

Motivation for SSL

SSL was introduced by Netscape due to many reasons.

- Authentication mechanism.
- HTTP content secrecy protection.
- HTTP content integrity protection.
- Secure HTTP connections.
- Secure IP connections.

History of SSL & TLS

SSLv1 was designed in Netscape but was never published, and thus in 1994 Netscape introduced SSLv2. Netscape 1.1 supported the standard, and very quickly the protocol was adopted as the standard for secure HTTP connections.

In 1996 Ian Goldberg and David Wagner presented an attack on SSLv2 which took an hour to perform (back then).

In 1995 after Netscape found out that SSLv2 needed some work out, Paul Kocher, Allan Freier and Phil Karlton worked together and published SSLv3.

In 1997 the IETF has started to work on a better version of SSLv3, which supports more ciphers and intended to provide better security and faster implemntations.

How Does it Work?

The basic scenario, is when a browser (client) is trying to approach a server.

The client would like to be assured that:

1. That he talks with the claimed server (i.e., identification of the server).
2. That the contents of the session are known only to him and to the server.
3. That no one altered the content.
4. That no one can add content to the session.
5. That no one can remove content from the session.

Note that in regular TCP connection, the removal of content can be found if the attacker cannot change the appropriate headers.

How Does it Work? (cont.)

The basic SSL connection looks like that:

1. The client sends the server list of supported ciphers, and a random value.
2. The server sends the chosen cipher, certificate and a random value.
3. The client sends encryption of a computed value.
4. Both client and server compute keys.
5. Client sends a MAC of steps 1–3.
6. Server sends a MAC of steps 1–3.

From this moment and after, the packets are encrypted, and each contains MAC (also encrypted).

Detailed Handshake - Steps 1–3

Steps 1–3 are an handshake. During this handshake both client and server agree on the protocol (SSLv2, SSLv3, TLS, PCT, STLP, WTLS) and what ciphers to be used. In the handshake both client and server agree on the private keys to be used during the following session, and the server proves (using a certificate) his identity.

In Step 1 the client sends to the server the list of ciphers he knows along with 256-bit random value. This public random value will be later used to create secret keys (as salt).

In Step 2 the server replies to the client and informs the client on which cipher they are going to use, the public key, and additional 256-bit (public) random value to be used as salt. The server also send a 256-bit random value to be used as session id. Additional data sent by the server is his certificate which proves the server's identity (actually that the sent public key belongs to him).

In Step 3 the client sends additional 384 random bits. This time however, they are encrypted under the server's public key (if RSA is used) or a Diffie Hellman key exchange message. In this message the client acknowledge the chosen ciphers.

Detailed Handshake - Step 4

In Step 4 both client and server compute 6 different keys derived from the two random values from Steps 1–2 (publicly known) and the one from Step 3 (known only to the client and the server).

The 6 keys are: Client's MAC, Server's MAC, Client Write, Server's Write, Client's IV and Server's IV.

Thus, the client will use the IV (initial value) derived and will encrypt using his "write" key. Obvious enough, the MACs the client will compute will be handled by the client's MAC key.

Detailed Handshake - Steps 5–6

In Step 5 the client send the server a MAC of all the handshake. This way th server knows that the client derived the right key (and therefore with very high probability is the real client).

Step 6 is equivalent in the sense that the server's proves to the client he also finished the computation successfully.

Certificates Structure

In the internet trust model, you can trust no one.

Certificates are issued by a *Certificate Authority (CA)*. Among the known and trusted CAs one can find VeriSign and Netscape. Anyone can be a CA, the only question is whether people will trust the certificate issued by the "new" CA.

Certificates are issued according to ANSI X.509 certificate standard. According to the standard the certificate contains: version, serial number, the algorithm used to sign this certificate, issuer, validity, subject (the owner of the certificate), the subject's public key (algorithm + public key itself) and the signature of the CA.

Certificates Structure (cont.)

for example:

version: v1
serial number: 2A 17 EF 73 . . .
signature algorithm: md5WithRSAEncryption
issuer:
C=US
O=RSA Data Security, Inc.,
OU=Secure Server Certification Authority
validity not before: Sat JAN 28 02:21:56 1995
not after: Thu Feb 15 12:23:44 2007
subject:
C=US
ST=Washington
L=Seattle
O=Amazon.com, Inc.
OU=Software
CN=www.amazon.com

Certificates Structure (cont.)

subject public key algorithm: rsaEncryption
modulus bit length: 1024
value: 00 C8 ... AF
public exponent:
bit length: 2
Value: 03
signature value: 03 43 ... 37

Ephemeral Public Key

SSL has an *ephemeral* mode. This mode is used when there are export restrictions on the security of the connection. In the ephemeral mode, a shorter public key is created by the server and is sent to the client in Step 2.

However, the certificate is valid only for the generic public key. Thus, the server signs the new public key, and sends the generic public key (authenticated by the certificate) and the short public key (signed by the generic public key). This way the longer key is used only as a signature key.

Step-Up (or SGC)

Many governments which regulate cryptography usage and export have realized that there are cases where strong cryptography for the people would be a good idea.

The *Step-Up* mechanism by Netscape (or Server Gated Cryptography by Microsoft) actually enables the server to activate the strong cryptography set in the client's side (the set is disabled when the client's software is exported). A server who wish to use Step-Up (or SGC) must have a special certificate which only special CAs are allowed to generate.

Once the client discovers he can use strong crypto he restart the handshake protocol (informing the server which ciphers he now supports).

Various Ciphers

There are many ciphers in the SSL/TLS standards. However, new ciphers are introduced almost daily, some of them are considered much more secure or faster than those in the standards. There are many extensions to the standards which include Elliptic Curves public key ciphers (and signatures), Kerberos based authentication and key exchange (RFC 2712), Fortezza (Skipjack) support, and even the new AES (Rijndael) has an extension.

Session Resumption

SSL supports session resumption. As Steps 1–4 of the handshake are computationally heavy (with respect to other operations), once the key was agreed between the client and the server there is no need to renegotiate it.

In order to resume a previous session, the client sends in Step 1 instead of a random number the session id.

Packages

One can buy SSL/TLS package from many companies. Among the famous ones are Certicom, Netscape and RSA Security.

However, we (i.e., Linux users) have several other options:

- OpenSSL (*http://www.openssl.org*)
- ApacheSSL (*http://www.apache-ssl.org*)
- mod_ssl (*http://www.modssl.org*)

OpenSSL

OpenSSL is an open source project which aim at releasing a full featuring SSLv2/SSLv3/TLS server. It also allows the generation of certificates (note that in many cases you still need a realiable CA to sign it).

The latest release (from 9/7/01) is version 0.9.6b (you get .tar.gz).

The latest RPM release is 0.9.6-1 from 2/1/01 (version 0.9.6 of OpenSSL) and you must update (Pseudo Random Generator weakness fixed).

ApacheSSL

ApacheSSL is an SSL secure web server based on openSSL and apache.

You have to get the integrated package (latest Apache-1.3.19+ssl-1.44 dated 4/6/01), and the OpenSSL package (version 0.9.5a or better). The package comes in source format (.tar.gz).

The latest RPM release is Apache-1.3.12+ssl-1.40 (5/5/00). The reasons for upgrade are due to Apache-1.3.12 and not the SSL extension.

mod_SSL

This is an SSL module for Apache web servers. It is based on OpenSSL, and intended to be used with Apache 1.3.x servers.

The latest version is 2.8.4-1.3.20 from 20/5/01, and is to be used with Apache 1.3.20 (.tar.gz package).

The latest RPM release is apache-mod_ssl-1.3.11.2.5.0-1 (6/2/00) upgrade is advisable (both Apache and OpenSSL needs an update).

Several RFCs and Standards

- RFC 2246 - TLS.
- RFC 2487 - SMTP over TLS.
- RFC 2712 - TLS using Kerberos.
- RFC 2818 - HTTPS (port number - 443).
- ANI X.509 - Certificate standard.

Bibliography

- **SSL and TLS**, Eric Rescorla.
- **RFC 2246**, available on-line at *http://www.faqs.org/rfcs/rfc2246.html*.
- **The SSL protocol Version 3.0**, Freier, Karlton, Kocher, available on-line at *http://home.netscape.com/end/ssl3/draft302.txt*.
- **Cryptography: Theory and Practice**, Stinson.