

INTRODUCCION

Dado que Secure Sockets Layer (SSL, capa de Sockets Segura) es uno de los protocolos basados en la encriptación más usados para la integridad y la confidencialidad, Sun ha desarrollado la Java Secure Socket Extensión, Extensión de Sockets Segura en Java (JSSE) como extensión de la plataforma de seguridad de Java. La JSSE proporciona una interfaz estándar junto con una implementación de referencia subyacente para construir aplicaciones Java con SSL. La JSSE también se define más genéricamente para proporcionar una interfaz estándar que soporta otros protocolos de socket seguro, como los protocolos Transport Layer Security, Seguridad de Nivel de Transporte (TLS) y Wireles Transport Layer Security, Seguridad de Nivel de Transporte sin Cable (WTLS).

DESARROLLO

SSL

Siglas de Secure Socket Layer. Es un protocolo desarrollado por Netscape Communications Corporation para dar seguridad a la transmisión de datos en transacciones comerciales en Internet. Utilizando la criptografía de llave pública, SSL provee autenticación del servidor, encriptar de datos, e integridad de los datos en las comunicaciones cliente/servidor.

SSL son las siglas para Secure Socket Layer. SSL es un protocolo de comunicación segura entre el servidor y un explorador web.

Es prácticamente lo mismo que http la única diferencia es que toda la comunicación (en ambos sentidos) entre el explorador y el servidor se encripta antes de ser transmitida a través de la Internet.

Es un sistema de seguridad ideado para acceder a un servidor garantizando la confidencialidad de los datos mediante técnicas de encriptación modernas.

CARACTERISTICAS QUE SOPORTA SSL

Confidencialidad

Mediante el uso de la Encriptación se garantiza que los datos enviados y recibidos no podrán ser interpretados por ninguna otra persona que no sea ni el emisor ni el receptor.

Integridad

Se garantiza que los datos recibidos son exactamente iguales a los datos enviados, pero no se impide que al receptor la posibilidad de modificar estos datos una vez recibidos

Autenticación

El vendedor se autentifica utilizando un Certificado Digital emitido por una empresa llamada Autoridad Certificadora, este documento es totalmente infalsificable y garantiza que el Vendedor es quien dice ser.

FUNCIONAMIENTO DE SSL

Para establecer una comunicación segura utilizando SSL se tienen que seguir una serie de pasos. Primero se debe hacer una solicitud de seguridad. Después de haberla hecho, se deben establecer los parámetros que se utilizarán para SSL. Esta parte se conoce como SSL Handshake. Una vez se haya establecido una comunicación segura, se deben hacer verificaciones periódicas para garantizar que la comunicación sigue siendo segura a medida que se transmiten datos. Luego que la transacción ha sido completada, se termina SSL.

Solicitud de SSL:

Antes de que se establezca SSL, se debe hacer una solicitud. Típicamente esto implica un cliente haciendo una solicitud de un URL a un servidor que soporte SSL. SSL acepta solicitudes por un puerto diferente al utilizado normalmente para ese servicio.

Una vez se ha hecho la solicitud, el cliente y el servidor empiezan a negociar la conexión SSL, es decir, hacen el SSL Handshake.

SSL Handshake:

Durante el handshake se cumplen varios propósitos. Se hace autenticación del servidor y opcionalmente del cliente, se determina que algoritmos de criptografía serán utilizados y se genera una llave secreta para ser utilizada durante el intercambio de mensajes subsiguientes durante la comunicación SSL.

El procedimiento que se lleva a cabo para establecer una comunicación segura con SSL es el siguiente:

Client Hello : El "saludo de cliente" tiene por objetivo informar al servidor que algoritmos de criptografía puede utilizar y solicita una verificación de la identidad del servidor. El cliente envía el conjunto de algoritmos de criptografía y compresión que soporta y un número aleatorio. El propósito del número aleatorio es para que en caso de que el servidor no posea un certificado para comprobar su identidad, aún se pueda establecer una comunicación segura utilizando un conjunto distinto de algoritmos. Dentro de los protocolos de criptografía hay un protocolo de intercambio de llave que define como cliente y servidor van a intercambiar la información, los algoritmos de llave secreta que definen que métodos pueden utilizar y un algoritmo de hash de una sola vía. Hasta ahora no se ha intercambiado información secreta, solo una lista de opciones.

Server Hello : El servidor responde enviando su identificador digital el cual incluye su llave pública, el conjunto de algoritmos criptográficos y de compresión y otro número aleatorio. La decisión de que algoritmos serán utilizados está basada en el más fuerte que tanto cliente como servidor soporten. En algunas situaciones el servidor también puede solicitar al cliente que se identifique solicitando un identificador digital.

Aprobación del Cliente: El cliente verifica la validez del identificador digital o certificado enviado por el servidor. Esto se lleva a cabo descriptando el certificado utilizando la llave pública del emisor y determinando si este proviene de una entidad certificadora de

confianza. Después se hace una serie de verificaciones sobre el certificado, tales como fecha, URL del servidor, etc. Una vez se ha verificado la autenticidad de la identidad del servidor. El cliente genera una llave aleatoria y la encripta utilizando la llave pública del servidor y el algoritmo criptográfico y de compresión seleccionado anteriormente. Esta llave se le envía al servidor y en caso de que el handshake tenga éxito será utilizada en el envío de futuros mensajes durante la sesión.

Verificación: En este punto ambas partes conocen la llave secreta, el cliente por que la generó y el servidor por que le fué enviada utilizando su llave pública, siendo la única forma posible de desencriptarla utilizando la llave privada del servidor. Se hace una última verificación para comprobar si la información transmitida hasta el momento no ha sido alterada. Ambas partes se envían una copia de las anteriores transacciones encriptada con la llave secreta. Si ambas partes confirman la validez de las transacciones, el handshake se completa, de otra forma se reinicia el proceso.

Ahora ambas partes están listas para intercambiar información de manera segura utilizando la llave secreta acordada y los algoritmos criptográficos y de compresión. El handshake se realiza solo una vez y se utiliza una llave secreta por sesión.

Intercambio de datos: Ahora que se ha establecido un canal de transmisión seguro SSL, es posible el intercambio de datos. Cuando el servidor o el cliente desea enviar un mensaje al otro, se genera un digest (utilizando un algoritmo de hash de una vía acordado durante el handshake), encriptan el mensaje y el digest y se envía, cada mensaje es verificado utilizando el digest.

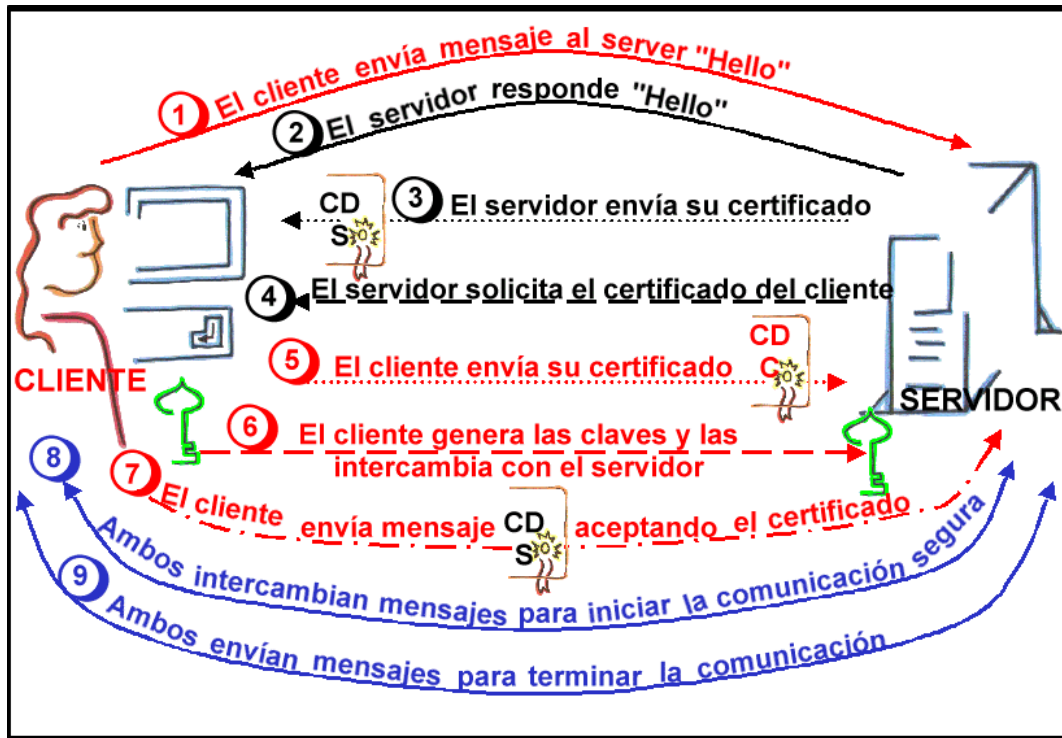
Terminación de una sesión SSL: Cuando el cliente deja una sesión SSL, generalmente la aplicación presenta un mensaje advirtiendo que la comunicación no es segura y confirma que el cliente efectivamente desea abandonar la sesión SSL.

SSL vs. S-HTTP

S-HTTP y SSL utilizan aproximaciones distintas con el fin de proporcionar servicios de seguridad a los usuarios de la Red. SSL ejecuta un protocolo de negociación para establecer una conexión segura a nivel de socket (nombre de máquina más puerto). Los servicios de seguridad de SSL son transparentes al usuario y a la aplicación.

Por su parte, los protocolos S-HTTP están integrados con HTTP. Aquí, los servicios de seguridad se negocian a través de las cabeceras y atributos de la página. Por lo tanto, los servicios de S-HTTP están disponibles sólo para las conexiones de HTTP.

Dado que SSL se integra en la capa de sockets, también permite ser usado por otros protocolos además del HTTP, mientras que el S-HTTP está concebido para ser usado exclusivamente en comunicaciones HTTP.



PROTOCOLO SSL

VENTAJAS CON SEGURIDAD SSL

1. Seguridad Criptográfica:

Se utiliza para establecer una conexión segura entre las dos partes.

2. Interoperabilidad:

Programadores independientes deben ser capaces de desarrollar aplicaciones utilizando S.S.L., capaces de intercambiar parámetros criptográficos sin conocer códigos de otros sistemas.

3. Extensibilidad:

S.S.L. busca proveer una estructura de trabajo en la cuál nuevas llaves públicas y métodos de encriptación puedan ser incorporados de acuerdo a las necesidades, para evitar: Crear nuevos protocolos (arriesgando posibles debilidades) y la necesidad de implementar una nueva librería de seguridad.

4. Eficiencia relativa:

Las operaciones encriptadas tienden a absorber capacidad del CPU, particularmente las operaciones de llaves públicas, por esta razón el protocolo S.S.L. ha incorporado una sesión opcional tomando el plan de reducir el número de conexiones que necesita para

establecerse desde el principio. Adicionalmente se ha tenido cuidado para reducir la actividad del área de trabajo.

PANORÁMICA DE LA EXTENSIÓN DE SOCKET SEGURO DE JAVA

La extensión de Socket Seguro de Java (JSSE) proporciona una API estándar a los protocolos de socket seguro, como SSL. La JSSE también soporta una API del protocolo de socket seguro Seguridad en el Nivel de Transporte (TLS). Se trata de versiones de protocolo específicas que están soportadas:

- SSL versiones 2 y 3.
- TLS versión 1.

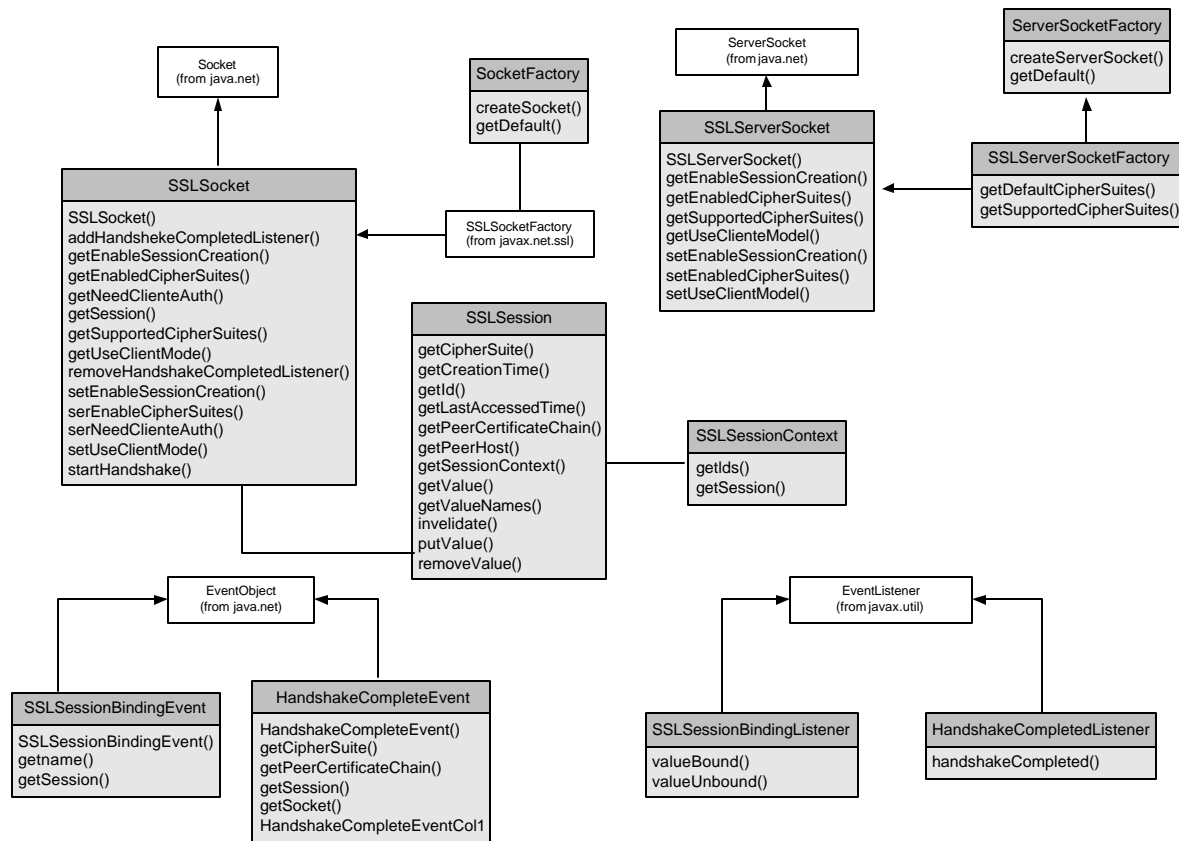
Estas versiones sólo se soportan debido a la API JSSE. La arquitectura JSSE sigue el mismo modelo adaptador de ofrecer una API y una interfaz de proveedor de servicio para las distintas implementaciones subyacentes para conectarse con la JSSE.

CLASES Y PAQUETES JSSE

Los siguientes paquetes componen la arquitectura JSSE v1.0:

- `javax.net.ssl`. Este paquete contiene el conjunto de clases e interfaces nucleares de las API JSSE.
- `javax.net`. Este paquete no es específico en la JSSE, pero es necesario para soportar la funcionalidad básica de factoría de socket de cliente y de socket de servidor.
- `javax.security.cert`. Este paquete no es específico de la JSSE, pero es necesario para soportar una funcionalidad básica de administración de certificados.

La arquitectura de la clase JSSE es útil principalmente para su encapsulación de objetos y factorías de socket SSL y de socket de servidor SSL. También pueden resultar útiles los marcadores de sesión SSL. Por último, también se ofrecen las API de enlace SSL y de evento y audición de despedidas. La siguiente figura ilustra la arquitectura nuclear de la JSSE.



La lista siguiente describe el papel que juega cada clase o interfaz principal de la API en la arquitectura JSSE:

- **SSLSocket**. Un socket que soporta los protocolos de socket seguro SSL, TLS y WTLS.
- **SocketFactory**. Una factoría para los objetos SSLSocket.
- **SSLSocketFactory**. Un socket de servidor que soporta los protocolos de socket seguro SSL, TLS y WTLS.
- **ServerSocketFactory**. Una factoría para los objetos ServerSocket.
- **SSLServerSocketFactory**. Una factoría para los objetos SSLServerSocket.
- **SSLSession**. Una interfaz de un objeto que encapsula una sesión SSL.
- **SSLSessionContext**. Una interfaz de un objeto que encapsula un conjunto de sesiones SSL identificadas con un ID de sesión.
- **SSLBindingEvent**. Una clase de evento que encapsula eventos de enlace y desenlace de sesión SSL.
- **SSLBindingListener**. Una interfaz de auditor que está implementada por objetos que desean conocer los eventos de enlace y desenlace de sesión SSL.
- **HandshakeCompleteEvent**. Una clase de evento que encapsula el hecho de que ha terminado una despedida SSL.
- **HandshakeCompletedListener**. Una interfaz de auditor que está implementada por objetos que desean conocer los eventos de terminación de despedidas SSL.

SOCKETS DE SERVIDOR SSL DE JSSE

La creación de sockets de servidor SSL puede ser tan sencilla como la creación de sockets de servidor normales de Java TCP/IP. Sin embargo, las abstracciones de socket de servidor SSL proporcionan ganchos adicionales para manipular la seguridad y las características de funcionamiento relacionado con SSL que tienen las conexiones de socket SSL. Los parámetros de cifras SSL, las propiedades de autenticación y la administración de despedida también son expuestos por las abstracciones SSL. Aunque existen API estándar que sirven para crear sockets de servidor SSL utilizando factorías de sockets de servidor estándar SSL, los medios con los que se obtienen marcadores de factorías de sockets de servidor SSL pueden ser específicos del proveedor JSSE.

SOCKETS DE CLIENTE SSL DE JSSE

La creación de sockets SSL de cliente a servidor también es muy fácil en JSSE. Los clientes primero deben conseguir marcadores de factorías de sockets SSL. Estos marcadores se consiguen de la forma que los servidores SSL obtienen marcadores con las factorías de sockets de servidor SSL. La creación de sockets de clientes se lleva a cabo utilizando unas cuantas llamadas muy sencillas de creación de sockets. Los sockets SSL se utilizan de forma muy parecida a como los objetos Sockets normales de Java se utilizan para comunicarse con un servidor remoto.

RESUMEN

SSL es una forma popular de encriptar el tráfico entre clientes y servidores. Las sesiones SSL se encriptan con información sobre certificados del lado del servidor y también pueden, opcionalmente, requerir a los clientes que se autentifiquen por sí mismos. La JSSE proporciona una API estándar que encapsula al protocolo SSL para que lo usen clientes y servidores SSL.

BIBLIOGRAFIA

<http://www.urbeinternet.com/soporte/ssl.html>
<http://www.vsantivirus.com/vul-ie-ssl.htm>
<http://www.pedroximenez.com/ssl.htm>
<http://www.delitosinformaticos.com/especial/seguridad/ssl.shtml>
<http://www.seguridata.com/basica/ssl.htm>

- Seguridad en Java
Jaime Jaworski, Paul J. Perrone
pp 251-269