

SSL

Tabla de Contenidos

8. El SSL.....	2
Implementación del protocolo SSL.....	9
Ventajas e inconvenientes de SSL.....	10
Diferencias entre SSL Shell y Telnet.....	12
Cliente de acceso SSH para Acceso Remoto - PuTTY.....	12



8. El SSL.

Toda transacción segura por la red debe contemplar los aspectos de Autenticidad, Integridad, Confidencialidad y No Repudio. Son varios los sistemas y tecnologías que se han desarrollado para intentar implementar estos aspectos en las transacciones electrónicas, siendo sin duda SSL el más conocido y usado en la actualidad. SSL permite la Confidencialidad y la Autenticación en las transacciones por Internet, siendo usado principalmente en aquellas transacciones en la que se intercambian datos sensibles, como números de tarjetas de crédito o contraseñas de acceso a sistemas privados. SSL es una de las formas base para la implementación de soluciones PKI (Infraestructuras de Clave Pública).

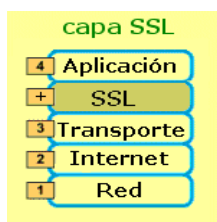
Secure Socket Layer es un sistema de protocolos de carácter general diseñado en 1994 por la empresa Netscape Communications Corporation, y está basado en la aplicación conjunta de Criptografía Simétrica, Criptografía Asimétrica (de llave pública), certificados digitales y firmas digitales para conseguir un canal o medio seguro de comunicación a través de Internet. De los sistemas criptográficos simétricos, motor principal de la encriptación de datos transferidos en la comunicación, se aprovecha la rapidez de operación, mientras que los sistemas asimétricos se usan para el intercambio seguro de las claves simétricas, consiguiendo con ello resolver el problema de la Confidencialidad en la transmisión de datos.

SSL implementa un protocolo de negociación para establecer una comunicación segura a nivel de socket (nombre de máquina más puerto), de forma transparente al usuario y a las aplicaciones que lo usan.

Actualmente es el estándar de comunicación segura en los navegadores web más importantes (protocolo HTTP), como Netscape Navigator e Internet Explorer, y se espera que pronto se saquen versiones para otros protocolos de la capa de Aplicación (correo, FTP, etc.).

La identidad del servidor web seguro (y a veces también del usuario cliente) se consigue mediante el Certificado Digital correspondiente, del que se comprueba su validez antes de iniciar el intercambio de datos sensibles (Autenticación), mientras que de la seguridad de Integridad de los datos intercambiados se encarga la Firma Digital mediante funciones hash y la comprobación de resúmenes de todos los datos enviados y recibidos.

Desde el punto de vista de su implementación en los modelos de referencia OSI y TCP/IP, SSL se introduce como una especie de nivel o capa adicional, situada entre la capa de Aplicación y la capa de Transporte, sustituyendo los sockets del sistema operativo, lo que hace que sea independiente de la aplicación que lo utilice, y se implementa generalmente en el puerto 443. (NOTA: Los puertos son las interfaces que hay entre las aplicaciones y la pila de protocolos TCP/IP del sistema operativo).



SSL proporciona servicios de seguridad a la pila de protocolos, encriptando los datos salientes de la capa de Aplicación antes de que estos sean segmentados en la capa de Transporte y encapsulados y enviados por las capas inferiores. Es más, también puede aplicar algoritmos de compresión a los datos a enviar y fragmentar los bloques de tamaño mayor a 214 bytes, volviendolos a reensamblarlos en el receptor.

La versión más actual de SSL es la 3.0. que usa los algoritmos simétricos de encriptación DES, TRIPLE DES, RC2, RC4 e IDEA, el asimétrico RSA, la función hash MD5 y el algoritmo de firma SHA-1.

Los algoritmos, longitudes de clave y funciones hash de resumen usados en SSL dependen del nivel de seguridad que se busque o se permita, siendo los más habituales los siguientes:

- **RSA + Triple DES de 168 bits + SHA-1:** soportado por las versiones 2.0 y 3.0 de SSL, es uno de los conjuntos más fuertes en cuanto a seguridad, ya que son posibles $3.7 * 10^{50}$ claves simétricas diferentes, por lo que es muy difícil de romper. Por ahora sólo está permitido su uso en Estados Unidos, aplicándose sobre todo en transacciones bancarias.
- **RSA + RC4 de 128 bits + MD5:** soportado por las versiones 2.0 y 3.0 de SSL, permite $3.4 * 10^{38}$ claves simétricas diferentes que, aunque es un número inferior que el del caso anterior, da la misma fortaleza al sistema. Análogamente, en teoría sólo se permite su uso comercial en Estados Unidos, aunque actualmente ya es posible su implementación en los navegadores más comunes, siendo usado por organismos gubernamentales, grandes empresas y entidades bancarias.
- **RSA + RC2 de 128 bits + MD5:** soportado sólo por SSL 2.0, permite $3.4 * 10^{38}$ claves simétricas diferentes, y es de fortaleza similar a los anteriores, aunque es más lento a la hora de operar. Sólo se permite su uso comercial en Estados Unidos, aunque actualmente ya es posible su implementación en los navegadores más comunes.
- **RSA + DES de 56 bits + SHA-1:** soportado por las versiones 2.0 y 3.0 de SSL, aunque es el caso de la versión 2.0 se suele usar MD5 en vez de SHA-1. Es un sistema menos seguro que los anteriores, permitiendo $7.2 * 10^{16}$ claves simétricas diferentes, y es el que suelen traer por defecto los navegadores web en la actualidad (en realidad son 48 bits para clave y 8 para comprobación de errores).
- **RSA + RC4 de 40 bits + MD5:** soportado por las versiones 2.0 y 3.0 de SSL, ha sido el sistema más común permitido para exportaciones fuera de Estados Unidos. Permite aproximadamente $1.1 * 10^{12}$ claves simétricas diferentes, y una velocidad de proceso muy elevada, aunque su seguridad es ya cuestionable con las técnicas de Criptoanálisis actuales.

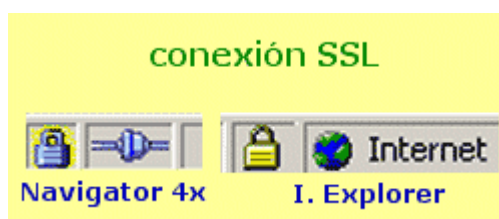
- **RSA + RC2 de 40 bits + MD5:** en todo análogo al sistema anterior, aunque de velocidad de proceso bastante inferior.
- **Sólo MD5:** usado solamente para autenticar mensajes y descubrir ataques a la integridad de los mismos. Se usa cuando el navegador cliente y el servidor no tienen ningún sistema SSL común, lo que hace imposible el establecimiento de una comunicación cifrada. No es soportado por SSL 2.0, pero si por la versión 3.0.

La clave de encriptación simétrica es única y diferente para cada sesión, por lo que si la comunicación falla y se debe establecer una nueva sesión SSL, la contraseña simétrica se generará de nuevo.

SSL proporciona cifrado de alto nivel de los datos intercambiados (se cifran incluso las cabeceras HTTP), autenticación del servidor (y si es necesario también del cliente) e integridad de los datos recibidos.

Durante el proceso de comunicación segura SSL existen dos estados fundamentales, el estado de sesión y el estado de conexión. A cada sesión se le asigna un número identificador arbitrario, elegido por el servidor, un método de compresión de datos, una serie de algoritmos de encriptación y funciones hash, una clave secreta maestra de 48 bytes y un flag de nuevas conexiones, que indica si desde la sesión actual se pueden establecer nuevas conexiones. Cada conexión incluye un número secreto para el cliente y otro para el servidor, usados para calcular los MAC de sus mensajes, una clave secreta de encriptación particular para el cliente y otra para el servidor, unos vectores iniciales en el caso de cifrado de datos en bloque y unos números de secuencia asociados a cada mensaje.

¿Cómo podemos saber si una conexión se está realizando mediante SSL?. Generalmente los navegadores disponen de un icono que lo indica, generalmente un candado en la parte inferior de la ventana. Si el candado está abierto se trata de una conexión normal, y si está cerrado de una conexión segura. Si hacemos doble click sobre el candado cerrado nos aparecerá el Certificado Digital del servidor web seguro.



Además, las páginas que proceden de un servidor SSL vienen implementadas mediante protocolo HTTP seguro, por lo que su dirección, que veremos en la barra de direcciones del navegador, empezará siempre por https, como por ejemplo:

<https://www.htmlweb.net>

Por último, cuando estamos en una conexión segura podemos ver el certificado del servidor acudiendo al menú "Archivo" del navegador y pinchando en "Propiedades". En la parte inferior tenemos una opción "Certificados", que nos

mostrará el del servidor actual.

Vamos a ver a continuación de forma detallada el proceso completo de trabajo de SSL.

Protocolos Secure Socket Layer

Para establecer una comunicación SSL es necesario que previamente el cliente y el servidor realicen un proceso de reconocimiento mutuo y de petición de conexión que, al igual que en otros tipos de comunicaciones, recibe el nombre de apretón de manos o Handshake, que en este caso está controlado por el Potocolo SSL Handshake, que se encarga de establecer, mantener y finalizar las conexiones SSL. Durante el mismo se negocian los parámetros generales de la sesión y los particulares de cada conexión.

Concretamente, y de forma general, el protocolo comienza con el saludo del cliente al servidor, conocido como Client Hello, por el que se informa al servidor de que se deséa establecer una comunicación segura con él. SSL soporta solicitudes de conexión por puertos diferentes al utilizado normalmente para este servicio. Junto con este saludo inicial, el cliente envía al servidor información de la versión de SSL que tiene implementada, de los algoritmos de encriptación que soporta, las longitudes de clave máximas que admite para cada uno de ellos y las funciones hash que puede utilizar. También se le solicita al servidor el envío de su Certificado Digital X.509 v3, con objeto de verificar el cliente la identidad del mismo y recoger su clave pública. En este momento se asigna un identificador a la sesión y se hace constar la hora y fecha de la misma.

Como medida adicional, el cliente envía asimismo una clave numérica aleatoria, para que se pueda establecer una comunicación segura mediante otros protocolos o algoritmos en el caso de que el servidor web no poséa un Certificado Digital.

En este paso no se intercambia en ningún momento información sensible, tan sólo información necesaria para establecer la comunicación segura.

A continuación, el servidor SSL responde al cliente en el proceso que se conoce con el nombre de Server Hello, enviándole su Certificado Digital (con su llave pública) e informándole de su versión de SSL, de los algoritmos y longitudes de clave que soporta.

Generalmente se obtiene el conjunto de algoritmos, longitudes de clave y funciones hash soportados por ambos, eligiéndose entonces los más fuertes. Si no hay acuerdo con los algoritmos a usar se envía un mensaje de error.

A veces, y si la comunicación posterior así lo exige, el servidor solicita al cliente su Certificado Digital, en el mensaje llamado `CertificateRequest`. Esto sólo suele ocurrir en SSL cuando los datos a transferir sean especialmente sensibles y precisen la previa autenticación del cliente. Si es el caso, el cliente debe contestar al servidor mediante el mensaje `CertificateVerify`, enviándole entonces su certificado.

En este momento el cliente verifica la validez del Certificado Digital del servidor, descriptando el resumen del mismo y comprobando su corrección, verificando que ha sido emitido por una Autoridad Certificadora de confianza, que esté correctamente firmado por ella y que el certificado no esté revocado. También se comprueba que la fecha actual está dentro del rango de fechas válidas para el certificado y que el dominio (URL) que aparece en el certificado se corresponde con el que se está intentando establecer la comunicación segura. Si alguna de estas validaciones falla, el navegador cliente rechazará la comunicación, dándola por finalizada e informando al usuario del motivo del rechazo.

En caso de que el servidor no tenga un Certificado X.509 v3 se puede utilizar un mensaje `ServerKeyExchange` para enviar la clave pública sin certificado, en cuyo caso queda en manos del cliente la elección de si acepta la llave o no, lo que finalizaría el proceso.

Como medida adicional de seguridad, el cliente genera una clave aleatoria temporal y se la envía al servidor, que debe devolvérsela cifrada con su clave privada. El cliente la descifra con la llave pública y comprueba la coincidencia, con lo que está totalmente seguro de que el servidor es quién dice ser. Y un proceso análogo a éste, pero en sentido inverso, se requiere si es necesaria la autenticación del usuario ante el servidor.

Si todo está correcto el cliente genera un número aleatorio que va a servir para calcular una clave de sesión correspondiente al algoritmo de encriptación simétrico negociado antes, conocida con el nombre de clave maestra, que es enviada al servidor de forma segura encriptándola asimétricamente con la llave pública del mismo que aparece en el Certificado Digital. Esta clave maestra se usará para generar todas las claves y números secretos utilizados en SSL.

Con esto servidor y cliente se han identificado y tienen en su poder todos los componentes necesarios para empezar a transmitir información cifrada simétricamente.

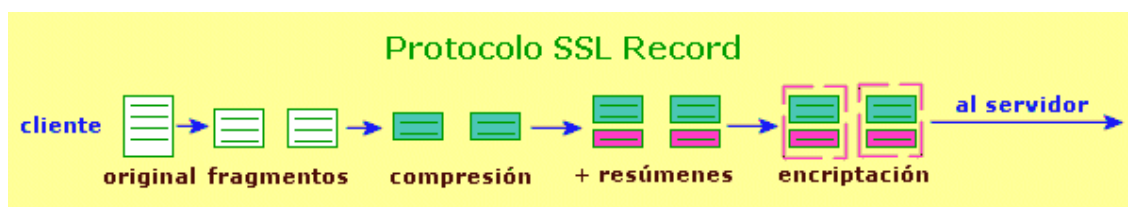
Se pasa entonces el control al subprotocolo `Change Cipher Spec`, iniciándose la conexión segura.

Así y todo, para que empiecen las transmisiones de datos protegidos se requiere otra verificación previa, denominada `Finished`, consistente en que cliente y servidor se envían uno al otro una copia de todas las transacciones llevadas a cabo hasta el momento, encriptándola con la llave simétrica común. Al recibir esta copia, cada host la descripta y la compara con el registro propio de las transacciones. Si las transacciones de los dos host coinciden significa que los datos enviados y recibidos durante todo el proceso no han sido modificados por un tercero. Se termina entonces la fase `Handshake`.

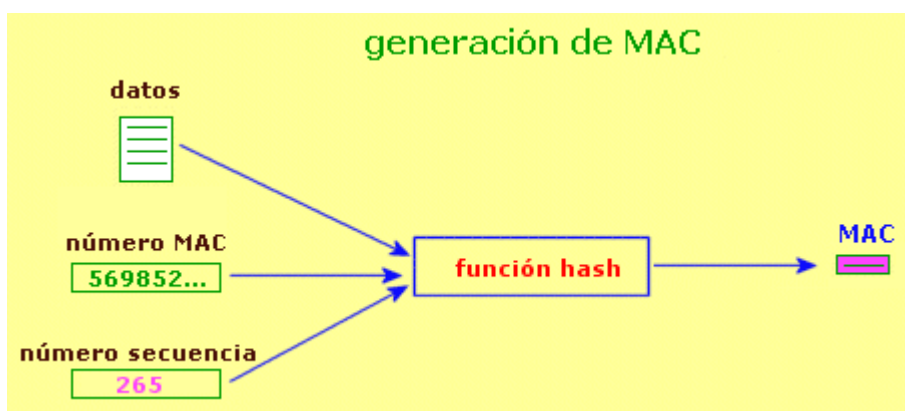
Para empezar a transmitir datos cifrados es necesario que cliente y servidor se pongan de acuerdo respecto a la forma común de encapsular los datos que se van a intercambiar, es decir, qué formato de datos se va a usar en la transmisión cifrada. Esto se realiza mediante el Protocolo SSL Record (Protocolo de Registro SSL), que establece tres componentes para la porción de datos del protocolo:

1. MAC-DATA: código de autenticación del mensaje.
2. ACTUAL-DATA: datos de aplicación a transmitir.
3. PADDING-DATA: datos requeridos para rellenar el mensaje cuando se usa un sistema de cifrado en bloque.

El Protocolo de Registro es el encargado de la seguridad en el intercambio los datos que le llegan desde las aplicaciones superiores, usando para ello los parámetros de encriptación y resumen negociados previamente mediante el protocolo SSL Handshake. Sus principales misiones son:



- La fragmentación de los mensajes mayores de 214 bytes en bloques más pequeños.
- La compresión de los bloques obtenidos mediante el algoritmo de compresión negociado anteriormente.
- La autenticación y la integridad de los datos recibidos mediante el resumen de cada mensaje recibido concatenado con un número de secuencia y un número secreto establecidos en el estado de conexión. El resultado de esta concatenación se denomina MAC, y se añade al mensaje. Con esta base, la autenticación se comprueba mediante el número secreto, compartido por el cliente y el servidor, y mediante el número de secuencia, que viaja siempre encriptado. La integridad se comprueba mediante la función hash negociada.



- La confidencialidad se asegura encriptando los bloques y sus resúmenes mediante el algoritmo simétrico y la clave correspondiente negociadas en la fase Handshake. Existen dos tipos posibles de encriptación:

A). **Cifrado en bloque:** se cifran los datos en bloques de 64 bits. Si el mensaje no es múltiplo de 64 bits se le añaden los bits de relleno necesarios para obtener un número entero de bloques completos, indicándose la adición en el formato del mensaje. Este método de cifrado se conoce con el nombre de Cipher Block Chaining, CBC, y precisa un vector inicial, que habrá sido negociado previamente en la fase Handshake. Como algoritmos de cifrado se usan RC2 y DES.

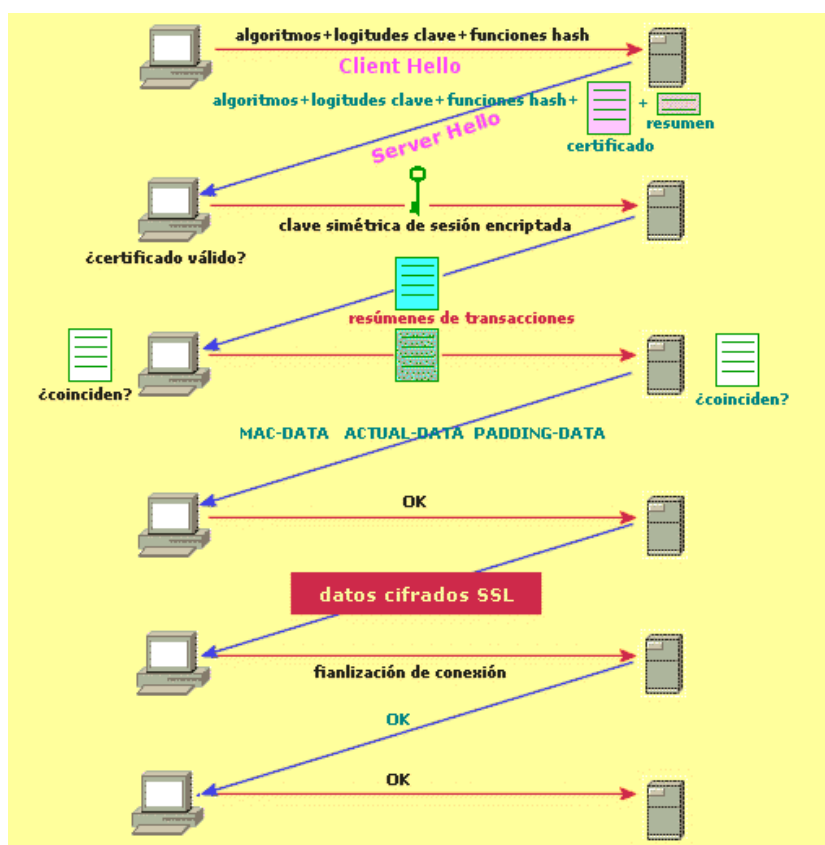
B). **Cifrado Stream:** o de flujo, en el que se encriptan los datos realizando una operación lógica OR-Exclusiva entre los bytes y un generador pseudoaleatorio usando el algoritmo RC4.

Trás todos estos requisitos, el canal seguro está listo para empezar la transmisión de datos de forma segura. Cuando el cliente o el servidor deséan transmitir algún mensaje al otro se genera automáticamente un resumen del mismo mediante la función has acordada, se encriptan mensaje y resumen con la clave simétrica acordada y se envían los datos. Cuando el destinatario los recibe, desencripta todo, vuelve a obtener el resumen a partir del original y lo compara con el recibido. Si coinciden hay seguridad de que la comunicación segura se ha producido satisfactorimente, sin intromisiones externas. Si no coinciden, se pone en conocimiento del otro host, y si es preciso se suspende la conexión SSL. Cada uno de los mensajes enviados por cliente o servidor sufre este proceso de verificación.

Por último, cuando la transferencia de mensajes ha finalizado y se desea cerrar la comunicación segura, generalmente porque el cliente así lo deséa, la aplicación cliente (el navegador web, p.e.) lanza una ventana de aviso de que se va a cerrar la comunicación SSL, y si es aceptada por el usuario, se sale de la misma y se regresa a una comunicación normal, finalizando el proceso SSL.

Una visión gráfica de todo el proceso lo tenéis en este esquema.





Implementación del protocolo SSL

Por la parte del cliente, SSL viene implementado por defecto en los navegadores Internet Explorer y Netscape Navigator, lo que permite a cualquier usuario con uno de estos navegadores poder realizar compras por Internet de forma segura sin tener que conocer el sistema a fondo ni preocuparse de instalar programas adicionales (por lo menos autenticando al servidor web y con confidencialidad e integridad asegurada en la transacción).

La implementación en la parte servidora (la tienda o banco por lo general) es un poco más compleja. En primer lugar, es obligatoria la obtención de un Certificado Digital para el vendedor o para el servidor seguro, solicitándolo a una Autoridad Certificadora de prestigio reconocido, conveniendo, si es posible, que dicha autoridad sea Verisign, ya que la misma está considerada como de toda confianza por los navegadores cliente, por lo que viene activada por defecto en los navegadores cliente.

Ya con el servidor certificado, el usuario podrá realizar su compra. En el momento del pago, el vendedor obtiene el PIN de la tarjeta de crédito del cliente, la fecha de caducidad y sus datos personales (si el pago se realiza por este método), por lo que deberá disponer de algún sistema que permita el envío de estos datos a una entidad financiera capaz de realizar la transferencia bancaria necesaria para completar el pago.

Existen en España diferentes entidades bancarias y financieras que ofrecen estos sistemas a los comerciantes,

realizándose la comunicación entre comerciante y banco a través de un protocolo seguro privado en la mayoría de los casos, de forma similar a lo que ocurre cuando pagamos en una tienda "real" con nuestra tarjeta de crédito. Estos sistemas se suelen conocer con el nombre genérico de Pasarelas de Pago.

Un sistema de pasarela más avanzado es el denominado TPV, Terminal de Punto de Venta. En el mismo se conecta una terminal especial al servidor web del vendedor, y mediante un software basado en script CGI se realiza la comunicación segura entre ellos.

Existen en la actualidad diferentes versiones del conjunto de protocolos SSL que se pueden implementar en los distintos servidores y que corren bajo los sistemas operativos más comunes (IIS en Windows NT-2000-XP, Apache en Unix, etc.).

Ventajas e inconvenientes de SSL

La tecnología basada en los protocolos Secure Socket Layer proporcionó grandes avances en la implantación de sistemas de comunicación seguros, que han hecho posible un crecimiento importante en las transacciones por Internet. Si estudiamos SSL desde el punto de vista de las bases necesarias para considerar una comunicación segura podemos sacar las siguientes conclusiones:

- **Autenticidad:** SSL requiere para su funcionamiento la identificación del servidor web ante el cliente y la realiza adecuadamente, pero normalmente no se produce una identificación en sentido contrario. Es decir, no es obligada en la mayoría de los casos la presencia del certificado del usuario que se está conectando al servidor.

Por ejemplo, una de las aplicaciones más comunes de SSL es el de las aplicaciones bancarias. Cuando nos conectamos a la página web de nuestro banco para consultar las cuentas o realizar alguna operación, el servidor web tan sólo nos pide las contraseñas de acceso, lo que conlleva los típicos problemas a la hora de manejar claves: cambiarlas cada cierto tiempo, mantenerlas bien protegidas, elegir las adecuadamente, etc. Y el tema se complica cuando tenemos que seguir las mismas precauciones con cada una de las diferentes claves que los diferentes bancos y servidores seguros nos requieren.

Otro de los usos comunes de SSL es la protección de números de tarjetas de crédito o débito en compras por Internet. Pero como no se exige el uso del Certificado de Cliente, cualquier persona que obtenga el número de nuestra tarjeta y unos pocos datos personales nuestros puede realizar compras en nuestro nombre. Esto conlleva el tener que prestar mucha atención a los resguardos de nuestras operaciones en cajeros automáticos, a desconfiar cuando un empleado de una tienda o cafetería desaparece con nuestra tarjeta para cobrar el importe de nuestra compra, etc.

Este es precisamente uno de los tipos de fraude más comunes y que causa mayores pérdidas a las compañías de crédito, lo que origina que éstas añadan una comisión en las compras bastante elevada (sobre un 5%), lo

que incrementa el precio final del producto a la venta.

- **Confidencialidad:** SSL proporciona una buena seguridad de que los datos no van a ser capturados por extraños de forma útil en el proceso de transferencia de los mismos, pero no proporciona ninguna seguridad después de finalizar la conexión.

Supongamos que realizamos una compra por Internet, para la cual enviamos los datos de nuestra tarjeta de crédito mediante SSL. Dichos datos quedan en poder del responsable de la tienda, que normalmente los almacena en una base de datos. Con ello, el número de nuestra tarjeta y demás datos quedan en un medio que no controlamos y que no tiene porqué ser seguro, pudiendo tener acceso a los mismos cualquier empleado de la tienda, un hacker que entre en el ordenador en el que reside la base de datos, etc.

- **Integridad:** ocurre algo parecido a lo anterior. En el corto proceso que dura el envío de datos sí podemos estar seguros de que éstos no van a ser modificados, puesto que SSL lo impide. Pero una vez que finaliza la conexión segura no podemos estar tranquilos.

Imaginemos ahora que tras realizar nuestra compra el responsable de la tienda decide cambiar los datos del pedido, y en vez de enviarnos una regrabadora a 30.000 pesetas nos envía 5 a 40.000 pesetas. ¿Qué podemos hacer cuando nos llegen a casa las regrabadoras y la factura del banco?. Nada, protestar, patear y llorar, pero nada más, ya que no hay ningún recibo válido del pedido que hicimos.

- **No Repudio:** en este aspecto SSL falla al máximo, ya que no hay por defecto establecido ningún método para dejar constancia de cuándo se ha realizado una operación, cuál ha sido y quiénes han intervenido en ella. SSL no proporciona formas de emitir recibos válidos que identifiquen una transacción.

Vamos ahora a suponer que realizamos un pedido a una tienda on-line, un ordenador por ejemplo, y que cuando nos llega a casa decimos que nosotros no hemos hecho ninguna compra, devolvemos el ordenador y requerimos la devolución del dinero. ¿Cómo puede demostrar el comerciante que en verdad le hicimos el pedido?. Mediante SSL, de ninguna forma.

A todo esto hay que añadir que SSL sólo proporciona seguridad en la transacción cliente-servidor seguro, pero queda otra fase de la transacción, la que va desde el servidor seguro a la empresa emisora de la tarjeta de crédito, y sobre ésta no tenemos ningún tipo de control.

Con SSL toda la seguridad de la transacción recae en la confianza que el cliente tenga en el vendedor, pues en las manos del mismo está el ser honrado y no realizar ningún fraude con los datos obtenidos y en la posterior entrega del producto comprado. Por este motivo, sólo las empresas con una honradez demostrada podrán a priori ganarse la confianza de los potenciales clientes.

Vemos pues que SSL carece de muchos de los elementos necesarios para construir un sistema de transacciones seguras usando Internet. Para intentar paliar estos fallos se han intentado sacar al mercado y estandarizar otros sistemas diferentes, como SET, que veremos a continuación, pero el caso es que hasta ahora ninguno de ellos ha

conseguido desplazar a SSL. ¿Porqué?.

Tal vez sea porque, a pesar de sus fallos, SSL es una tecnología rápida, fácil de implementar, barata y cómoda para el usuario, que no tiene que conocer cómo funciona, tan sólo usarla. Y desde el punto de vista del comerciante o de la empresa que le facilita el hosting, SSL es igualmente sencillo de implementar, no precisando de servidores de especiales características.

SSL actúa computacionalmente como una máquina de estados: durante el intercambio de datos hay en todo momento un estado de escritura activo y otro pendiente y lo mismo ocurre respecto a la lectura de datos, realizándose el cambio de estados mediante un subprotocolo especial del Handshake denominado Change Cipher Spec.

SSL Handshake posee además otro subprotocolo específico, denominado Alerta, que se encarga de avisar de los problemas que ocurren durante la conexión, y que pueden llevar a la finalización brusca de la sesión.

Diferencias entre SSL Shell y Telnet

Telnet sólo debe ser usado por usuarios avanzados que necesitan ejecutar shell scripts o utilizar comandos shell. Desgraciadamente, por este método tradicional, tanto el login como el password (así como el resto de la sesión) se transmiten en texto claro a través de nuestra red local o incluso a través de routers y nodos ajenos al nuestro. Esto quiere decir que cualquiera que tenga activado un sniffer puede capturar nuestras sesiones con el potencial peligro que ello conlleva.

Si desea acceder remotamente a su servidor de forma segura, utilice el protocolo seguro SSH. Es similar a una sesión telnet tradicional, pero con la particularidad de que todas las comunicaciones serán encriptadas. El manejo de cualquier programa cliente de SSH es muy sencillo.

Cliente de acceso SSH para Acceso Remoto - PuTTY

Existen varios programas para conectarse del protocolo SSH, entre ellos PuTTY.

PuTTY es una implementación libre de Telnet y SSH para plataformas Win32, diseñado y mantenido principalmente por Simon Tatham desde Gran Bretaña:

Nombre: PuTTY

Autor: Simon G. Tatham

Web: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>

Web del autor: <http://www.chiark.greenend.org.uk/~sgtatham/>

Sistema operativo: Windows 95, 98, ME, NT, 2000 y XP en Intel x86

Versión: 0.54



Tamaño: 324 kb

Manual De Acceso SSH Con PuTTY

PuTTY no se instala en su sistema operativo. Descárguelo y ejecútelo directamente. También puede crear un acceso directo a su escritorio para futuras conexiones.

1. Ejecute PuTTY
2. Seleccione la categoría Session
3. Introduzca el nombre de su dominio en el campo Host Name
4. Seleccione el protocolo SSH



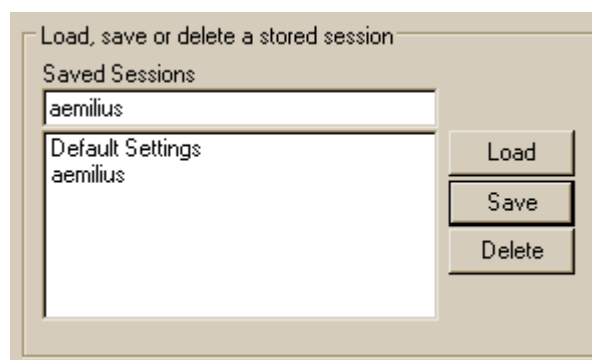
Specify your connection by host name or IP address

Host Name (or IP address)	Port
aemilius.net	22

Protocol:

☐ Raw ☐ Telnet ☐ Rlogin ☒ SSH

5. Introduzca un nombre para esta conexión en el campo Saved Sessions
6. Para guardar la configuración pulse Save

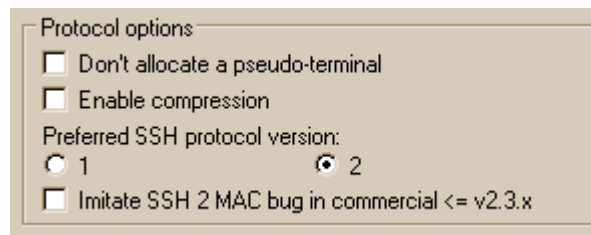


Load, save or delete a stored session

Saved Sessions
aemilius
Default Settings
aemilius

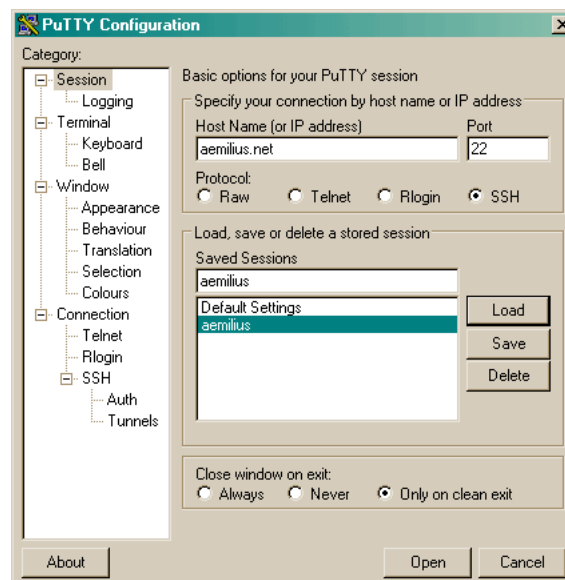
Load Save Delete

7. Seleccione la categoría SSH y en Preferred SSH protocol version seleccione 2



8. Seleccione nuevamente la categoría Session
9. Seleccione la conexión guardada anteriormente y pulse Load

La ilustración muestra un ejemplo de conexión SSH con aemilius.net



10. Para iniciar la conexión pulse Open

Al iniciar la conexión, se abrirá una nueva ventana



11. Introduzca su nombre de usuario y pulse Enter

12. Introduzca su contraseña y pulse Enter

Si el nombre de usuario y password son correctos podrá iniciar la sesión SSH

Nota: Cuando se realiza una conexión SSH por primera vez, el servidor entrega al cliente de SSH la clave pública del servidor. El software de conexión SSH PuTTY le alertará de ello y le ofrecerá la opción de aceptar la clave o rechazarla. Si acepta la clave, se almacenará en el registro y se utilizará para contrastarla con la que el servidor envíe en cada conexión. Si por algún motivo la clave cambia, PuTTY generará un nuevo aviso en el que se planteará la autenticidad de la clave recibida, ya que alguien podría estar haciéndose pasar por el servidor al que nos queremos conectar.

Muy importante:

Para cerrar una sesión no cierre PuTTY como cualquier otra aplicación. Escriba exit y pulse Enter. A continuación ya puede cerrar la aplicación.

