SSL/TLS

Secure Socket Layer (SSL)

- Is a protocol developed by Netscape in the early 1990s.
- has been developed to provide a secure communication channel for data in transit.
- Operates between TCP/IP and other higher−level protocols.
- In today's Internet society, SSL is primarily used to secure communication between Web browser(client) and a Web server.

History

- SSL 1.0 – never publicly released due to security issues.
- SSL 2.0 – released in 1995. Deprecated in 2011. Has known security issues.
- SSL 3.0 – released in 1996. Deprecated in 2015. Has known security issues.
- TLS 1.0 – released in 1999 as an upgrade to SSL 3.0. Planned deprecation in 2020.
- TLS 1.1 – released in 2006. Planned deprecation in 2020.
- TLS 1.2 – released in 2008.
- TLS 1.3 – released in 2018.

SSL Functions

- Allows an SSL−enabled server to authenticate to an SSL−enabled client.
- Allows an SSL−enabled client to authenticate to an SSL−enabled server.
- Establishes an encrypted communication channel between both the SSL−enabled server and the SSL−enabled client

Authentication of Servers

- Any SSL−enabled server is issued an SSL server certificate by a trustworthy CA.
- This SSL server certificate indicates the integrity and authenticity of the server while the server is communicating with a client.
- An SSL−enabled client can verify the validity and authenticity of the server’s SSL certificate and the public key by using various public key cryptography techniques

SSL authentication process- Step 1

- The SSL–enabled client checks for the validity of the date of the server's certificate and compares it with the current date and time.
- If the current date and time are not within the **validity period** of the server's certificate, the authentication process does not proceed further, else the authentication proceeds.

SSL authentication process- Step 2

- The SSL−enabled client then matches the distinguished name(DN) of the CA that has issued the server certificate with its list of trusted CA certificates.
- The clients accept only those CA certificates that are included in their list of trusted CA certificates.

SSL authentication process- Step 3

- The issuing CA’s digital signature on the server certificate is then validated by using the public key that is present in the CAs certificate (in the list of trusted CA certificates of the client).

SSL authentication process- Step 4

- The SSL–enabled client also checks whether the SSL–enabled server is located in the same network address as specified in the server certificate.
- This step ensures that no man–in–the–middle attacks take place.
- If both the domain names match, then the client can authenticate the server.

Authentication of Clients

- Similar to Server authentication
- Used in situations where the sever need to send some confidential information to a customer over the Internet.

The SSL protocol is made up of the following two layers:

- **SSL Record Protocol:** The SSL Record Protocol operates on a reliable transport medium, such as TCP. SSL Record Protocol encapsulates higher−level protocols.
- **SSL Handshake Protocol:** The SSL handshake Protocol allows a user’s Web browser and a Web server to authenticate each other. After authentication, it negotiates an encryption algorithm and cryptographic keys before the application protocol transmits or receives any data.

SSL Record Protocol

- The record protocol uses the negotiated session key to encrypt data in transit.
- This protocol fragments the data into non−empty blocks.
- These blocks of data can be of any size and can undergo the following stages:
  - Fragmentation ·
  - Compression ·
  - Record Payload protection

SSL Handshake Protocol

- During this stage (around 8 steps), a master secret gets created.
- The client and the server use the master secret to create *session keys*. These session keys are **symmetric** and are used to encrypt and decrypt the data transmitted during the SSL session.

Cryptographic Algorithms Used with SSL

- Key Exchange Algorithm (KEA) ·
- Data Encryption Standard (DES) ·
- Triple−DES ·
- Digital Signature Algorithm (DSA) ·
- Message Digest Algorithm (MD5) ·
- Secure Hash Algorithm (SHA)

Demo

- For a Self Signed Certificate
  - Create a CA
  - Create a Certificate Signing Request (CSR)
  - Sign the CSR by CA
  - Install the Certificate in the Web server
  - Use https in the browser to access the site
  - See the Certificate issued from the server
  - See the Trusted CAs available in the browser
  - Importing Certificates into the browser

UserDir (mod_userdir), SSL (mod_ssl) and Application Layer Firewall (mod_security) on Apache Web Server/CentOS 8.2

Demo Scenario - UserDir

- Enable UserDir
- Reload httpd
- Create a few linux users
- Create public_html folder in their home directory. Put sample html files in those.
- User apache should be able to reach and read the public_html folder and the files inside it.
- Access the site http://rlab.calicut.nielit.in/~username

Scenario - mod_ssl

- Install mod_ssl and reload httpd
- Access the site with https
- Generate csr, submit to CA, issue certificate, install in apache, reload httpd
  - openssl req -newkey rsa:2048 -keyout rlab.key -out rlab.csr -nodes
- Install CA certificate in browser
- Access the site, and see ssl details
- Install valid ssl certificate from LetsEncrypt.org
- Test the site using Qualys.com

Scenario - mod_security

- Create a simple html form with textarea
- Install mod_security, mod_security_crs
- Reload httpd
- Access the site using http, fill the form and submit...with normal text/code/sql injection code
- See the mod_security audit log