INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA UNIDAD CULHUACAN

SEMINARIO DE TITULACIÓN
"SEGURIDAD DE LA INFORMACIÓN".

ANÁLISIS DE TÚNELES ENTRE LOS PROTOCOLOS SSH Y SSL PARA LA TRANSMISIÓN SEGURA DE LA INFORMACIÓN.

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMUNICACIONES Y ELECTRÓNICA PRESENTAN:

Arias Martínez Nancy Ayde
Martínez Flores Esteban
Saavedra Arvizu Luis Vicente

ASESOR:
Ing. Arturo de la Cruz Téllez.

México D, F. Mayo del 2011

Agradecimientos

Esta tesis esta dedicada a mis Padres, a quien agradezco todo su apoyo, cariño y compresión. Que en todo momento llevo en mi corazón.

Nancy

Mis alegrías, mis triunfos y mis éxitos no son míos son para ti:
Para ti PAPA por haber estado siempre a mi lado construyendo un sueño,
Para ti MAMA que siempre creíste en mi y que sabes que todo lo que hago es pensando en ti,
Y para ti hermana, hermano, tío, tía amiga etc. que eres MI FAMILIA,
Por haber llegado hasta este momento en el que, el sueño se vuelve realidad.

Esteban

Son muchas las personas especiales a las que me gustaria agradecer su amistad, apoyo, animo y compañia en las diferentes etapas de mi vida. Algunas estan aqui conmigo y otras en mis recuerdos y en el corazon. Sin importar en donde esten o si alguna vez llegan a leer esta dedicatoria quiero darles las gracias por formar parte de mi, por todo lo que han brindado y por todas sus bendiciones.
EN ESPECIAL A MI FAMILIA.

Luis

IPN
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD CULHUACAN

TESINA

POR LA OPCIÓN DE TITULACIÓN SEMINARIO EN SEGURIDAD DE LA INFORMACIÓN

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

DEBERÁN DESARROLLAR:

ARIAS MARTÍNEZ MANCY AYDE.
MARTINEZ FLORES ESTEBAN.
SAAVEDRA ARVIZU LUIS VICENTE.

"ANÁLISIS DE TÚNELES ENTRE LOS PROTOCOLOS SSH Y SSL PARA LA TRANSMISIÓN SEGURA DE LA INFORMACIÓN"

INTRODUCCION

LA INFORMACION PUEDE LLEGAR A SER EL ACTIVO MAS IMPORTANTE PARA UNA ORGANIZACIÓN, Y LA NECESIDAD DE TENER EN TODO MOMENTO ESTA INFORMACION, ES UN PUNTO FUNDAMENTAL PARA EL DESARROLLO, LA CONTINUIDAD Y EL CRECIMIENTO DE LA ORGANIZACIÓN, LOS CANALES DE COMUNICACIÓN ACTUALMENTE UTILIZADOS EN INTERNET, CREA UN PUNTO DE ANALISIS PARA LAS MEDIDAS DE SEGURIDAD DE PROTECCION A LA INFORMACION, PARA ELLO ES NECESARIO ANALIZAR LOS CANALES DE COMUNICACION POR DONDE VIAJA LA INFORMACION ATRAVEZ DE INTERNET PARA VERIFICAR O IMPLEMENTAR COMUNICACIONES QUE CUMPLEN CON EL REQUISITO DE DISPONIBILIDAD, INTEGRIDAD Y AUTENTICACION DE LA INFORMACION; PARA VERIFICAR ESTOS MECANISMOS, SE PROPONE UN ANALISIS DE TUNELES IMPLEMETADOS CON LOS PROTOCOLOS DE SEGURIDAD SSH Y SSL.

CAPITULADO

I. INTRODUCCION
II. PROTOCOLOS SSH Y SSL
III. DESARROLLO PRACTICO
IV. PRUEBAS Y RESULTADOS

México D.F., Mayo de 2011

VIGENCIA: DES/ESIME-CUL-2008/23/3/10

ING. ARTURO DE LA CRUZ TELLEZ
Instructor del Seminario

DR. GABRIEL SÁNCHEZ PÉREZ
Coordinador del Seminario

M. EN C. LUIS CARLOS CASTRO MADRID
Jefe de la carrera de I.C.

Índice

Objetivo General. III
Objetivos Específicos. III
Justificación. IV

CAPITULO I. INTRODUCCIÓN. 5
1.1 Seguridad de la información. 5
1.2 Política de seguridad. 8
1.3 Protocolos de red para seguridad. 10
1.4 Conceptos y características para establecer un túnel de comunicación. 12

CAPITULO II. PROTOCOLOS SSH Y SSL. 14
2.1 SSH (Secure Shell) 14
2.1.1 Concepto. 15
2.1.2 Características. 17
2.1.3 Herramientas Putty y OpenSSH. 26
2.2 SSL (Secure Socket Layer) 31
2.2.1 Concepto. 32
2.2.2 Características. 37
2.2.3 Herramienta Stunnel. 38

CAPITULO III. DESARROLLO PRÁCTICO. 41
3.1 Implementación de túneles para el acceso remoto a un servidor de datos MySQL. 41
3.2 Túnel SSH. 45
3.3 Túnel SSL. 58

CAPITULO IV. PRUEBAS Y RESULTADOS. 68
4.1 Velocidad. 69
4.2 Rendimiento. 71
4.3 Vulnerabilidad. 75

CONCLUSIONES. 78

Referencias. 79

Anexos. 80

Índice de Tablas y Figuras.

Tabla 4.1 Resultados de peticiones al servidor MySQL.....69
Tabla 4.2 Resultados de medición de uso de CPU. 73

Figura 2.1 Arquitectura de SSH..... 16
Figura 2.2 Autenticación, cifrado e integridad en SSH. 18
Figura 2.3 SSH Forwarding..... 21
Figura 2.4 Conexión Forwarding. 22
Figura 2.5 SSL con sus sub capas y sus sub protocolos. 33
Figura 2.6 Handshake de cliente y servidor 35
Figura 2.7 Change Cipher Spec de cliente y servidor 36
Figura 3.1 Captura de datos de una comunicación MySQL. 42
Figura 3.2 Escenario de prueba. 44
Figura 3.3 Configuración de sesión putty. 46
Figura 3.4 Usuario de conexión..... 47
Figura 3.5 Putty túnel SSH. 48
Figura 3.8 Túnel SSH. 50
Figura 3.9 Establecimiento del túnel. 51
Figura 3.10 Host key en putty..... 52
Figura 3.11 Autenticación para SSH. 52
Figura 3.13 Prueba de conexión via Telnet. 53
Figura 3.14 Error de conexión. 53
Figura 3.15 Conexión MySQL mediante túnel SSH. 54
Figura 3.19 Esquemea de archivos de configuración para el túnel SSL. 59
Figura 3.20 Stunnel-server-conf..... 60
Figura 3.21 Establecimiento del túnel SSL..... 61
Figura 3.22 Comprobación de puerto en equipo servidor. 61
Figura 3.23 Establecimiento del túnel SSL en el cliente. 63
Figura 3.24 Verificación del servicio stunnel en el cliente. 64
Figura 3.25 Prueba de conexión del túnel SSL en el cliente. 64
Figura 3.26 Respuesta del servidor utilizando el túnel SSL en el cliente. 65
Figura 3.27 Establecimiento túnel SSL en el cliente. 65
Figura 3.28 Consulta realizada al servidor através del túnel SSL. 66
Figura 3.29 Escenario de prueba para el túnel SSL. 67
Figura 3.30 Captura de datos en el túnel SSL..... 67
Figura 4.1 Grafica de tiempos de respuesta en los túneles. 70
Figura 4.2 Uso de CPU sin túnel. 71
Figura 4.3 Uso de CPU túnel SSH. 72
Figura 4.4 Uso de CPU túnel SSL. 72
Figura 4.5 Grafica resultados de medición de uso de CPU..... 73

Objetivo General.

Comparar los protocolos SSH y SSL en la implementación de túneles de seguridad para el envió de información.

Objetivos Específicos.

- Mencionar las características necesarias para establecer un túnel de comunicación.
- Describir las características de los protocolos SSH y SSL.
- Hacer los túneles con las herramientas PUTTY para el protocolo SSH y STUNEL para el protocolo SSL.
- Examinar la velocidad, establecimiento del túnel y la vulnerabilidad en la transmisión de los datos en cada uno de los túneles.

Justificación.

Debido a la necesidad de establecer comunicación hacia un punto externo de la red local y que requieran pasar por medios inseguros, es importante generar un canal seguro para la transmisión de la información ahorrando en infraestructura, optimizando tiempos de implementación y protegiendo de una posible vulnerabilidad.

Todo esto con la finalidad de proteger los datos; para esto se propone crear un canal seguro el cual establece comunicación puerto a puerto indistintamente de la aplicación, de esta manera se minimiza el riesgo de ver comprometida la integridad de los datos.

CAPITULO I. INTRODUCCIÓN.

1.1 Seguridad de la información.

En la actualidad son muchos los factores a tener en cuenta en lo relativo a la seguridad de la información, un área que día a día va adquiriendo más protagonismo en los presupuestos e inversiones empresariales.

Los planes de contingencia y de continuidad de negocio cobran especial relevancia a la hora de abordar cualquier proyecto. Ya son muchos y muy frecuentes los escenarios donde la pérdida de información puede ocasionar daños importantes en los desarrollos corporativos. A lo anterior se suman los ataques externos que vulneran el funcionamiento correcto de los sistemas, incluso la interrupción esporádica de ciertos sistemas y servicios, puede ocasionar importantes pérdidas económicas. Es por esto que la información es el elemento principal a proteger, resguardar y recuperar dentro de las redes empresariales.

Habitualmente los usuarios finales no tienen en consideración la seguridad cuando hacen uso de un sistema, ya que, frecuentemente se ignoran los aspectos relacionados con la seguridad. De igual forma, estos aspectos a veces pueden considerarse una molestia, es por esto que los usuarios en ocasiones puedan tener una imagen negativa de la seguridad, por considerarlo algo molesto y que interrumpe su capacidad de realización de un trabajo determinado.

Seguridad es un concepto asociado a la certeza, falta de riesgo o contingencia. Conviene aclarar que no siendo posible la certeza absoluta, el elemento de riesgo esta siempre presente, independiente de las medidas que tomemos, por lo que debemos hablar de niveles de seguridad. La seguridad absoluta no es posible y en adelante entenderemos que la seguridad informática es un conjunto de técnicas encaminadas a obtener altos niveles de seguridad en los sistemas informáticos. Además, la seguridad informática precisa de un nivel organizativo, por lo que diremos que:

Sistema de Seguridad = TECNOLOGIA + ORGANIZACIÓN.

Con lo anterior para un sistema de seguridad lo más importante es proteger la información.

Si bien es cierto que todos los componentes de un sistema informático están expuestos a un ataque (hardware, software y datos) la información es el sujeto principal de protección para las técnicas de seguridad. La seguridad informática se dedica principalmente a proteger la confidencialidad, la integridad y disponibilidad de la información. [1]

Confidencialidad. La confidencialidad es la propiedad de prevenir la divulgación de información a personas o sistemas no autorizados.

Por ejemplo, una transacción de tarjeta de crédito en Internet requiere que el número de tarjeta de crédito debe ser transmitida desde el comprador al vendedor a través de un sistema de procesamiento de transacciones. El sistema intenta hacer valer la confidencialidad mediante el cifrado del número de la tarjeta y los datos que contiene la banda magnética durante la transmisión de los mismos. Si una parte no autorizada obtiene el número de la tarjeta en modo alguno, se ha producido una violación de la confidencialidad.

La pérdida de la confidencialidad de la información puede adoptar muchas formas. Cuando alguien mira por encima de su hombro, mientras usted tiene información confidencial en la pantalla, cuando se publica información privada, cuando una laptop con información sensible sobre una empresa es robada, cuando se divulga información confidencial a través del teléfono, etc. Todos estos casos pueden constituir una violación de la confidencialidad.

Integridad. Para la Seguridad de la Información, la integridad es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. (No es igual a integridad referencial en bases de datos.) La violación de integridad se presenta cuando un empleado, programa o proceso (por accidente o con mala intención) modifica o borra los datos importantes que son parte de la información, así mismo hace que su contenido permanezca inalterado a menos que sea modificado por personal autorizado, y esta modificación sea registrada, asegurando su precisión y confiabilidad. La integridad de un mensaje se obtiene adjuntándole otro conjunto de datos de comprobación de la integridad: la firma digital es uno de los pilares fundamentales de la seguridad de la información

Disponibilidad. La Disponibilidad es la característica, cualidad o condición de la información de encontrarse a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones.

En el caso de los sistemas informáticos utilizados para almacenar y procesar la información, los controles de seguridad utilizados para protegerlos, y los canales de comunicación protegidos que se utilizan para acceder a ella deben estar funcionando correctamente. La alta disponibilidad en los sistemas informáticos es el objetivo que se debe seguir en todo momento, evitando interrupciones del servicio debido a cortes de energía, fallos de hardware, y actualizaciones del sistema. Garantizar la disponibilidad implica también la prevención de ataques de denegación de servicio.

La disponibilidad además de ser importante en el proceso de seguridad de la información, es además variada en el sentido de que existen diversos mecanismos para cumplir con los niveles de disponibilidad de servicio que se requiera, tales mecanismos se implementan en infraestructura tecnológica, servidores de correo electrónico, de bases de datos, de web etcétera, mediante el uso de clúster o arreglos de discos, equipos en alta disponibilidad a nivel de red, servidores espejo, replicación de datos, redes de almacenamiento (SAN), enlaces redundantes, etc. La gama de posibilidades dependerá de lo que queremos proteger y el nivel de servicio que se quiera proporcionar. [2]

1.2 Política de seguridad.

En un entorno seguro, un usuario se encuentra con tareas que le pueden resultar incómodas (como por ejemplo, recordar contraseñas, cambiarlas periódicamente, etc.) y que pueden limitar las operaciones que puede realizar, así como los recursos a los que se le permite acceder. Generalmente, la seguridad de los sistemas informáticos se concentra en garantizar el derecho a acceder a datos y recursos del sistema configurando los mecanismos de autentificación y control que aseguran que los usuarios de estos recursos sólo posean los derechos que se les han otorgado.

Los mecanismos de seguridad pueden sin embargo, causar inconvenientes a los usuarios. Con frecuencia, las instrucciones y las reglas se vuelven cada vez más complicadas a medida que la red crece. Por consiguiente, la seguridad informática debe estudiarse de modo que no evite que los usuarios desarrollen usos necesarios y así puedan utilizar los sistemas de información en forma segura.

Para ello, resulta importante establecer políticas de seguridad, las cuales van desde el monitoreo de la infraestructura de red, los enlaces de telecomunicaciones, la realización del respaldo de datos y hasta el reconocimiento de las propias necesidades de seguridad, para establecer los niveles de protección de los recursos.

Este tipo de políticas permitirá desplegar una arquitectura de seguridad basada en soluciones tecnológicas, así como el desarrollo de un plan de acción para el manejo de incidentes y recuperación para disminuir el impacto, ya que previamente se habrán identificado y definido los sistemas y datos a proteger.

En este sentido, no son sólo los administradores de informática son los encargados de definir los derechos de acceso; sino sus superiores. El rol de un administrador de informática es el de asegurar que los recursos de informática y los derechos de acceso a estos recursos coincidan con la política de seguridad definida por la organización.

Debido a que el administrador es la única persona que conoce perfectamente el sistema, deberá proporcionar información acerca de la seguridad a sus superiores, eventualmente aconsejar a quienes toman las decisiones con respecto a las estrategias que deben implementarse, y constituir el punto de entrada de las comunicaciones destinadas a los usuarios en relación con los problemas y las recomendaciones de seguridad.

La seguridad informática de una compañía depende de que los empleados (usuarios) aprendan las reglas a través de sesiones de capacitación y de concientización. Sin embargo, la seguridad debe ir más allá del conocimiento de los empleados y cubrir las siguientes áreas:

- Un mecanismo de seguridad física y lógica que se adapte a las necesidades de la compañía y al uso de los empleados.
- Un procedimiento para administrar las actualizaciones.
- Una estrategia de realización de copias de seguridad planificada adecuadamente.
- Un plan de recuperación luego de un incidente.
- Un sistema documentado actualizado.

Sin embargo, la seguridad es fundamental a la hora de afrontar tareas que se realizan en sistemas informáticos ya que son las únicas medidas que pueden garantizar que éstas se realicen con una serie de garantías que se dan por sentado en el mundo físico.

En el mundo intangible de la informática, tan cerca de un servidor están sus usuarios legítimos como los usuarios que hacen uso de la misma red de comunicaciones. Estos usuarios, en el caso de una red global, se cuentan por millones. Algunos serán buenos vecinos pero otros serán agentes hostiles. Para que un sistema sea seguro, deben identificarse las posibles amenazas y por lo tanto, conocer y prever el curso de acción del enemigo.

1.3 Protocolos de red para seguridad.

Un protocolo de red es una estándar o un conjunto de normas que especifica el método para enviar y recibir datos entre varios ordenadores. No existe un único protocolo de red, y es posible que en un mismo ordenador coexistan instalados varios protocolos, pues es posible que un ordenador pertenezca a redes distintas.

Finalmente una aclaración: una conexión de red implica una relación entre ordenadores a muchos niveles: necesitamos una conexión física (cable, etc.) necesitamos manejar los datos transportados; necesitamos un sistema de transporte; necesitamos mostrar los datos. Normalmente los protocolos de red trabajan en grupos, encargándose de aspectos parciales de la comunicación.

A continuación mencionamos algunos protocolos de seguridad establecidos para la realización de comunicaciones entre ordenadores dentro de una red.

SSL. (Secure Socket Layer)
Este protocolo proporciona comunicaciones de seguridad sobre el Internet. Este protocolo da acceso de comunicación para aplicaciones cliente-servidor, esta comunicación esta basada sobre un canal diseñado para prevenir espionaje, manipulación o falsificación de mensajes, actúa en la capa de comunicación y es como un túnel que protege a toda la información enviada y recibida. [3]

SSH. (Secure Shell)
Es un protocolo para la autenticación remota y otros servicios de red seguros sobre una red insegura, este protocolo típicamente trabaja sobre las capas superiores de TCP/IP. Este protocolo puede ser utilizado como base para un gran numero de servicios de seguridad dentro de red, este proporciona cifrado robusto, autenticación de servidor y protección de la integridad. Y en algunos casos proporciona compresión de datos. [4]

IPSec. (Internet Protocol Security)
IPsec es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre IP autenticando y/o cifrando cada paquete IP en un flujo de datos. IPsec también incluye protocolos para el establecimiento de claves de cifrado.

Los protocolos de IPsec actúan en la capa de red, la capa 3 del modelo OSI. Otros protocolos de seguridad para Internet de uso extendido, como SSL, TLS (*Transport Layer Security*) y SSH operan de la capa de transporte (capas OSI 4 a 7) hacia arriba. Esto hace que IPsec sea más flexible, ya que puede ser utilizado para proteger protocolos de la capa 4, incluyendo TCP y UDP, los protocolos de capa de transporte más usados. IPsec tiene una ventaja sobre SSL y otros métodos que operan en capas superiores. Para que una aplicación pueda usar IPsec no hay que hacer ningún cambio, mientras que para usar SSL y otros protocolos de niveles superiores, las aplicaciones tienen que modificar su código.

PPTP. (Point-To-Point Tunneling Protocol)
PPTP permite el intercambio seguro de datos de un cliente a un servidor formando un Virtual Private Network (VPN ó red privada virtual), basado en una red de trabajo vía TCP/IP. El punto fuerte del PPTP es su habilidad para proveer en la demanda, multi-protocolo soporte existiendo una infraestructura de área de trabajo, como INTERNET. Esta habilidad permitirá a una compañía usar Internet para establecer una red privada virtual (VPN) sin el gasto de una línea alquilada.

Esta tecnología que hace posible el PPTP es una extensión del acceso remoto del PPP (*Point-to-Point-Protocol*). La tecnología PPTP encapsula los paquetes ppp en datagramas IP para su transmisión bajo redes basadas en TCP/IP. El PPTP es ahora mismo un boceto de protocolo esperando por su estandarización. Las compañías "involucradas" en el desarrollo del PPTP son: Microsoft, Ascend Communications, 3com / Primary Access, ECI Telematics y US Robotics.

L2TP. (Layer 2 Tunneling Protocol)
L2TP es un estándar de la IETF desarrollado en el RFC 2661 y que combina lo mejor de L2F (*Layer 2 Forwarding*) de Cisco Systems y de PPTP de Microsoft y es u(*Point-To-Point Tunneling Protocol*) utilizado fundamentalmente para creación de VPN _s (Virtual Private Networks). L2TP utiliza PPP para proporcionar acceso telefónico y que puede ser dirigido a través de un túnel que pasa por una red IP como Internet hasta un punto determinado.

L2TP define su propio protocolo de establecimiento de túneles, basado en L2F. El transporte de L2TP está definido para una gran variedad de tipos de paquetes, incluyendo X.25, Frame Relay y ATM.[5]

1.4 Conceptos y características para establecer un túnel de comunicación.

Un túnel, es un de canal de comunicación entre dos equipos informáticos, el cual es altamente seguro y utiliza el protocolo TCP/IP. Básicamente, se encarga de mantener un canal informativo privado entre dos máquinas, encapsulando toda la información que fluya sobre este canal, ya sea cifrada o no, para aportar un nivel de seguridad que satisfaga a los usuarios que utilicen dicho canal de comunicación.

Los equipos informáticos que se encuentren dentro del túnel, mantienen una dirección IP asignada por la red o por los usuarios, lo que hace a este túnel completamente dependiente de la dirección IP de las máquinas que lo están utilizando, para poder establecer una conexión puerto a puerto. Para alcanzar los objetivos de seguridad y de confiabilidad que se plantean en un túnel es necesario apelar a protocolos que se ocupen de la creación de un túnel, al cifrado y el descifrado de los paquetes que son transmitidos.

El túnel es un método por el cual podemos transferir datos de un extremo a otro. La técnica de “tunneling” consiste en encapsular sobre otro encabezado correspondiente al protocolo de túnel, este nuevo encabezado contiene la información necesaria para que el paquete viaje por la red para que lleguen a su destino correspondiente, una vez que están en su destino son desencapsulados y dirigidos al servicio que los solicito.

El túnel es creado encapsulando un protocolo de red dentro de los paquetes del mismo protocolo, que serán llevados por la red real. Adicionalmente, el paquete encapsulado es cifrado por el emisor, en acuerdo con el receptor (el sistema que se encuentra en del otro lado del túnel) de manera que sólo ambos extremos puedan acceder a los datos transportados. Éste tipo de comunicación solo es posible si el protocolo soporta esta facilidad, denominada *modo túnel*. La otra modalidad posible, *modo transporte*, provee protección sólo para protocolos de la capa superior.

De esta forma, el túnel es simplemente la ruta que toman los paquetes encapsulados y cifrados, dentro de un paquete del mismo protocolo. Un atacante puede interceptar los mensajes que viajen por el túnel, pero los datos encapsulados están cifrados y solo pueden ser recuperados por el destinatario final. En el sistema de destino, el mensaje encapsulado es extraído del paquete recibido, descifrado, y re direccionado al servicio al que pertenece en el receptor.

Gracias a esto, una organización puede usar de forma segura una red pública para comunicarse con sus usuarios, ya que los paquetes son cifrados antes de ser enviados a través del “túnel”. Con el uso en modo túnel, el encabezado IP interno (encapsulado) es cifrado, ocultando la identidad del destinatario y del origen del tráfico.

Las características más importantes de los protocolos que soportan “tunneling” son cifrado de datos, autenticación, autorización e integridad de datos; muchas de estas características son posibles gracias al cifrado completo del paquete encapsulado. Una distinción a destacar es el hecho de que un paquete esté encapsulado en otro no implica que esté cifrado, tampoco lo inverso. De esta forma se obtienen distintos beneficios que responden a necesidades y conveniencias específicas.

Se pueden destacar como requerimientos básicos de un protocolo de túnel que cumpla con las siguientes condiciones:

- Autenticación de usuario.
- Asignación de direcciones.
- Compresión de datos.
- Cifrado de datos.
- Administración de llaves.
- Soporte multiprotocolo.

Los protocolos se seguridad SSH y SSL cumplen con algunas de las características mencionadas anteriormente para poder operar en modo túnel y de esta forma poder utilizarlos para establecer túneles de comunicación, con características de seguridad para la protección de la información. De esta forma al crear el canal de comunicación se garantiza con políticas de seguridad la integridad de la información, dejando al usuario la creación, su implementación y la gestión de las políticas de seguridad.

CAPITULO II. PROTOCOLOS SSH Y SSL.

2.1 SSH ***(Secure Shell)***

La privacidad es un derecho humano básico, en redes de computadoras de hoy en día, la privacidad no es garantizada. Gran parte de los datos que viajan en las redes locales o en Internet se transmite como texto plano, puede ser capturado y estar al alcance de cualquiera con un poco de conocimientos técnicos. Como las direcciones de correo que envían información, los archivos que se transmiten entre las computadoras, incluso las contraseñas que escriben pueden ser leídas por otros. Imagínese el daño que se puede hacer si la persona no es de confianza y resulta ser un tercero como un competidor de la CIA, o interceptar las comunicaciones donde se maneja información muy sensible.

La seguridad de la red es un gran negocio ya que las empresas luchan por proteger su información con ciertos mecanismos como cortafuegos, establecer redes privadas virtuales (VPNs), y cifrar los archivos y el canal por donde es transmitida esta. Sin embargo, escondido de todo el bullicio, hay una modesta pero robusta solución que muchas grandes empresas han perdido. Es confiable y razonablemente fácil de usar, barata y disponible para la mayoría de los sistemas operativos de hoy.

Es SSH. Quien protege su red con un bajo costo, es una solución basada en software para mantener los ojos curiosos lejos de los datos en una red. No resuelve todos los problemas de privacidad y seguridad, pero elimina varios de ellos con eficacia. Sus características principales son:

- Un protocolo cliente-servidor para la codificación y transmisión de datos a través de una red.
- Autenticación (reconocimiento) de los usuarios con llaves publicas o privadas, además de la integración opcional de otros sistemas de autenticación como: Kerberos, SecurID, PGP, TIS Gauntlet y PAM.
- La capacidad de agregar seguridad a aplicaciones inseguras de red tales como Telnet, FTP, TCP / IP y muchos otros programas y protocolos, casi resulta transparente para el usuario.
- Implementación para la mayoría de los sistemas operativos.

2.1.1 Concepto.

SSH, es un software popular, poderoso y enfocado para la seguridad de la red. Cuando los mensajes son enviados por un ordenador a la red, SSH cifra automáticamente su contenido. Cuando los mensajes llegan a su destinatario, SSH descifra (decodifica) automáticamente este. El resultado es la encriptación transparente; los usuarios pueden trabajar normalmente, sin saber que sus comunicaciones son codificadas de manera segura en la red.

Además, SSH utiliza algoritmos modernos de cifrado, es seguro y es lo suficientemente eficaz como para ser utilizado dentro de las aplicaciones más importantes de las grandes corporaciones.

SSH tiene una arquitectura cliente-servidor, como se muestra en la Figura 2.1. Un servidor de SSH, por lo general instalado y ejecutado por un administrador del sistema, acepta o rechaza las conexiones entrantes a la computadora. Después, los usuarios ejecutan SSH cliente, por lo general en otros equipos, para hacer peticiones al servidor SSH, tales como "Por favor, iniciar sesión", "Por favor, enviar un archivo", o "Por favor, ejecutar este comando." Todas las comunicaciones entre clientes y servidores están cifradas y protegidas de modificaciónes.

La descripción se ha simplificado, pero debe darle una idea general de lo que es SSH; Por ahora, sólo es preciso conocer que los clientes SSH pueden comunicarse con los servidores SSH a través de conexiones de red cifradas.

Un producto basado en SSH podría incluir clientes, servidores, o ambos. Un producto como Unix contiene generalmente estos clientes y servidores, los de otras plataformas son por lo general solo clientes, aunque servidores basados en Windows están empezando a aparecer.

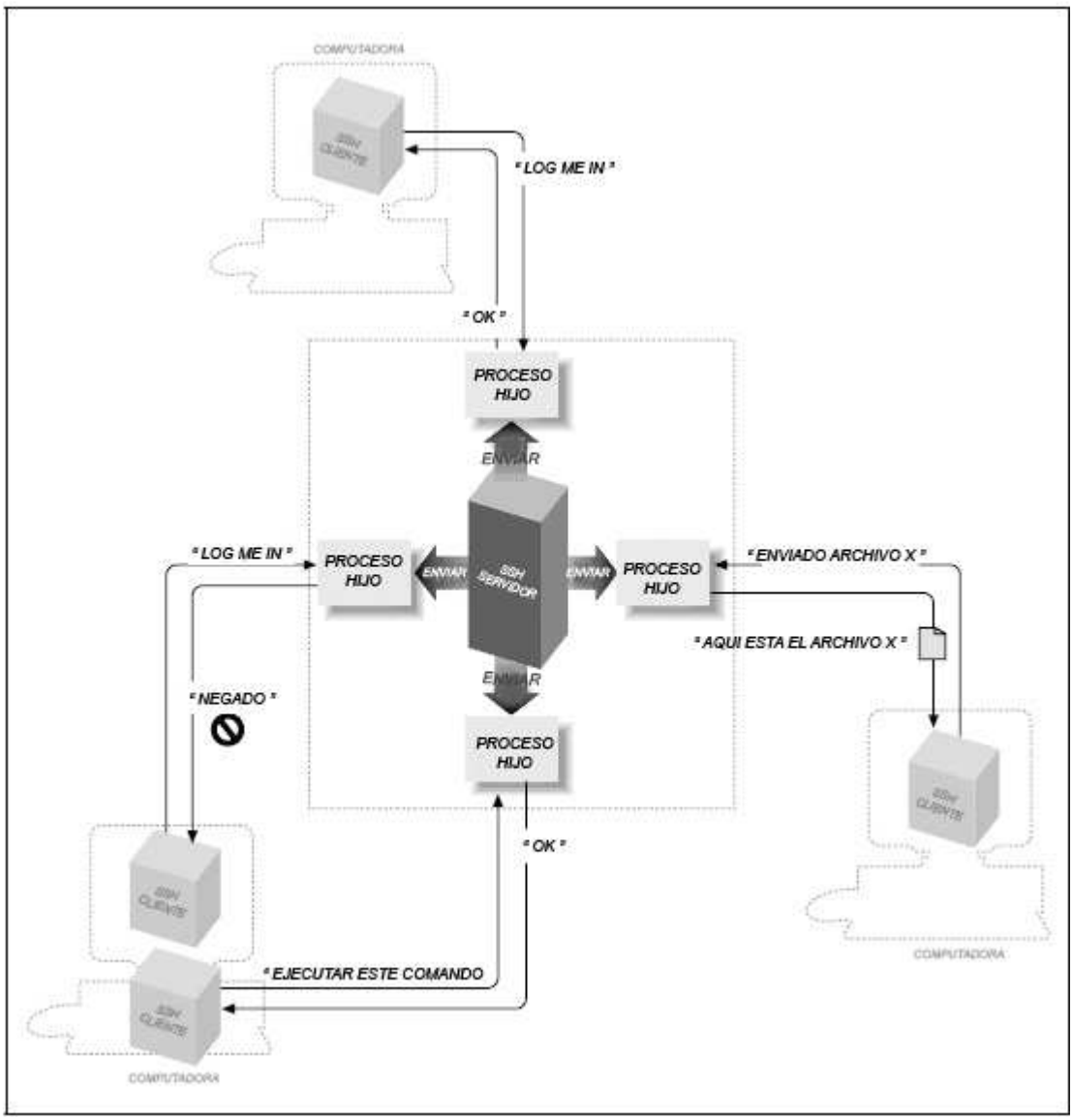

Figura 2.1 Arquitectura de SSH

Si es usuario de Unix, hay que pensar en SSH como una forma segura con los r-comandos: rsh (shell remoto), rlogin (login remoto), y rcp (copia remota). De hecho, el SSH original de Unix incluye comandos con nombres similares como ssh, scp, y slogin as secure, drop-in reemplazo los r-commands. Se puede finalmente deshacerse de los inseguros. rhosts y hosts.equiv files (. A pesar de esto SSH puede trabajar con ellos, si lo desea) Si todavía está utilizando los r-commands, cambie a SSH de inmediato: la curva de aprendizaje es pequeña, y la seguridad es mucho mejor. [6]

2.1.2 Características.

SSH es un protocolo de acceso remoto seguro y otros servicios de seguridad de red a través de una red insegura. Se trata de tres componentes principales: [7]

El Protocolo de la capa de transporte [SSH-TRANS]. Proporciona autenticación, confidencialidad e integridad. En una conexión hacia el servidor; opcionalmente, puede también proporcionar compresión. La capa de transporte típicamente trabaja con conexiones TCP/IP. [8]

El protocolo de autenticación de usuario [SSH-USERAUTH]. Autentica del lado del cliente a un usuario con el servidor, trabaja en la capa de transporte. El nivel de autenticación de usuario utiliza la conexión establecida y se basa en los servicios prestados por la capa de transporte. Se prevé varios mecanismos para la autenticación de usuario. Estos incluyen la autenticación de contraseña tradicional, así como mecanismos de autenticación de clave pública o basada en Host. [9]

El Protocolo de conexión [SSH-CONNECT]. La capa de conexión establece canales diferentes simultáneamente sobre una conexión autenticada y permite hacer un túnel de sesiones de usuario y el tunneling de TCP. Ofrece un servicio de control de flujo para estos canales. Además, diversas opciones específicas para cada canal se pueden negociar. [10]

El cliente envía la solicitud de un servicio, sobre una conexión establecida a través de una capa de transporte segura. Un segundo servicio envía una solicitud después de haber determinado su autenticación. Esto permite que otros protocolos sean utilizados y la vez coexistir con la lista de protocolos aceptados.

El protocolo de conexión proporciona canales que pueden ser utilizados por un amplio rango de propósitos. Algunos métodos estándar proporcionan las características de seguridad para sesiones de Shell seguro y el forwarding (tunneling).

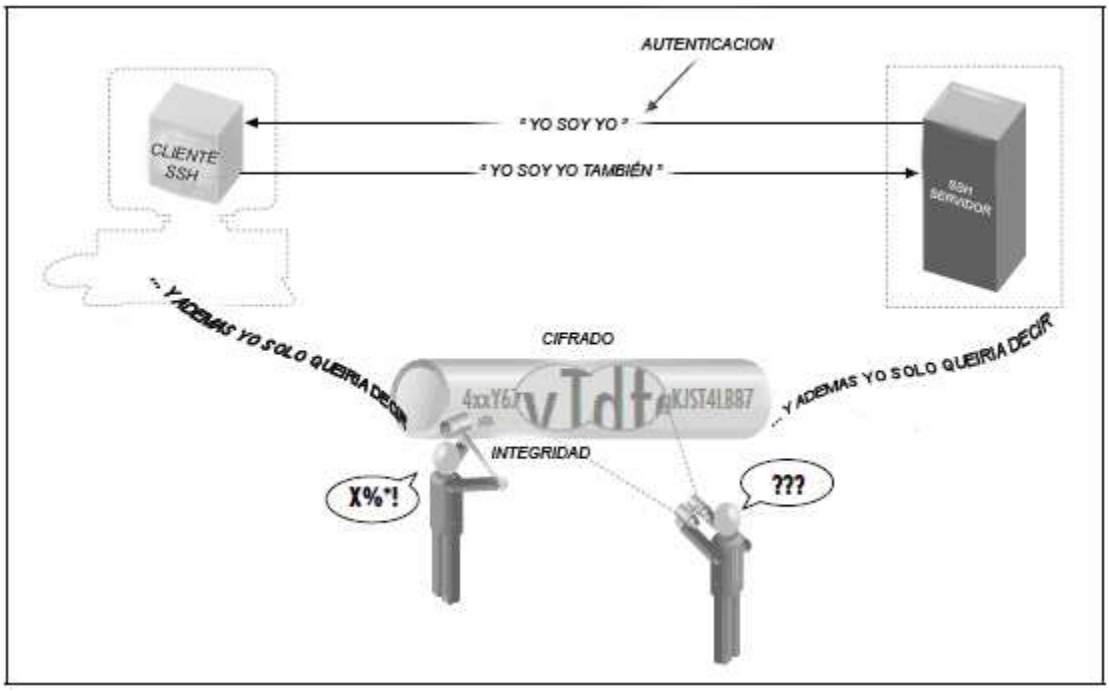

Figura 2.2 Autenticación, cifrado e integridad en SSH.

El protocolo SSH es un protocolo, no un producto. Se trata de una especificación de cómo llevar a cabo la comunicación segura a través de una red. El protocolo SSH cubre la autenticación, el cifrado, la integridad y un dato ligado a la autenticación es la autorización de los datos transmitidos a través de una red, como se muestra en la Figura 2.2. A continuación se definen estos términos:

Autenticación. Determina con fiabilidad la identidad de alguien. Si intenta iniciar sesión en una cuenta en un equipo remoto, SSH pide la prueba digital de su identidad. Si pasa la prueba, se puede acceder, de lo contrario rechaza la conexión SSH.

La autenticación del servidor garantiza que el servidor de SSH es genuino, no un impostor. La autenticación del servidor también protege contra los ataques "man-in-the-middle", en donde el atacante es invisible entre el usuario y el servidor, haciéndose pasar por el cliente en un lado y en el servidor por el otro, engañando a ambos lados y tener la lectura de todo el tráfico en el proceso.

Esta es una gran mejora con respecto a otros protocolos comunes de acceso remoto (Telnet, FTP), que generalmente envían la contraseña en texto claro (es decir, sin cifrar) a través de la red, donde cualquier persona con acceso a la red puede robarla. Sin embargo, todavía es sólo la autenticación de contraseña simple, por lo que SSH proporciona otros mecanismos más fuertes y más manejables: por usuario, llaves públicas, firmas, y una autenticación mejorada rlogin-style, con la identidad del anfitrión verifica por clave pública.

Cifrado. La estructura del mensaje es incomprensible para la mayoría, excepto para los destinatarios. Esto protege los datos a su paso por la red.
SSH proporciona privacidad mediante el cifrado de los mensajes que pasan por la red. Este cifrado de extremo a extremo se basa en claves aleatorias que estén bien negociadas para esa sesión y luego destruidas cuando la sesión ha terminado. SSH soporta una variedad de algoritmos de cifrado para la transmisión de datos, tales como sistemas de cifrado estándar ARCFOUR, Blowfish, DES, IDEA y triple-DES (3DES).

Integridad. Garantiza que los datos que viajan por la red lleguen inalterados. Si alguien captura y modifica los datos en tránsito, SSH detecta este hecho. El transporte subyacente de SSH, TCP / IP, tiene el control de integridad para detectar alteraciones debidas a problemas de red (ruido eléctrico, pérdida de paquetes debido al exceso de tráfico, etc.) Sin embargo, estos métodos no son efectivos contra la deliberada manipulación y puede ser engañado por un atacante inteligente.

A pesar de que SSH cifra el flujo de datos y por lo general un atacante no puede cambiar fácilmente las partes de un archivo para lograr un resultado especifico, el control de integridad de TCP / IP 's no puede impedir, por ejemplo, la inyección deliberada de basura de un atacante en su período de sesión.

Un ejemplo más complejo es un ataque de reproducción. Imagine que un atacante tiene el control de su sesión de SSH (ya sea físicamente o mediante el control de la terminal). En el curso de su trabajo el usuario se ve que escribe el comando rm-rf dentro de un pequeño directorio. Él atacante no puede leer los datos cifrados por SSH durante el período de la sesión y no pudo correlacionar la información capturando los paquetes que contienen la versión cifrada de su comando.

Más tarde, cuando el usuario está trabajando en su directorio personal, el atacante inserta los bits que capturo en su sesión de SSH, y en la terminal misteriosamente se borran todos los archivos. El ataque de reproducción tiene éxito porque los paquetes que se insertan son válidos, el no hubiera podido reproducir esos paquetes (debido al cifrado), pero los puede copiar y reproducir más tarde.

La integridad de TCP / IP 's se realiza sólo en función de cada paquete, por lo que no puede detectar un ataque de este tipo. Es evidente que la comprobación de integridad debe aplicarse a la secuencia de datos como un todo, asegurándose de que los bits que lleguen son los mismos que se enviaron en orden y sin duplicación.

El protocolo SSH-2 utiliza la comprobación de integridad criptográfica, que verifica que los datos transmitidos no han sido alterados y que realmente vienen del otro extremo de la conexión. SSH-2 utiliza llaves con algoritmo hash basado en MD5 y SHA-1 para este fin: es un algoritmo conocido y de amplia confianza. SSH-1, por

el contrario, utiliza un método relativamente débil: un control de redundancia cíclica de 32 bits (CRC-32) sobre los datos no cifrados en cada paquete.

Autorización. Significa lo que alguien puede o no hacer. Esto ocurre al alterar la autenticación, ya que no puede conceder privilegios a alguien hasta que sepa quién es. Los servidores SSH tienen varias maneras de restringir las acciones de los clientes. El acceso a sesiones interactivas de entrada, el puerto TCP y el reenvío de ventanas X (X windows forwarding), el reenvío de agente clave (key agent forwarding), etc,

Todo puede ser controlado, aunque no todas estas características están disponibles en todas las implementaciones de SSH, y no siempre son tan generales o flexibles como uno puede desear. La autorización podrá ser controlando en un nivel serverwide (por ejemplo, el archivo / etc / sshd_config para SSH1), o por su cuenta, dependiendo del método de autenticación utilizado (por ejemplo, los archivos de cada usuario ~ / .ssh / authorized_keys, ~ / .ssh2/authorization , ~ /. shosts, ~ /. k5login, etc.)

En resumen, SSH permite conexiones de red entre los equipos, con sólidas garantías de que las partes en ambos extremos de la conexión son auténticas. También asegura que cualquier dato que pasa sobre estas conexiones llega sin modificaciones y no leídos por espías.

Las principales características y garantías del protocolo SSH son:

- La privacidad de sus datos, a través de un fuerte cifrado.
- Integridad de las comunicaciones, garantizando que no se han modificado.
- Autenticación, es decir, la prueba de la identidad de los remitentes y receptores.
- Autorización, es decir, controlar el acceso a las cuentas.
- El tunneling o túneles para cifrar otras sesiones TCP / IP.

Forwarding (*tunneling*). Tunneling esto es, encapsula otro servicio basado en TCP, como Telnet o IMAP, dentro de una sesión SSH. Esto hace que los beneficios de seguridad de SSH (privacidad, integridad, autenticación y autorización) pasen a otros servicios basados en TCP. Por ejemplo, una conexión Telnet ordinaria transmite su nombre de usuario, contraseña, y el resto de su sesión de inicio en texto claro. Pero el forwarding de telnet a través de SSH, todos estos datos se cifran automáticamente y la integridad es comprobada, y es posible autenticarse con credenciales SSH.

SSH es compatible con tres tipos de transmisión. Port Forwarding, X forwarding incluye características adicionales para garantizar el protocolo X (es decir, X windows). El tercer tipo, el agente de forwarding, permite a los clientes SSH intercambiar claves públicas en las máquinas remotas.

En algunas situaciones sin embargo, la transparencia es difícil de lograr. Un firewall de red puede estar a la mitad del camino, lo que interfiere con el tráfico de red. Las políticas corporativas de seguridad le puede prohibir el almacenamiento de claves SSH en algunas máquinas. O puede que tenga que utilizar aplicaciones inseguras dentro de la red en un entorno seguro.

SSH puede cifrar en un flujo transparente (texto plano) de otra aplicación de datos. Esto se conoce como el forwarding de puertos. El forwarding de SSH no es completamente transparente, ya que se produce a nivel de aplicación, no a nivel de la red. Las solicitudes deberán ser configuradas para participar en Forwarding, algunos protocolos tienen alguna problemática al transmitir (canales de datos FTP es un ejemplo notable).

Una vez que un túnel seguro está configurado, las solicitudes de participación se muestran al usuario para operar normalmente. Transparencia total a nivel de aplicación, necesita una técnica a nivel de red, tales como IPSEC o una VPN (Virtual Private Network). Mientras que las VPN proporcionan una solución más completa, requieren mucho más trabajo y dinero para establecer a comparación del forwarding de SSH.

Así, cuando decimos "transparente", nos referimos a "transparente para la aplicación, una vez que se ha hecho un poco de configuración."

¿Qué es Forwarding? El forwarding es un tipo de interacción con otra aplicación de red, como se muestra en la Figura 2.3 SSH intercepta una solicitud de servicio de algún otro programa de un lado de una conexión SSH, lo envía a través de la conexión cifrada, y lo entrega al destinatario en el otro lado. Este proceso es casi transparente a ambos lados de la conexión, cada uno cree que está hablando directamente a su socio y no tiene conocimiento de Forwarding que se lleva a cabo.

Figura 2.3 SSH Forwarding

Incluso con el forwarding de SSH se puede lograr la comunicación que en algunos casos son imposibles sin ella. El Forwarding no es un concepto nuevo. El funcionamiento básico de una conexión de terminal en una red (por ejemplo, a través de telnet) es también una especie de forwarding. En una conexión telnet, usted se sienta en un extremo, el shell remoto del otro lado y ambas partes actúan como si estuvieran directamente conectados por un cable serial.

Sin embargo, sentado en medio esta un cooperante Telnet cliente y servidor. El forwarding de SSH es muy similar, excepto que SSH juega trucos de lujo con los datos para aumentar la seguridad. También hemos visto otro tipo de forwarding de SSH, el agente de forwarding. Esto nos permite crear conexiones SSH desde un ordenador, a través de un segundo equipo, y un tercio de autenticación mediante una clave pública, pero sin necesidad de instalar nuestra clave privada en el segundo equipo.
Para lograr esto, un servidor de SSH fingió ser un agente de SSH, mientras de forma transparente reenvía datos desde un agente alejado. Este paradigma es válido para el forwarding de puertos TCP y redireccionamiento de X (X forwarding), como el servidor de SSH de forma transparente se disfraza de otra aplicación de red.

Redireccionamiento de puertos (Port Forwarding). SSH utiliza TCP / IP como mecanismo de transporte, por lo general el puerto TCP 22 de la máquina servidor, ya que cifra y descifra el tráfico que pasa a través de la conexión. Es característica que cifra y descifra el tráfico TCP / IP que pertenecen a otras aplicaciones, en otros puertos TCP, usando SSH. Este proceso, es llamado Port Forwarding (túnel de puertos), es muy transparente y muy poderosa.

Telnet, SMTP, NNTP, IMAP y otros protocolos TCP inseguros, pueden asegurarse por Forwarding a través de SSH. El redireccionamiento de puertos es a veces llamado efecto túnel, porque la conexión SSH proporciona un "túnel seguro" a través del cual otra conexión TCP / IP puede pasar. Como se muestra en la figura 2.4. Suponga que tiene una máquina de origen H que ejecuta un lector de correo electrónico IMAP, que desea conectarse a un servidor IMAP en la máquina S para leer y enviar correo.

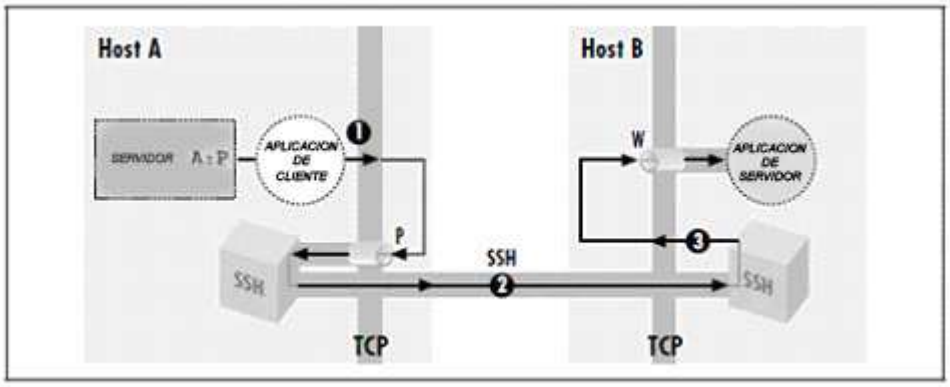

Figura 2.4 Conexión Forwarding.

Normalmente, esta conexión es insegura, introduciendo la contraseña de su cuenta de correo como texto sin formato seguro de transmisión entre su programa de correo y el servidor. Con el redireccionamiento de puertos SSH, puede redirigir de forma transparente la conexión IMAP (que se encuentra en el puerto S servidor TCP 143) para así pasar a través de SSH de forma segura y cifrando los datos a través de la conexión.

La máquina del servidor IMAP debe ejecutar un servidor SSH para el reenvío de puertos para ofrecer una protección real. En resumen, con los cambios de configuración mínima para sus programas, el redireccionamiento de puertos SSH protege arbitrariamente conexiones TCP / IP mediante la reorientación de ellos a través de una sesión SSH. El Forwarding de puertos, incluso puede pasar de una conexión segura a través de un firewall si se configuran las cosas bien.

Éstos son ejemplos de lo que puede hacer:

- Acceso a diferentes tipos de servidores TCP (por ejemplo, SMTP, IMAP, POP, LDAP, etc) a través de un firewall que impide el acceso directo.
- Proporcionar protección para sus sesiones con estos mismos servidores TCP, la prevención de la divulgación o alteración de las contraseñas y otros contenidos que de otra forma sería enviado en texto claro durante la sesión.
- Túnel de conexión en una sesión de FTP, para cifrar su nombre de usuario, contraseña y los comandos. (Normalmente no es posible para proteger a los canales de datos que llevan el contenido del archivo).
- Utilizar servidores de su ISP's SMTP para el envío de correo, incluso si está conectado fuera de la red del ISP y el proveedor de Internet prohíbe la retransmisión de correo desde su ubicación actual.

Reenvió local (Forwarding Local). En nuestro ejemplo anterior, teníamos un servidor IMAP se ejecuta en la máquina S, y un lector de correo electrónico en una maquina H, y piensa que era para asegurar la conexión SSH utilizando IMAP.

IMAP utiliza el puerto TCP 143, lo que significa que un servidor IMAP escucha las conexiones en el puerto 143 del servidor. Para el túnel de conexión IMAP a través de SSH, usted necesita escoger un puerto local en la máquina de origen H (entre 1024 y 65535) y lo remitirá al puerto (S, 143). Suponga que usted escoge al azar el puerto local 2001. El siguiente comando crea el túnel:

```
$ ssh –L 2001:localhost:143 S
```

La opción `-L` especifica el Forwarding local. TCP en la máquina local con el cliente SSH. La opción es seguida por tres valores separados por dos puntos: un puerto local para escuchar en (2001), el nombre del equipo remoto o la dirección IP (S), y un puerto de destino (143).

El comando anterior se registra en S, ya que si sólo tienes que escribir ssh. Sin embargo, este período de sesiones SSH también ha remitido el puerto TCP 2001 en H para el puerto 143 en S, la transmisión se mantiene en efecto hasta que cierre la sesión de la reunión. Para hacer uso del túnel, el último paso es decirle a su lector de correo electrónico para utilizar el puerto redirigido. Normalmente, su programa de correo electrónico se conecta al puerto 143 en el servidor, es decir, la toma (S, 143). En su lugar, está configurado para conectarse al puerto 2001, sobre las máquinas de la casa H, es decir, localhost por el puerto 2001. Así que el camino de la conexión es el siguiente:

1. El lector de correo electrónico en la maquina H envía datos al puerto local 2001.
2. El cliente local SSH en H lee el puerto 2001, cifra los datos, y la envía a través de la conexión SSH al servidor SSH en S.
3. El servidor SSH en S descifra los datos y lo envía al servidor IMAP escuchando en el puerto 143 en S.
4. Los datos son enviados desde el servidor IMAP para la máquina H por el mismo proceso a la inversa.

El reenvío de puertos sólo se puede especificar cuando se crea una conexión SSH. No se puede añadir un reenvío a una conexión existente de SSH con cualquier implementación de SSH que conocemos, aunque no hay nada intrínseco en el protocolo SSH que se lo impediría, y que a veces sería una característica útil. En lugar de utilizar la opción-L para establecer una transmisión local, puede utilizar el LocalForward palabra clave en el archivo de configuración del cliente:

```
# SSH1, OpenSSH
LocalForward 2001 localhost:143
# SSH2 only
LocalForward "2001:localhost:143"
```

Tenga en cuenta las pequeñas diferencias sintácticas. En SSH1 y OpenSSH, hay dos argumentos: el número de puerto local y la toma de distancia expresada como host: puerto. En SSH2, la expresión es igual que en la línea de comandos, excepto que debe ir entre comillas dobles. Si se olvida de las comillas, ssh2 no alerta nada, pero no reenvía el puerto.

El ejemplo de una maquina cliente (H) y un servidor IMAP (S) se puede configurar de esta manera:

```
# SSH1, OpenSSH
Host local-forwarding-example
HostName S
LocalForward 2001 localhost:143
# Run on home machine H
$ ssh local-forwarding-example
```

Reenvio remote (Forwarding remoto). Un puerto transmite de forma remota como si fuera local, pero las instrucciones están invertidas. Esta vez, el cliente TCP es remoto, su servidor es local, y una conexión se inicia desde una máquina remota.

Continuando con nuestro ejemplo, supongamos que en lugar de estar registrado en la máquina servidor S para empezar, el servidor IMAP está en marcha. Ahora puede crear un túnel seguro para los clientes remotos para acceder al servidor IMAP en el puerto 143. Una vez más, se selecciona al azar un número de puerto que transmita (por ejemplo, 2001) y crear el túnel:

```
$ ssh –R 2001:localhost:143 H
```

La opción –R especifica el reenvío a distancia. Es seguido por tres valores, separados por dos puntos como antes, pero interpretado ligeramente diferente. El puerto remoto que se transmitirá (2001) está ahora en primer lugar, seguido por el nombre del equipo o la dirección IP (localhost) y el número de puerto (143). SSH ahora puede reenviar conexiones de (localhost, 143) a (H, 2001).

Una vez que este comando se ha ejecutado, un túnel seguro se ha construido desde el puerto 2001 en el equipo remoto H, para el puerto 143 en el equipo servidor S. Ahora, cualquier programa de H puede utilizar el túnel seguro mediante la conexión a (localhost, 2001). Como antes, el comando también se ejecuta una sesión de terminal SSH en la máquina remota H, al igual que ssh H lo hace.

Al igual que con el reenvío local, se puede establecer una transmisión a distancia utilizando una palabra clave en el archivo de configuración del cliente. La palabra clave RemoteForward es análoga a LocalForward, con las diferencias sintácticas entre las mismas SSH1 y SSH2:

```
# SSH1, OpenSSH
RemoteForward 2001 S:143
# SSH2 only
RemoteForward "2001:S:143"
Por ejemplo, la transmision anterior se define en un
formato SSH2 archivo de configuracion:
# SSH2 only
remote-forwarding-example:
Host H
RemoteForward "2001:S:143"
$ ssh2 remote-forwarding-example [11]
```

2.1.3 Herramientas Putty y OpenSSH.

PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Disponible originalmente sólo para Windows, ahora también está disponible en varias plataformas Unix, y se está desarrollando la versión para Mac OS clásico y Mac OS X. Otra gente ha contribuido con versiones no oficiales para otras plataformas, tales como Symbian para teléfonos móviles. Es software beta escrito y mantenido principalmente por Simon Tatham, open source y licenciado bajo la Licencia MIT. [12]

Algunas características de PuTTY son:

- El almacenamiento de hosts y preferencias para uso posterior.
- Control sobre la clave de cifrado SSH y la versión de protocolo.
- Clientes de línea de comandos SCP y SFTP, llamados "pscp" y "psftp" respectivamente.
- Control sobre el redireccionamiento de puertos con SSH, incluyendo manejo empotrado de reenvío X11.
- Completos emuladores de terminal xterm, VT102, y ECMA-48.
- Soporte IPv6.
- Soporte 3DES, AES, RC4, Blowfish, DES.
- Soporte de autenticación de clave pública.
- Soporte para conexiones de puerto serie local.

El nombre PuTTY proviene de las siglas Pu: Port unique TTY: terminal type. Su traducción al castellano sería: Puerto único para tipos de terminal

Historial de versiones.

Anteriormente a la versión 0.58, se hicieron tres releases consecutivos (0.55–0.57) para arreglar agujeros de seguridad significativos en versiones previas, algunos permitían comprometer al cliente incluso antes de que el servidor fuera autenticado.

A la versión 0.58 se le agregaron nuevas características, incluyendo soporte Unicode mejorado, para caracteres internacionales y lenguajes bidireccionales o de derecha a izquierda.

Después de casi un año desde el release previo, la versión 0.59 implementa nuevas características como la conexión a puertos serie, proxies locales, mejoras de velocidad SSH y SFTP, cambia el formato de documentación (para compatibilidad con Vista) y tiene varias correcciones de errores.

La versión 0.60 implementa 3 nuevas características y corrige algunos errores:1 Ctrl+Break ahora envía una señal de ruptura en el otro extremo

En Windows, ya no es necesario configurar líneas alto-numeradas tales como COM10; PuTTY hace esto automáticamente. Ahora se puede almacenar un nombre de host en las opciones por defecto.

Aplicaciones, las funciones principales están realizadas por los mismos ficheros PuTTY:

PuTTY - los clientes Telnet y SSH.
PSCP - un cliente SCP, copia de ficheros segura por línea de comandos.
PSFTP - un cliente SFTP, sesiones de transferencia de ficheros generales como en FTP.
PuTTYtel - un cliente de solo Telnet
Plink - una interfaz de línea de comandos al PuTTY back ends.
Pageant - un agente de autenticación SSH para PuTTY, PSCP y Plink.
PuTTYgen - una utilidad de generación de claves RSA y DSA.
pterm - un emulador de terminal X.
PuTTYtray, básicamente incorpora mejoras en el aspecto gráfico. Como un icono más colorido, la posibilidad de minimizarlo al tray, configurar la transparencia de la ventana, conversión de URLs en enlaces o poder mantener la ventana siempre encima, entre otros.

También cuenta con mejoras de funcionalidad como la reconexión automática al volver del modo suspendido, reconexión cuando hay fallos o el guardado de datos en ficheros para hacerlo potable en discos USB. [13]

OpenSSH es una versión LIBRE del paquete de herramientas de comunicación segura del protocolo SSH/SecSH para redes, una solución de seguridad que está ganando la confianza de un número cada vez mayor de usuarios de Internet. Muchos usuarios de telnet, rlogin, ftp y otros programas parecidos, no se dan cuenta que sus contraseñas se están transmitiendo sin cifrar a través de la red.

OpenSSH cifra todo el tráfico (incluidas las contraseñas) para eliminar de un modo efectivo las «escuchas», los secuestros de las conexiones y otros ataques a nivel de red. Además, OpenSSH ofrece amplias posibilidades para la creación de túneles seguros, aparte de una variedad de métodos de autenticación.

En el sistema operativo OpenBSD se encuentran integrados los programas ssh, que substituye a rlogin y telnet, scp, que substituye a rcp, y sftp que substituye a ftp. También están incluidos sshd, que es el programa servidor del paquete, y otras utilidades básicas como ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen y sftp-server. OpenSSH dispone de soporte para las versiones 1.3, 1.5, y 2.0 del protocolo SSH.

OpenSSH es un proyecto desarrollado principalmente por el Proyecto OpenBSD, y su primera integración en un sistema operativo fue en OpenBSD 2.6. Estos programas se desarrollan fuera de los EE.UU., usando código desarrollado en unos 10 países distintos. Este código es de libre utilización bajo la licencia BSD.

La gestión de la distribución de OpenSSH está dividida entre dos equipos. Un equipo sólo lleva a cabo el desarrollo basado en OpenBSD, y su objetivo es el de producir un código que sea tan limpio, simple y seguro como sea posible.

Estamos convencidos de que el sacrificio de la portabilidad en favor de la simplicidad permite un mejor control de calidad y facilita la revisión del código. El otro equipo toma la versión «pura» y la convierte en una versión portable, para que pueda funcionar en muchos otros sistemas operativos (estas versiones se conocen como «versiones p», y se distinguen por nombre.

Características
OpenSSH es una implementación libre del paquete de protocolos *SSH/SecSH* que provee cifrado para los servicios de red, como ingreso remoto (*"remote login"*) o transferencia de archivos remota, de un sistema de cifrado.

La lista siguiente enumera las características de Openssh:
Proyecto de Código Abierto
Licencia Libre.
Cifrado Fuerte (3DES, Blowfish, AES, Arcfour).
Reenvío por X11 (cifra el tráfico de X Window System).
Reenvío por Puertos (canales cifrados por protocolos de legado).
Autenticación Fuerte (Clave Pública, Contrasña de un sólo uso y Autenticación con Kerberos).
Reenvío por Agente (ingreso único).
Interoperabilidad (Conforme con las Regulaciones del Protocolo SSH 1.3, 1.5, y 2.0).
Soporte para cliente y servidor de SFTP en los protocolos SSH1 y SSH2.
Pases de Ticket de Kerberos y AFS.
Compresión de Datos.

Proyecto de Código Abierto
El código fuente de Openssh es de libre disponibilidad para todo aquel que desee obtenerlo en Internet. Se anima a todo el que desee reutilizar el código o hacer una auditoría sobre éste a que lo lleve a cabo. La revisión del código asegura que cualquier persona pueda encontrar y corregir errores. El resultado es un código seguro.
Licencia Libre
OpenSSH no se encuentra bajo ninguna licencia restrictiva. Se puede usar para cualquier propósito, y esto incluye su uso comercial. La licencia para OpenSSH se encuentra incluida en la distribución. Pensamos que el mundo estaría mejor si enrutadores, aparatos para redes, sistemas operativos, y el resto de dispositivos de red tuvieran ssh integrado en ellos.

Todos los componentes de naturaleza restrictiva (o sea, patentes, ver ssl) han sido eliminados del código fuente; los componentes bajo licencia o patentados se obtienen de bibliotecas externas (v.g. OpenSSL). El algoritmo de cifrado simétrico IDEA ya no se encuentra disponible, debido a que está patentado en muchos

países. En su lugar, recomendamos que use cualquiera de los otros algoritmos de cifrado disponibles (no creemos que se pueda justificar el uso de un algoritmo de cifrado simétrico patentado, cuando existen muchos otros libres).

Cifrado Fuerte
OpenSSH soporta 3DES, Blowfish, AES y Arcfour como algoritmos de cifrado. Todos ellos están libres de patentes. Triple DES es un algoritmo de cifrado muy conocido y que ha pasado la prueba del tiempo, que provee cifrado fuerte. Blowfish es un algoritmo de cifrado rápido de bloque inventado por Bruce Schneier, que pueden usarlo aquéllos que requieran un cifrado más rápido. AES es la Norma Avanzada de Cifrado (AES por sus siglas en inglés) de la Norma Federal de Procesamiento de Información de los Estados Unidos de América (FIPS por sus siglas en inglés) desarrollado para reemplazar a DES.

Es un algoritmo de cifrado rápido de bloque. Arcfour es un cifrador rápido de flujo. Se piensa que es compatible con RC4[TM], un cifrador privativo de RSA Security Inc. El cifrado empieza antes de la autenticación, y ninguna contraseña ni otro tipo de información se transmite sin cifrar. El cifrado también se utiliza como protección contra paquetes falsificados.

Reenvío por X11
El envío por X11 permite el cifrado del tráfico en entornos X windows remotos, de tal modo que nadie pueda fisgonear en sus xterm remotas o insertar órdenes dañinas. El programa activa DISPLAY automáticamente en el servidor, y envía cualquier conexión de X11 por un canal seguro. Información falsa sobre Xauthority se genera automáticamente y se envía a la máquina remota; el programa cliente en la máquina local examina las conexiones entrantes de X11 y reemplaza los datos de autorización falsos con datos reales (pero nunca le pasa a la máquina remota la información real).

Reenvío por Puertos
El envío por puertos permite enviar conexiones de TCP/IP a una máquina remota por un canal cifrado. Las aplicaciones típicas de Internet como POP se pueden asegurar de este modo.

Autenticación Fuerte
Una fuerte autentificación protege contra varios problemas de seguridad, como por ejemplo suplantación de IP (*IP spoofing*), rutas falsas (*fake roots*), y fisgoneo de DNS (*DNS spoofing*). Los métodos de autenticación son: *.rhosts* junto con huésped de autenticación basado en RSA, autenticación RSA pura, contraseñas para uso de una sola vez, y finalmente autenticación mediante Kerberos.

Reenvío por Agente
Un agente de autenticación que se encuentre en la estación de trabajo o el portátil de un usuario, se puede usar para contener las claves de autenticación de RSA o DSA. OpenSSH envía la conexión automáticamente al agente de autenticación por medio de cualquier conexión y de este modo no existe la

necesidad de guardar las claves de autenticación de RSA o DSA en ninguna máquina de la red (exceptuando la máquina del usuario).

Los protocolos de autenticación nunca revelan las claves; sólo se pueden usar para verificar que el agente del usuario tenga cierta clave. El agente podría hacer uso de una tarjeta inteligente para llevar a cabo todas las computaciones de autenticación.

Interoperabilidad
Las versiones de OpenSSH anteriores a la 2.0 contienen soporte para los protocolos SSH 1.3 y SSH 1.5, permitiendo de este modo la comunicación con la mayoría de implementaciones comerciales de SSH en Unix y Windows.

A partir de la versión 2.0 de OpenSSH, además del soporte para el protocolo SSH 1.3 y protocolo SSH 1.5, OpenSSH también dispone de soporte para el protocolo SSH 2.0. Este protocolo evita el uso del algoritmo RSA -- ya que cuando se inventó el protocolo 2.0, la patente sobre RSA todavía no existía -- y en su lugar usa los algoritmos libres DH y DSA. Por lo tanto, OpenSSH le ofrece lo mejor de los dos mundos. Puede ínter operar con ambos tipos de clientes y servidores de SSH.

Soporte para cliente y servidor en los protocolos SSH1 y SSH2
A partir de la versión 2.5.0, OpenSSH incluye soporte completo para SFTP; la orden que se usa para el cliente es sftp(1). El subsistema sftp-server(8) funciona de forma automática en los protocolos SSH1 y SSH2.

Pases de Tickets de Kerberos y AFS
OpenSSH también pasa tickets para Kerberos y AFS en la máquina remota. Un usuario puede por tanto acceder a todos sus servicios de Kerberos y AFS sin la necesidad de introducir una contraseña de nuevo.

Compresión de Datos
La compresión de datos antes del cifrado mejora los resultados en los enlaces con redes lentas. [14]

2.2 SSL (Secure Socket Layer)

Netscape desarrolló la primera versión de SSL en 1994. Está primera versión jamás fue implementada de forma pública. Tan sólo unos pocos meses después liberó una importante actualización que vino a llamarse SSL 2.0 y que si tuvo una implementación real a pesar de ir aquejada de importantes errores de diseño. En noviembre de 1995 Netscape publica la especificación para SSL 3.0 la cual, desde entonces, se ha convertido en el estándar 'de hecho' para comunicaciones seguras entre clientes y servidores en Internet.

El objetivo de Netscape era crear un canal de comunicación seguro entre un cliente y un servidor que fuese independiente del sistema operativo usado por ambos y que se beneficiara de forma dinámica y flexible de los nuevos adelantos en materia de cifrado a medida de que estos estuvieran disponibles. SSL fue diseñado como un protocolo seguro de propósito general y no teniendo en mente las necesidades específicas del comercio electrónico.

SSL trabaja sobre el protocolo TCP y por debajo de protocolos como HTTP, IMAP, LDAP, etc., y puede ser usado por todos ellos de forma transparente para el usuario. Opera entre la capa de transporte y la de sesión del modelo OSI (o entre la capa de transporte y la de aplicación del modelo TCP) y está formado, a su vez, por dos capas y cuatro componentes bien diferenciados.

2.2.1 Concepto.

SSL trabaja entre la capa de transporte y la de sesión del modelo OSI (o entre la capa de transporte y la de aplicación del modelo TCP) y está formado, a su vez, por dos capas la capa SSL Record Protocol y la Application Layer Protocol, además en esta ultima capa que se hace mención se forma por cuatro componentes bien diferenciados, el SSL Handshake Protocol, el SSL Change Chipre Spec Protocol, el SSL Alert Protocol y por ultimo Application Data Protocol como lo podemos observar en la Figura 2.5.

Ahora realizamos una breve explicación a modo de introducción para el protocolo de capa de aplicación ya que, en esta capa se realiza el proceso para establecer una sesión basada en el protocolo SSL y es el principal componente para asegurar un medio de transmisión seguro, estas características están explicadas en el RFC SSL 4346, para mayor detalle de este protocolo y queda como referencia.

El protocolo de registro (Record Protocol) se encarga de encapsular los datos de las capas superiores, construyendo un canal de comunicaciones entre los dos extremos objeto de la comunicación.

El verdadero corazón de SSL está en el protocolo de Handshake que es el encargado de intercambiar la clave que se utilizará para crear un canal seguro mediante un algoritmo eficiente de cifrado simétrico. También es responsabilidad de este protocolo coordinar los estados de ambos extremos de la transmisión.

El Change Cipher Spec Protocol está formado por un único mensaje consistente en un único byte de valor 1 y se utiliza para notificar un cambio en la estrategia de cifrado.

El protocolo de Alerta es el encargado de señalizar problemas y errores concernientes a la sesión SSL establecida.

El SSL Application Data Protocol se utiliza para la segunda función mencionado anteriormente (es decir, la transmisión segura de datos de la aplicación). Este protocolo es el caballo de batalla real de SSL, toma las capas superiores por lo general la capa de aplicación de datos y alimenta el Protocolo SSL Record protección criptográfica y la transmisión segura.

A grandes rasgos, podríamos decir que SSL trabaja de la siguiente forma: en primer lugar intercambiamos una clave de longitud suficiente mediante un algoritmo de cifrado asimétrico. Mediante esa clave establecemos un canal seguro utilizando para ello un algoritmo simétrico previamente negociado. A continuación, toma los mensajes a ser transmitidos, los fragmenta en bloques, los comprime, aplica un algoritmo hash para obtener un resumen (MAC) que es

concatenado a cada uno de los bloques comprimidos para asegurar la integridad de los mismos, realiza el cifrado y envía los resultados.

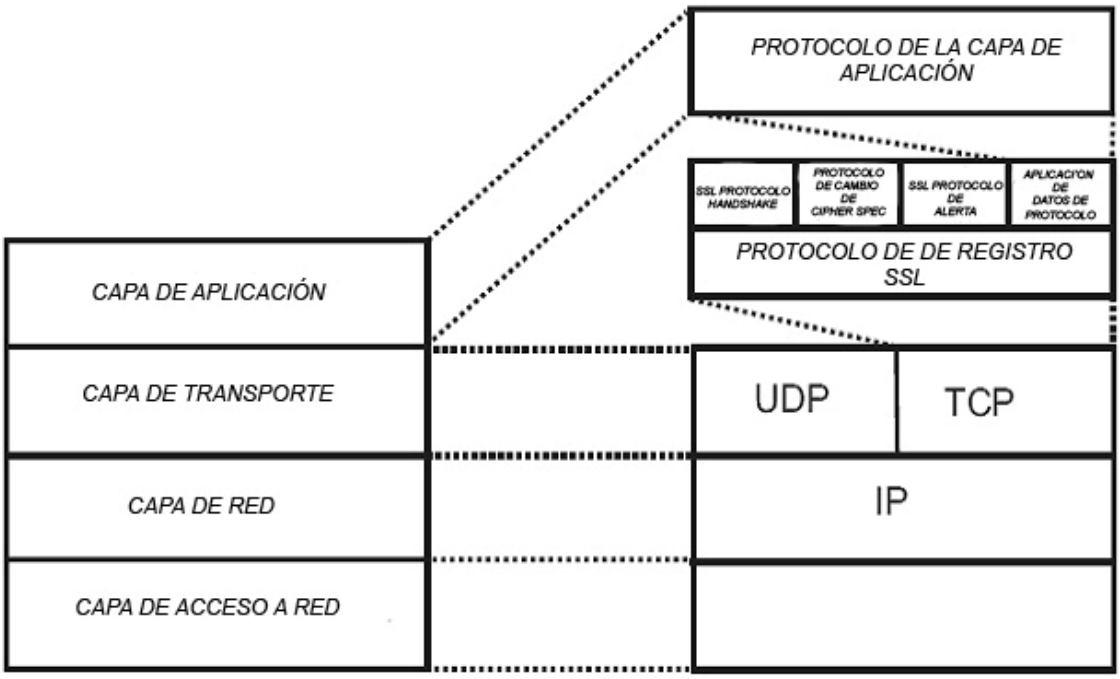

Figura 2.5 SSL con sus sub capas y sus sub protocolos.

El estado de todas estas operaciones son controladas mediante una máquina de control de estados. Una sesión SSL puede comprender múltiples conexiones. Adicionalmente, se pueden establecer múltiples sesiones SSL simultáneas. A continuación veremos todos estos procesos con un poco más de detalle. El protocolo de Handshake es el encargado de negociar los atributos de la sesión SSL que permitirán construir un canal seguro de comunicaciones.

En primer lugar el cliente envía un mensaje Client Hello al servidor el cual debe de responder con un mensaje similar de Server Hello. Estos mensajes son utilizados para dar a conocer ciertas características de ambos: versión del protocolo usado, algoritmos de cifrado conocidos y preferidos, longitudes máximas de clave que admite para cada uno de ellos, funciones hash y métodos de compresión a utilizar. En este momento, además, el servidor asigna un identificador a la sesión y se hace constar la fecha y hora de la misma.

El identificador de sesión es enviado al cliente en el mensaje de Server Hello. Si el servidor no respondiera con un mensaje de Server Hello o este no fuese valido o reconocible la sesión abortaría inmediatamente. Generalmente el servidor, es el segundo en contestar, elige los algoritmos más fuertes de entre los soportados por el cliente. Si no hay acuerdo en este punto se envía un mensaje de error y se aborta la sesión.

A continuación del mensaje de Server Hello, el servidor puede enviar su Certificado (típicamente un X.509) de forma que sea autenticado por el cliente y que, además, este reciba su clave pública. Si no es así, le envía al cliente su clave pública mediante un mensaje de Server Key Exchange (o también si ha enviado su Certificado y este es únicamente para firma y autenticación). Está claro que al menos uno de estos dos mensajes es necesario para establecer el canal seguro. Un último mensaje que puede enviar el servidor en esta fase de negociación es una solicitud de certificado al cliente.

Por último, la fase concluye con el envío, por parte del servidor, de un mensaje de Server Hello Done. Si el Servidor ha solicitado su certificado al cliente, este debe de responder con el o con un mensaje de alerta indicando que no lo posee. A continuación se envía un mensaje de Client Key Exchange donde el cliente envía al servidor, cifrada mediante la clave pública de este, la clave maestra, un número aleatorio generado por el y que actuará como clave del algoritmo simétrico acordado para el intercambio de datos.

Si el cliente ha enviado un certificado y este tiene capacidades de firma, enviará adicionalmente un mensaje de Certificate Verify firmado digitalmente con objeto de que el servidor pueda verificar que la firma es válida. En este punto el cliente da por concluida la fase mediante un mensaje de Change Cipher Spec seguido, inmediatamente, de un mensaje de Finished que ya va cifrado mediante los algoritmos y claves recién negociados.

En respuesta, el servidor envía su propio mensaje de Change Cipher Spec y, a continuación, su mensaje de Finished cifrado con los parámetros negociados. En este momento finaliza la fase de Handshake y cliente y servidor pueden intercambiar datos libremente. Podemos ver un esquema de este intercambio de mensajes en la siguiente figura:

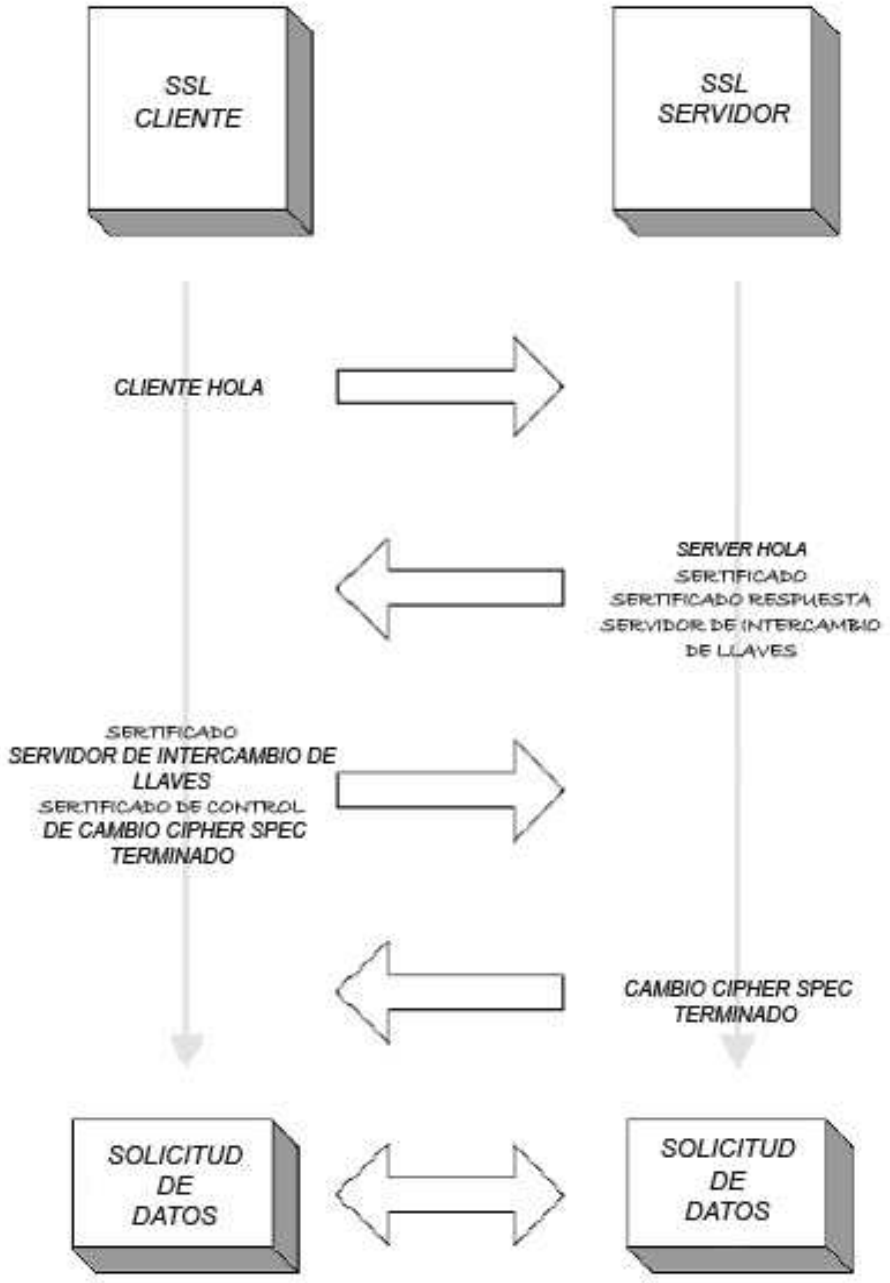

Figura 2.6 Handshake de cliente y servidor .

Durante la transmisión de datos los mensajes son fragmentados y comprimidos por el protocolo de registro antes de su envío y descomprimidos y reconstruidos por el mismo protocolo al otro extremo de la comunicación. El algoritmo de compresión utilizado es característico de la sesión y se negocia, como hemos visto, en la fase de Handshake.

SSL puede establecer múltiples conexiones dentro de una misma sesión o reanudar una sesión previamente interrumpida. En ambos casos el intercambio de mensajes de la fase Handshake es mucho más reducido como veremos a continuación. El cliente envía un mensaje de Client Hello usando el identificador de la sesión previamente negociada. El servidor verifica si ese identificador es válido y en caso afirmativo devuelve un mensaje de Server Hello usando el mismo identificador de sesión.

Acto seguido, el server envía al cliente un mensaje de Change Cipher Spec y a continuación un mensaje de Finished cifrado ya con los parámetros de la sesión reanudada. El cliente responde con sus propios mensajes de Change Cipher Spec y Finished y seguidamente comienzan a intercambiar datos. Podemos ver este proceso en la siguiente imagen:

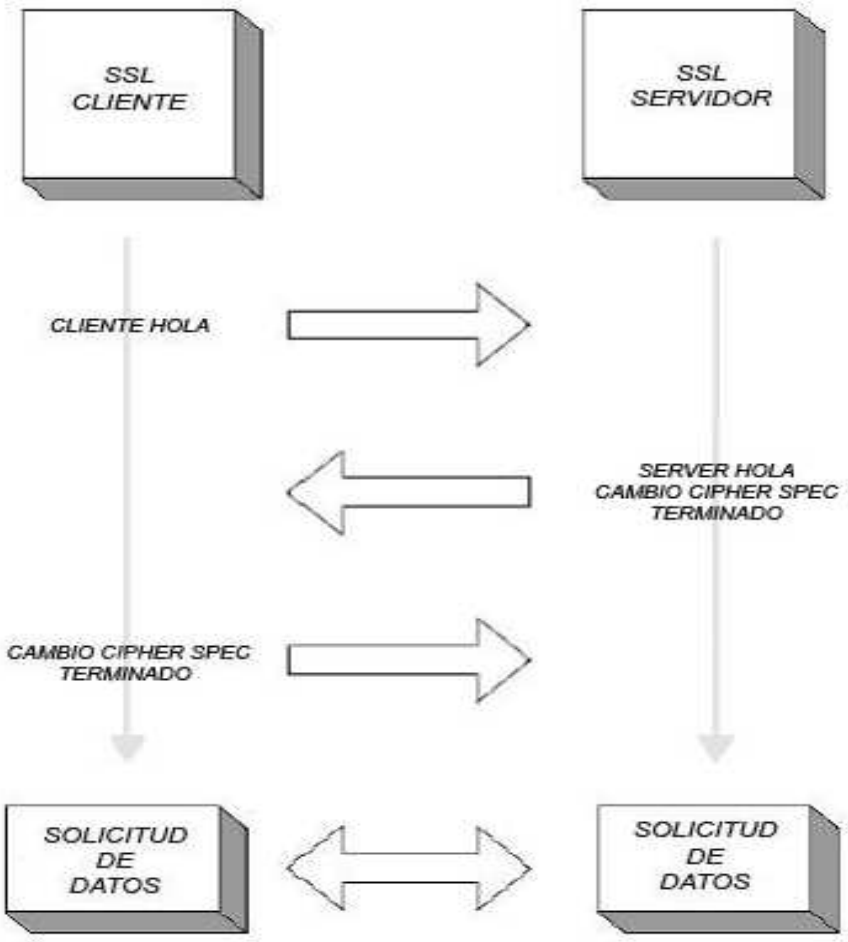

Figura 2.7 Change Cipher Spec de cliente y servidor.

Como se mencionó anteriormente, los protocolos SSL comprenden de SSL Record Protocol, the SSL Handshake Protocol, SSL Change Cipher Spec Protocol, the SSL Alert protocol y SSL Application Data Protocol. Enunciaremos en este orden estos protocolos. [15]

2.2.2 Características.

SSL Record Protocol. Como ya mencionamos SSL Record Protocol se utiliza para el encapsulamiento de datos de protocolo higherlayer, y que por lo tanto, fragmenta los datos en pedazos manejables (llamados fragmentos), y procesos para cada uno de los fragmentos por separado. Más concretamente, cada fragmento es opcionalmente comprimido y protegido criptográficamente según el método de compresión y cifrado de especificaciones del estado de la sesión SSL y los parámetros criptográficos del estado de la conexión SSL. El resultado representa el fragmento del registro SSL envía al destinatario. [16]

SSL Handshake Protocol. El SSL Handshake Protocolo está envuelto en la parte superior del Protocolo SSL Record. Permite a un cliente y el servidor autenticar mutuamente y negociar temas como conjuntos de cifrado y métodos de compresión. Los mensajes que se escriben entre corchetes son opcionales o dependientes de la situación, lo que significa que no siempre se envían.

Tenga en cuenta que ChangeCipherSpec no es en realidad un mensaje de Protocolo SSL Handshake, pero representa un protocolo SSL y por lo tanto un tipo de contenido de su cuenta, se escribe con un valor de un byte (es decir, un número decimal entre 0 y 255). [17]

SSL Change Cipher Spec Protocol. Consiste en un único mensaje, que encripta y comprime bajo las *current cipherspec.* Este mensaje tanto puede ser enviado por el cliente como el servidor, para notificar al receptor de que los siguientes registros se basaran en las llaves y las *cipherspec.*

Cuando se recibe este mensaje para cambiar las especificaciones de cifrado se copia en el estado de lectura pendiente al estado de lectura que se esta utilizando, igualmente sucede en el caso de quien lo escribe (pendiente-corriente). El cliente envia este mensaje seguido del intercambio de llaves y del mensaje de verificación de certificado y el servidor después de procesarlo envía el mensaje de intercambio de llaves que ha recibido del cliente. [18]

SSL Alert Protocol. Son mensajes de errores fatales que hacen que la conexión se acabe, tales como: mensaje inesperado. Incorrecto MAC, fallo de descompresion, certificado revocado, parámetros ilegales, etc…[19]

Application Data Protocol. Los mensajes son llevador a la capa de registro donde son fragmentados y comprimidos y encriptados en el estado de conexión. Estos mensajes son de información transparente para la capa de registro. [20]

2.2.3 Herramienta Stunnel.

Stunnel es un programa de computadora libre multi-plataforma, utilizado para la creación de túneles TLS/SSL. Stunnel puede ser utilizado para proveer conexiones cifradas seguras para clientes o servidores que no utilizan TLS o SSL de forma nativa. Corre en una gran variedad de sistemas operativos, incluyendo a la mayoría de los basados en el sistema operativo Unix y la familia Windows.

Su funcionamiento se basa en una biblioteca independiente como puede ser OpenSSL o SSLeay para implementar el protocolo TLS o SSL de capas inferiores. Stunnel utiliza criptografía de clave pública con el certificado digital X.509 para asegurar la conexión SSL. Los clientes se pueden autentificar de manera opcional a través de un certificado digital.

Si es conectado a una libwrap, se puede configurar también para que funcione como un servicio de firewall-proxy. Stunnel es mantenido por Michal Trojnara y Brian Hatch. Liberado bajo los términos de la GNU General Public License. stunnel es un programa muy útil que nos permite utilizar conexiones SSL con clientes y/o servidores que no soportan este protocolo.

Algunos ejemplos de uso son:

Utilizar el modo de conexión SSL entre un cliente que tenga esa opción y un servidor POP o IMAP que no lo tenga.
Lo mismo pero al revés (el servidor soporte SSL y el cliente no).
Cifrar cualquier conexión TCP entre dos ordenadores, permitiendo opcionalmente el control de acceso al servidor desde determinados clientes.

Certificados. Cuando stunnel funciona en modo servidor, debe presentar un certificado al cliente, como todo servidor SSL. La distribución de stunnel trae uno, pero es conveniente generar uno propio, usando OpenSSL por ejemplo. La clave privada debe residir en un fichero no cifrado para que el programa pueda acceder a ella sin tener que pedir clave. Lo mismo cabe decir si se van a usar certificados en modo cliente, para su autentificación.

Esto es un problema importante de seguridad, sobre todo en el caso de un PC, donde cualquier persona con acceso al mismo podrá acceder a la clave privada. Estaría bien que el programa soportara tenerla cifrada y contemplara formas de acceder a la misma al arrancar, como hace mod_ssl (como pedir la frase de paso al usuario, obtenerla mediante un programa o variables de entorno, etc.).

La autentificación de cliente puede imponerse ejecutando el servidor con la opción **-v**, cuyo argumento indica el nivel de autentificación exigido. Los valores que puede tomar son:

El cliente puede presentar o no un certificado. Si lo hace, comprobar que es válido.
El cliente debe presentar un certificado, que es verificado por el servidor. Si no es válido no acepta la conexión.
El cliente debe presentar un certificado, el cual debe ser figurar en una lista de certificados válidos almacenados en el servidor.

En las versiones previas de Stunnel (versión 4) aceptaba todas las configuraciones desde la línea de comandos. En la versión en curso (versión 4 y posteriores), sin embargo, Stunnel utiliza un archivo de configuración, stunnel.conf. De hecho, la localización de este archivo de configuración es lo único que se puede especificar con los parámetros desde la línea de comandos.

Stunnel utiliza solo un programa binario stunnel, que puede ejecutarse en dos modos: cliente y servidor. Funcionan de forma similar, excepto por una diferencia principal: en modo cliente, stunnel escucha las conexiones descifradas (por ejemplo, en la maquina local) y las reenvía a través de una conexión cifrada SSL a una maquina remota que ejecuta stunnel; en modo servidor, stunnel escucha las conexiones cifradas SSL (por ejemplo, de los procesos remotos stunnel) y después descifra y reenvía estas sesiones a un proceso local.

Los parámetros de stunnel.conf utilizados tanto para la configuración del cliente como la del servidor son por tanto similares; difieren en su utilización.Ahora se presentan los parámetros utilizados en los archivos stunnel.conf utilizados en la configuración de los archivos para el cliente y para el servidor.

Cliente = yes | no: El parámetro –c le indica a stunnel que se ejecute en modo cliente e interprete las opciones relacionadas acorde a este (por ejemplo, -d, -r).Sin este parámetro actúa como demonio únicamente.

Cert=/directorio/para/certificado.pem: Esta opción especifica el directorio completo del certificado de host. Es necesario que se encuentre en modo cliente solo cuando se vaya a presentar un certificado cliente a los servidores donde se conectara, pero un certificado siempre es necesario en modo servidor.

[nombre-de-servicio]: Esta etiqueta contenida entre corchetes, indica el principio de la definición de un nombre de servicio a pasar por stunnel en las llamadas a la librería libwrap(es decir, para comprobar con las entradas del archivo /etc/hosts.allow). Todos los parámetros que s encuentren por encima de la primera definición de servicio son aplicados de forma global. La definición de un servicio finaliza cuando se encuentre el comienzo de otro o finaliza el archivo.

Accept [IP-de-host:]puerto-de-demonio: Este parámetro especifica en que dirección IP y puerto escuchara stunnel. IP-de-host, una dirección IP local o nombre de host, especifica la dirección donde se desea que stunnel escuche (por ejemplo, especificando 127.0.0.1 restringe el uso del túnel a los usuarios locales). El parámetro puerto-de-demonio puede ser un puerto TCP numérico o un nombre de servicio existente dentro de /etc/services. En modo servidor, esta opción se utiliza para especificar el puerto donde se escuchan los paquetes de texto legible que van a ser tunelizados.

Connect[IP-remota:]puerto-remoto: El parámetro connect especifica a que puerto de stunnel va a enviar los paquetes. En modo servidor, seria el puerto TCP al que se podrían enviar los paquetes recibidos por el puerto accept (después de descifrarlos). En modo cliente, indica el puerto donde el sistema remoto (especificado en IP-remota, que puede ser una dirección IP o nombre de host) esta escuchando las conexiones del túnel. Puesto que IP-remota por defecto es localhost (127.0.0.1), se puede omitir esta parte en los servidores Stunnel.

Se tiene que considerar que se puede utilizar el parametro accept para limitar en que interfaz Stunnel acepta las conexiones. Stunnel no es la unica aplicación capacitada para establecer una conexión a un demonio Stunnel.Por ejemplo, es posible ejecutar Stunnel en un servidor POP3 escuchando en el puerto estándar pop3s (TCP 995) y reenviar a un demonio de correo local POP3, como Outlook Express y Eudora en sistemas que no ejecuten Stunnel.

CAPITULO III. DESARROLLO PRÁCTICO.

3.1 Implementación de túneles para el acceso remoto a un servidor de datos MySQL.

Los datos contenidos en una base de datos pueden llegar a ser un activo muy valorado para una entidad. Es por esta razón que se deben de tener mecanismos para asegurar en lo máximo la integridad de los datos cuando son solicitados, y si esta solicitud es realizada de forma remota, entonces se enfrenta a una situación, que eleva el riesgo de ver comprometida la integridad de los datos.

La información que tiene que viajar por un medio en el cual, puede llegar a ser interceptado; ve comprometida su integridad, porque no se puede garantizar su integridad y la disponibilidad, premisas fundamentales en la seguridad de la información. (Figura 3.1)

El problema que se plantea es simple, cuando realizamos una solicitud de forma remota a una base de datos en este caso MySQL, la respuesta que envía el servidor de datos viaja en texto plano, esto lo verificamos al interceptar la respuesta, por lo tanto, podemos ver la información, e incluso alterarla al aplicar un ataque, como puede ser hombre de en medio. Además la configuración del MySQL permite el acceso mediante una IP, o por segmento de red, con lo cual puede sufrir de un ataque de negación de servicio.

Para un sistema informático, esta situación carece de medidas de seguridad, por lo cual es necesario, proponer canales de comunicación confiables, en los cuales, la información si es interceptada en algún punto durante su viaje, esta no sea fácil de interpretar, esta característica se puede obtener, al cifrar la información; esta viaja de tal forma que al interceptarla se vea texto, que no tiene una interpretación gramatical lógica.

De esta forma se puede asegurar que la integridad de los datos se mantiene y asegurar su disponibilidad, para cumplir con dos objetivos de la seguridad de la información, la cual es la integridad y la disponibilidad de la información.

Para este proyecto, se realizo una conexión entro dos equipos, que se encuentran en redes locales distintas (Figura 3.2), estas dos redes están comunicadas mediante un enlace, que se establece en la red de redes, la Internet, entonces cada vez que se realice una petición del equipo cliente hacia el equipo servidor mediante IP y puerto, esta petición viaja a través de Internet en texto plano, si se intercepta la información se puede conocer la IP del servidor, el puerto del servicio y sobre todo la información.

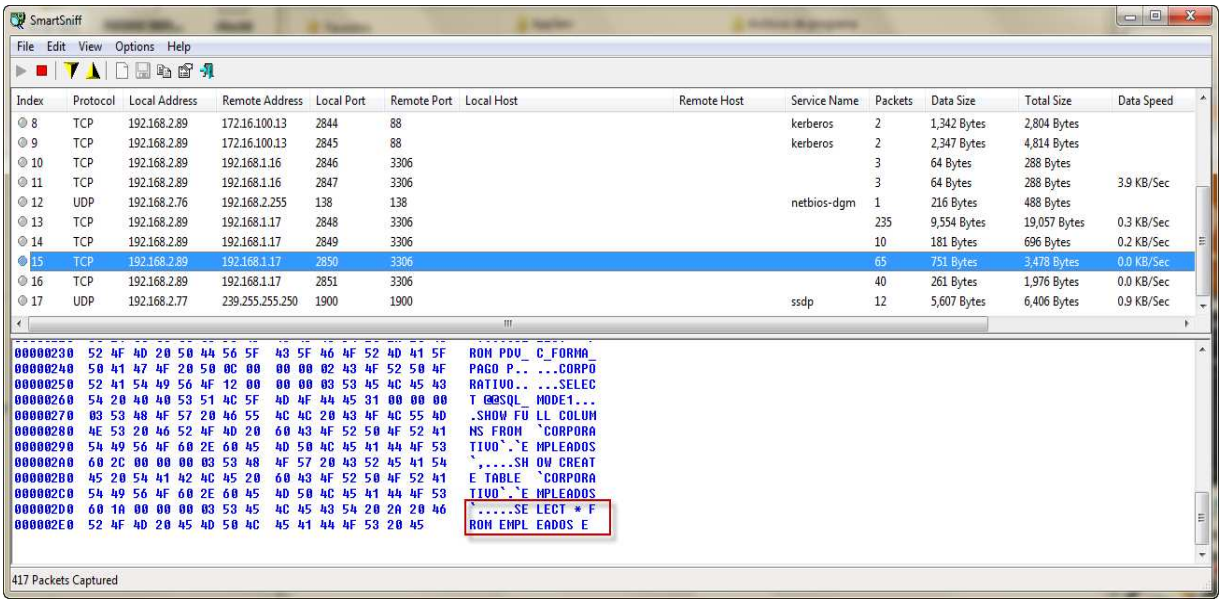

Figura 3.1 Captura de datos de una comunicación MySQL.

Con las características anteriores, pueden ocurrir ataques que comprometen la integridad de la información, suplantación de identidad mediante un ataque de tipo hombre de en medio y de negación de servicio al descubrir la IP del servidor y el puerto. Por lo cual se propone establecer un túnel entre el cliente y el servidor, a través del puerto perteneciente al servicio de MySQL por default el 3306 o cualquier otro establecido en su configuración.

El túnel se establece para comunicar el servicio de MySQL, por el puerto 3306. Esto significa que el túnel re direcciona el servicio, en la configuración por default de la instalación del MySQL el puerto de comunicación es el 3306 y la conexión la realizamos como un localhost, cuando se realiza una conexión remota, se realiza indicando la IP del servidor y el puerto, esto significa que la configuración del MySQL debe aceptar conexiones remotas.

Hasta este punto se observa que se deben se proponer políticas de seguridad, una de estas tiene que ver con el acceso, como es el acceso remoto, se puede restringir por IP o por segmento de red, existe la configuración que acepte todo tipo de conexión y solo restringir el acceso a las bases, ya que la configuración de usuarios también debe tener permisos de conexión remota. Además también implica que mediante esta configuración toda solicitud y respuesta viaje en texto plano. (Figura 3.1)

Por lo tanto se debe de especificar políticas de acceso remoto de usuarios a los equipos, además de mecanismos para que los datos no viajen en texto plano. La implementación de los túneles tienen como objetivo establecer características de seguridad de acceso a los usuarios de forma remota a los equipos, además proporcionan la característica de cifrar la información para que viaje por la red, esto agrega sin dudad alguna seguridad.

Los túneles de establecieron utilizando los protocolos SSH v2.0 y el SSL v2.0, la razón principal, permiten el tunneling y además cifran la información, trabajan con TCP; la conexión para este proyecto se realizo mediante identificación de usuario y contraseña, pero permiten el uso de certificados digitales, para evitar la intervención del usuario.

Su funcionamiento es simple, la comunicación se establece utilizando PortForwarding primero local y luego remoto es decir, redireccionando primero a un puerto local y a continuación a un puerto remoto. Esto significa que en el equipo cliente la conexión se realiza de forma local (localhost) al puerto 3306, ¿cómo es posible esto, si no está instalado el MySQL en el equipo cliente? La respuesta tiene que ver con el re direccionamiento del servicio, es decir, conectar el puerto 3306 del equipo cliente con el puerto 3306 del equipo servidor que es, en donde está instalado el MySQL a través de PortForwarding remoto, esta característica es posible utilizarla con los dos protocolos el SSH y el SSL.

Para establecer el túnel, primero se identifica a los puertos. Ahora su funcionamiento básico es, cuando se hace una petición al MySQL, esta petición se realiza a través del puerto 3306, que esta redireccionado al puerto 22 del putty para el caso del protocolo SSH.

Una vez que la información fue re direccionada inician los mecanismos de autenticación y cifrado, una vez establecida la sesión, se cifra la información para que viajen por el túnel, cuando llega a el equipo servidor, la información es descifrada y re enviada al puerto 3306 para que el servicio realice el procesamiento de la solicitud. De igual forma la respuesta de la solicitud es re direccionada al puerto, en donde será cifrada y enviada al equipo cliente, esta secuencia se repite para cada solicitud, solo el establecimiento de la sesión de realiza una única vez.

Las herramientas utilizadas son putty y stunnel para el equipo cliente con un sistema operativo Windows XP, para el equipo servidor openSSH y Stunnel con sistema operativo Ubuntu 10.10 para la configuración de los túneles, y el MySQL Query Browser para realizar las peticiones desde el cliente. Se utilizo SmartSniff para capturar paquetes de esta comunicación y verificar que estén cifrados. (Figura 3.2)

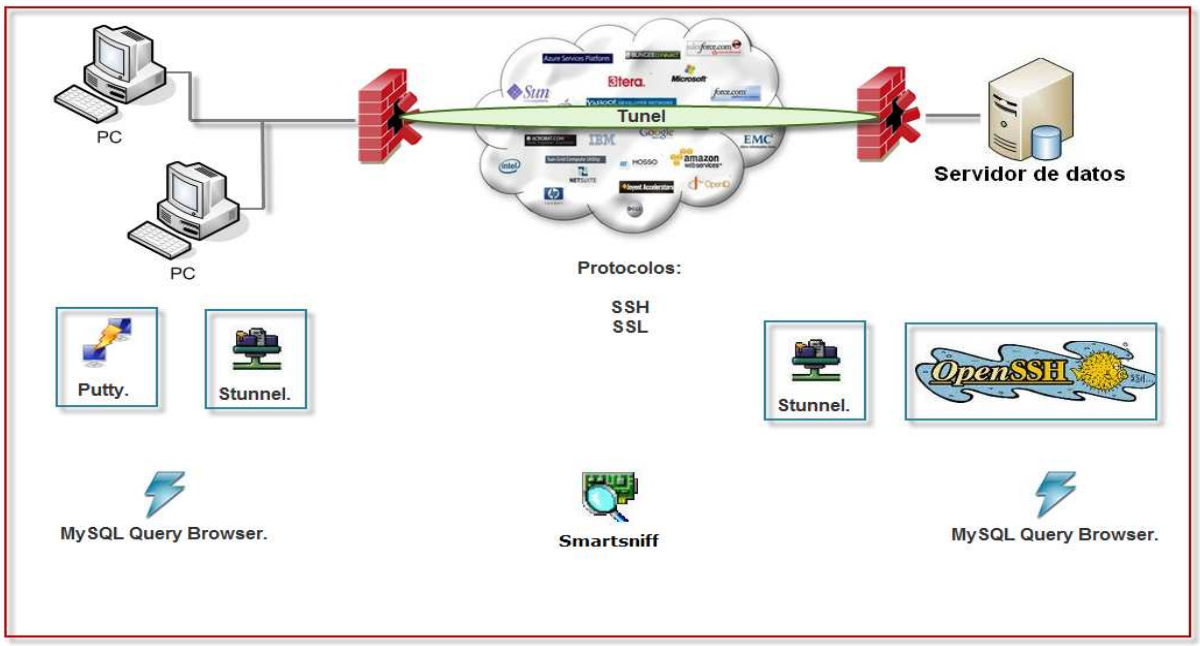

Figura 3.2 Escenario de prueba.

Al tener los túneles se realizaron pruebas de petición al servidor de datos, para capturar los datos en cada tunel y sin túnel con el fin de examinar la información. De esta forma se establece el escenario de pruebas para la obtencion de datos que se van a examinar en la parte final de este documento.

Las características de los equipos del escenario son:

Equipo Servidor:

- Sistema operativo Ubuntu 10.10
- openSSH.
- Stunnel4.
- MySQL.
- Una cuenta de acceso al servidor. (Usuario y Contraseña)
- Una cuenta de acceso al servidor de datos MySQL. (Usuario y Contraseña)

Equipo Cliente.

- Sistema Operativo Windows XP.
- Putty.
- Stunnel.
- MySQL Query Browser.

3.2 Túnel SSH.

La configuración para establecer el túnel utilizando el protocolo SSH consta de dos secciones una del lado del servidor y otra del lado del cliente, que a continuación se mencionan.

Configuración en el Servidor.

En el equipo servidor se instala el software openSSH, que funciona utilizando el protocolo SSH, en su configuración utiliza el puerto 22 para establecer comunicación. En el lado del cliente se utiliza la herramienta Putty. Para la instalación de openSSH, se utiliza la configuración que trae por default, para este caso práctico es suficiente, se puede configurar el openSSH para establecer mecanismos de seguridad, que quedan fuera del alcance de este trabajo.

OpenSSH es una versión libre del protocolo Secure Shell (SSH) que es una familia de herramientas para control remoto o transferencia de archivos entre equipos. Las herramientas utilizadas tradicionalmente para realizar estas funciones, eran el telnet o el rcp, que son inseguras y transmiten la contraseña de los usuarios en texto plano cuando son usadas. OpenSSH proporciona un demonio y unos clientes para facilitar un control remoto seguro y cifrado, así como operaciones de transferencia de archivos, reemplazando de forma efectiva las herramientas heredadas.

El componente servidor de OpenSSH, sshd, esta a la espera de conexiones de clientes desde cualquiera de las herramientas cliente. Cuando aparece una petición de conexión, sshd establece la conexión correcta dependiendo del tipo de herramienta cliente que está conectándose. Por ejemplo, si el equipo remoto se está conectando con la aplicación cliente ssh, el servidor OpenSSH establecerá una sesión de control remoto tras la autenticación. Si el usuario remoto se conecta al servidor OpenSSH con scp, el demonio del servidor OpenSSH iniciará una copia segura de archivos entre el servidor y el cliente tras la autenticación. OpenSSH puede usar muchos métodos de autenticación, incluyendo contraseñas planas, claves públicas y tickets de Kerberos

Desde una terminal se debe de ejecutar el siguiente comando:

```
$ apt-get install openssh-server
```

Ahora se realiza una conexión de prueba al mismo equipo de la siguiente forma:

```
$ ssh 127.0.0.1
```

Solicita que se acepta la llave de autenticación la primera vez, la aceptamos, y a continuación solicita la contraseña de usuario, para validar el acceso. De este modo se ha probado el funcionamiento del servidor open SSH.

Configuración en el Cliente.

Para establecer el túnel mediante el protocolo SSH, ejecutar el putty.exe e iniciar la configuración del túnel, las características de este túnel, son: establece el túnel de puerto a puerto, es decir conectamos el puerto 22 del equipo servidor de datos con el puerto 22 del equipo cliente, a su vez el puerto 22 esta redireccionado al servicio de MySQL, es decir al puerto 3306 en ambos equipos; como requisito, la maquina cliente debe de tener disponible este puerto, ya que la conexión de la aplicación, en este caso el MySQL Query con el cual se realiza las consultas, se conecta como localhost es decir, escucha al puerto 3306.

La instalación del putty es simple, solo es necesario obtener el ejecutable de la pagina oficial, y guardarlo en un directorio para después ejecutarlo e inmediatamente iniciar una conexión hacia un servidor OpenSSH. En la primera ventana ingresar la IP del servidor SSH, se debe especificar el puerto de configuración del OpenSSH e iniciar la conexión, la cual solicita usuario y contraseña una vez aceptada la llave de host para iniciar la sesión.

Para iniciar la configuración del túnel ejecutar el putty.exe en la pantalla que que se ve a continuación (Figura 3.3), especificar en el Host Name la dirección IP del equipo servidor de datos, como es una conexión SSH se realiza a través del puerto 22, si se ha configurado el SSH en otro puerto, entonces es necesario especificar el puerto. En el apartado de Saved Sessions, escribir un nombre para guardar la sesión, esto es útil porque cada vez que se requiera establecer el túnel solo se selecciona la sesión.

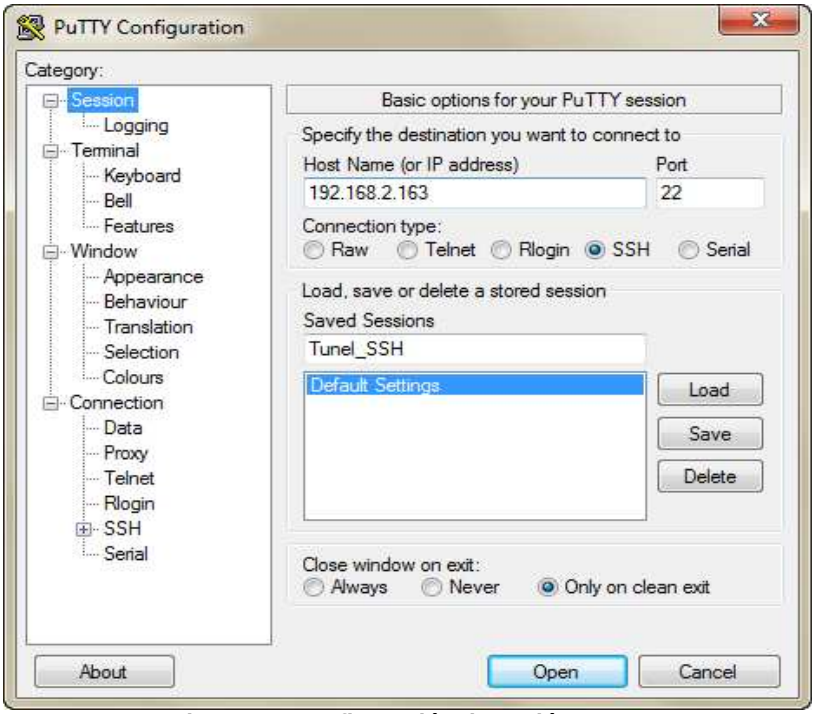

Figura 3.3 Configuración de sesión putty.

También es útil para automatizar el proceso, ya que se puede ejecutar desde un ms-dos y como parámetro el nombre de la sesión, es útil a la hora de establecer un proceso automatizado. Por el momento solo se especifica un nombre para la sesión, pero no se que guarda nada aun, se debe de esperar para tener todos los parámetros correctos porque después ya no es posible modificarlos, en caso de equivocación se debe borrar la entrada y nuevamente configurarla.

Dentro del apartado de Connection, seleccionar la opción Data, ahora se debe especificar el nombre de usuario, que tiene acceso al servidor linux, escribir el nombre en el campo Auto-login-username, de esta manera al iniciar sesión, lo hace con el usuario que se la ha indicado. (Figura 3.4)

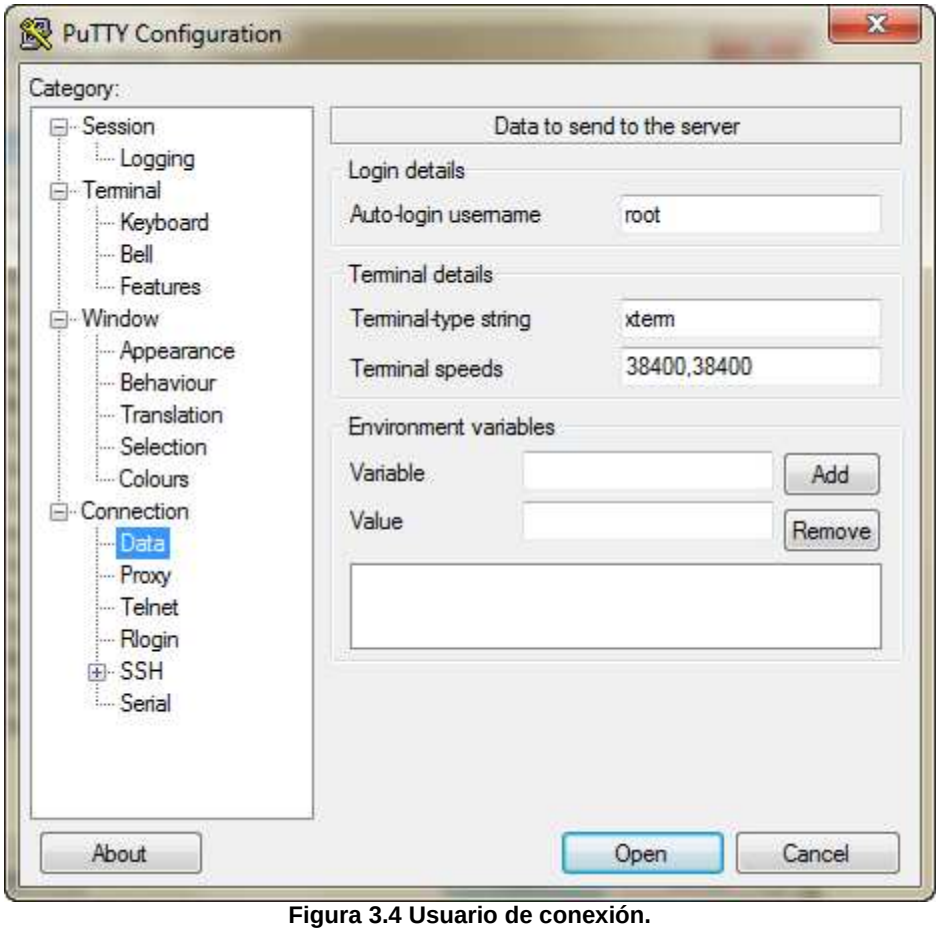

Figura 3.4 Usuario de conexión.

Dentro del apartado SSH de Conecction, se inicia la configuración del túnel, primero dentro de protocol options; habilitar la opción de no permitir que se inicie una consola, esto es debido a que solo es necesario el túnel, ya que no se van a ejecutar comandos remotos. Seleccionar la versión del protocolo SSH, en este caso, es la versión 2. (Figura 3.5)

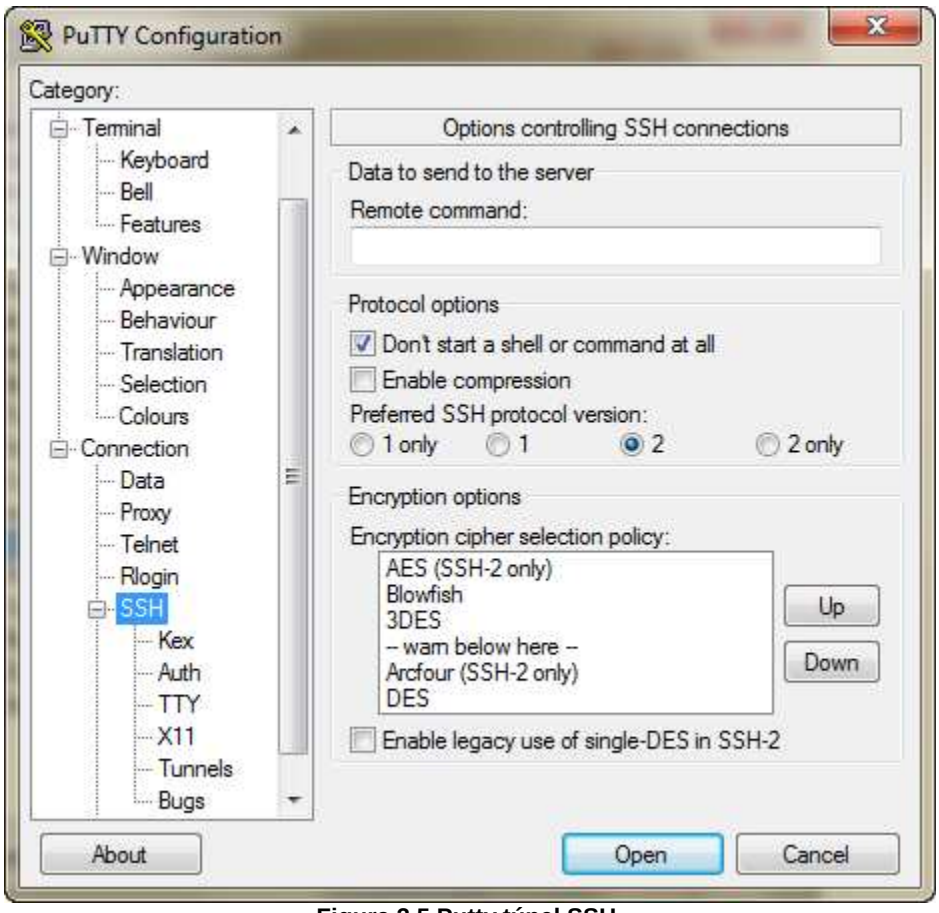

Figura 3.5 Putty túnel SSH.

A continuación en la configuración del servicio que va a trabajar utilizando el protocolo SSH, En la opción Tunnels, dentro de Source port ingresar el número de puerto que se va re direccionar en el equipo cliente; a continuación especificar en Destination ingresar la dirección IP del servidor asi como el puerto que se va a redireccionar. (Figura 3.6)

En esta sección se indica el source port, en este caso el 3306 se va a redirecionar al puerto 3306 de la IP X.X.X.X, esto significa que cada solicitud que realice el cliente, primero se hace al puerto 3306 que esta conectado al puerto 22, una vez llega a este puerto, se conecta al puerto 22 del equipo servidor, una vez que se encuentra en el puerto 22 del equipo servidor, este envía la solicitud al puerto 3306, para que el servicio reciba la solicitud, la procese y regrese una respuesta que viaja de igual forma.

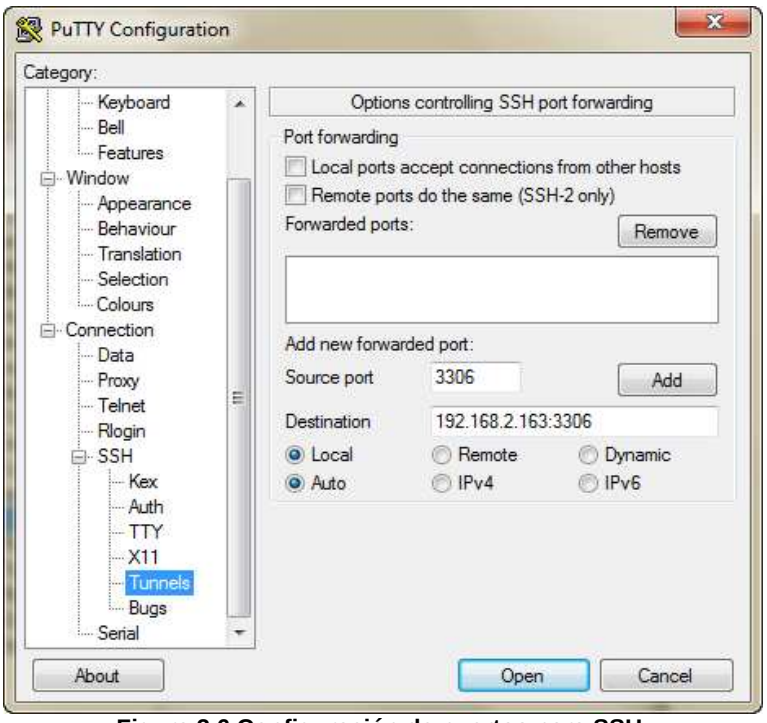

Figura 3.6 Configuración de puertos para SSH.

Una vez que se han ingresado los datos, se verifica que las opciones local y auto estén activadas, esto indica que la conexión se va a realizar solo en el equipo local, y la de auto es para que seleccione la versión de IP, una vez verificado los datos se pulsa la opción de agregar. (Figura 3.7)

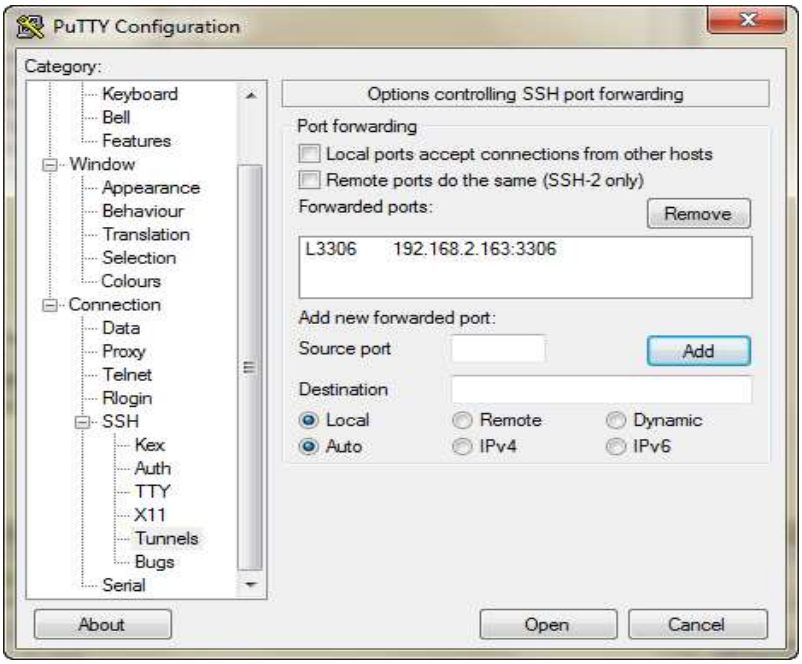

Figura 3.7 Establecimiento de puertos para el túnel.

Con el paso anterior se ha finalizado la configuración del túnel, ahora solo resta, regresar a la ventana inicial, la de sesión para guardar la configuración bajo un nombre asignado, y de esta forma de concluye la configuración del túnel. (Figura 3.8)

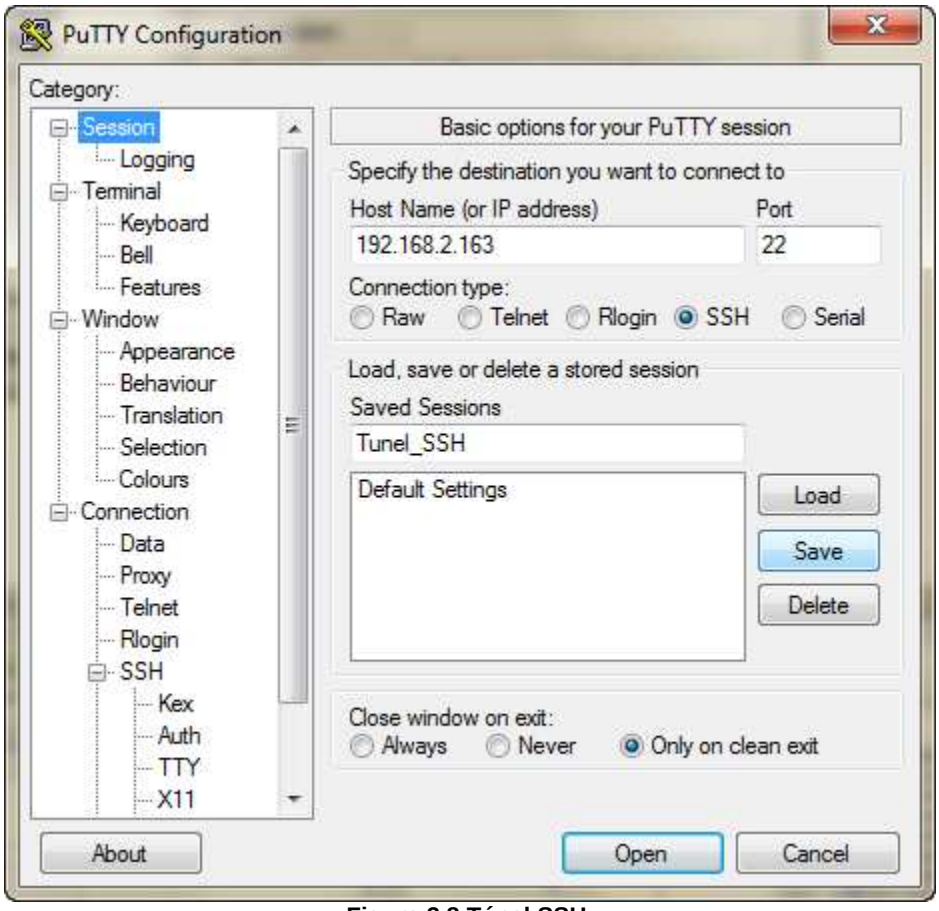

Figura 3.8 Túnel SSH.

Ahora para iniciar la sesión del putty para establecer el túnel, se tiene que seleccionar el nombre de la sesión y pulsar abrir. (Figura 3.9)

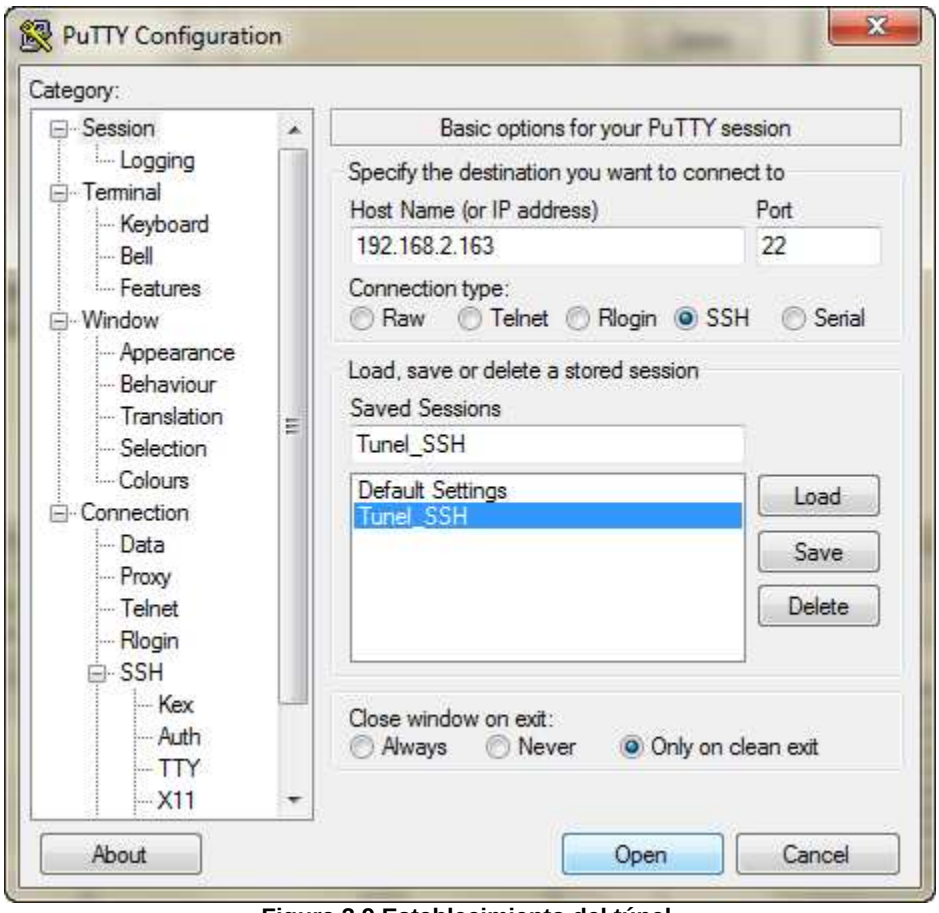

Figura 3.9 Establecimiento del túnel.

Ahora al iniciar la sesión, primero verifica una llave de sesión la cual envía el servidor al cliente, se debe aceptar la llave ya que através de esta se inicia la comunicación con el servidor de forma cifrada, en este punto aun no se ha establecido la sesión. (Figura 3.10)

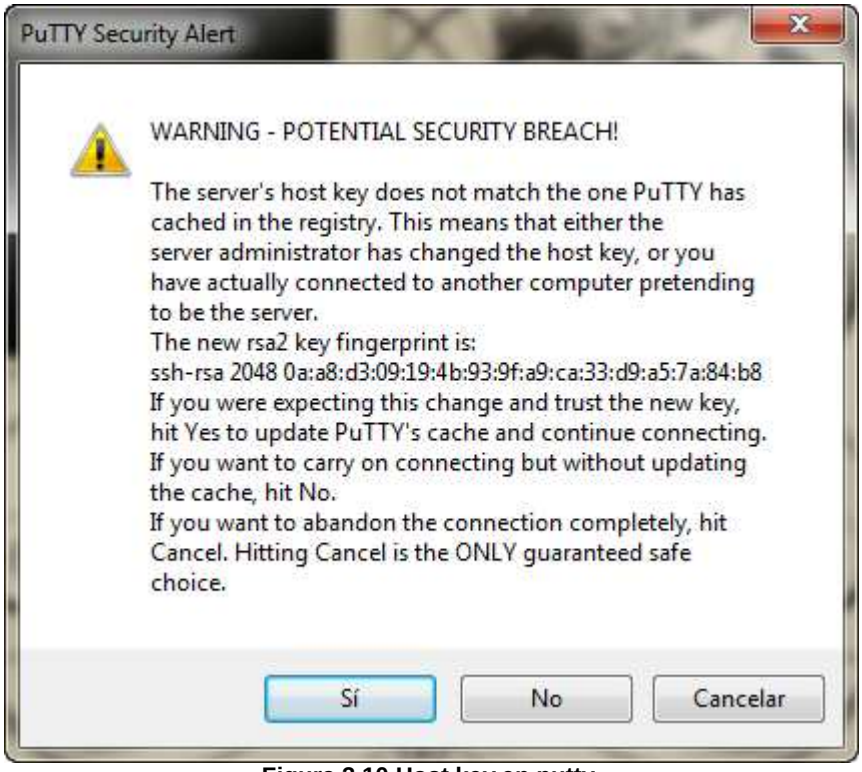

Figura 3.10 Host key en putty.

Al iniciar la sesión se abre una ventana que inicia la conexión, con el usuario configurado. (Figura 3.11)

Figura 3.11 Autenticación para SSH.

Solicita la contraseña para acceder al servidor, se debe de ingresar, y pulsar enter, para iniciar la autenticación, si todo ha sido correcto se ha estable el túnel, de forma contraria mandara un mensaje de error. (Figura 3.12)

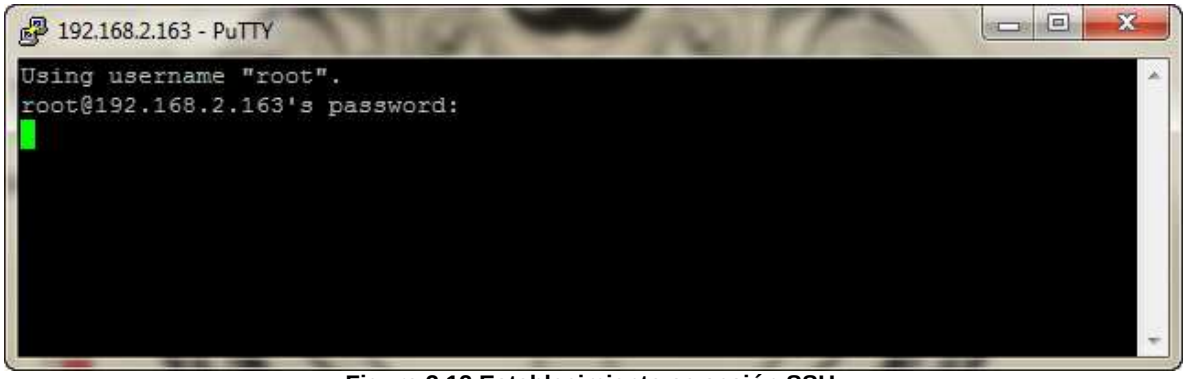

Figura 3.12 Establecimiento se sesión SSH.

Una forma de verificar el túnel es realizando un simple Telnet al puerto local 3306, (localhost) y verificar que la respuesta es el nombre del servidor MySQL en Ubuntu. (Figura 3.13)

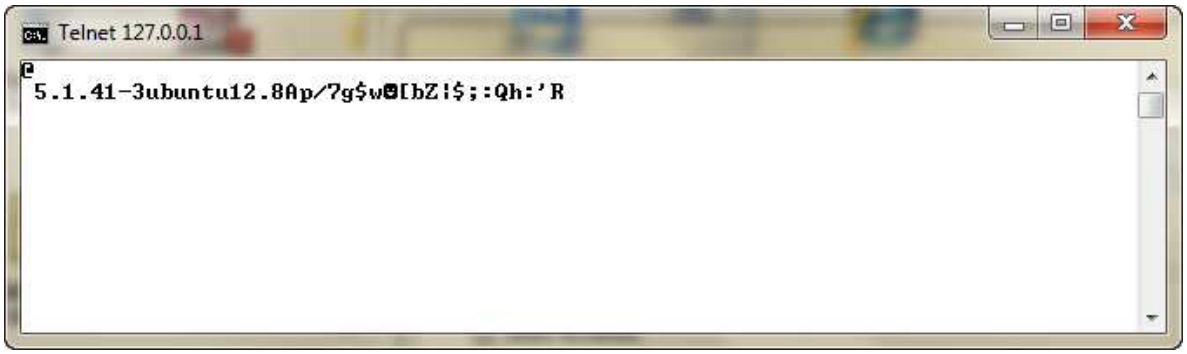

Figura 3.13 Prueba de conexión via Telnet.

en caso contrario muestra un error. (Figura 3.14)

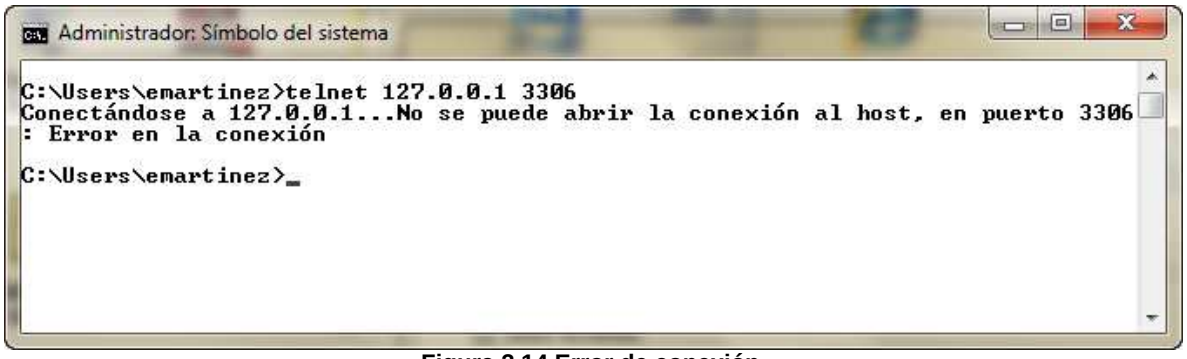

Figura 3.14 Error de conexión.

De esta manera se finaliza la configuración de la conexión del lado del cliente, se ha visto que ahora la conexión de forma local procesa información solicitada de un servicio remoto.

Una vez que se a finalizado la configuración en el equipo cliente y el servidor, solo falta realizar una conexión, para verificar que desde el equipo cliente se realiza una solicitud al servidor de datos con una configuración de conexión local y que el servidor envié una respuesta a la solicitud, para que de esta forma; una vez que se tiene el escenario iniciar la fase pruebas.

Para ello se ejecuta la herramienta MySQL Query Browser, e ingresar los datos, en el Server Host indicar que la conexión es de forma local y no remota, esto también ayuda a la seguridad del MySQL no permitiendo conexiones remotas, en lugar de esto todas las conexiones se realizan de forma local, simulando que el servidor de datos esta instalado en el equipo cliente y no en un equipo remoto. (Figura 3.15)

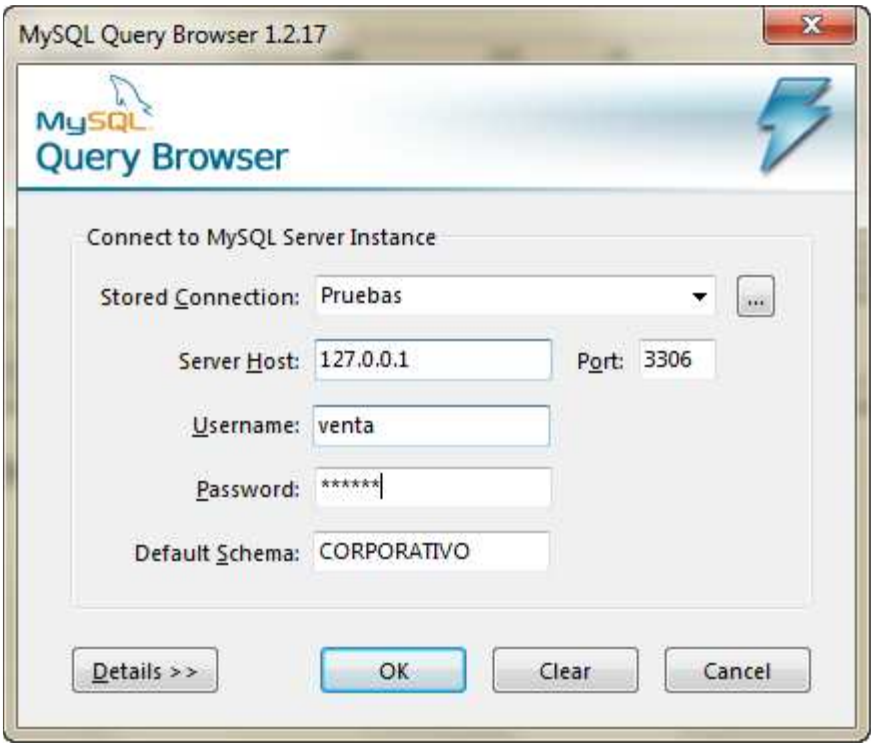

Figura 3.15 Conexión MySQL mediante túnel SSH.

Ahora las peticiones y la conexión al servidor de datos viajan por el túnel y los datos van cifrados(Figura 3.16), y con ello se ve disminuido el riesgo de que alguien intercepte los datos y vea su contenido, con este paso ya se ha asegurado en parte la integridad de los datos, ya que al realizar una petición al servidor de datos sin el túnel, al capturar los paquetes, podemos ver el contenido fácilmente.

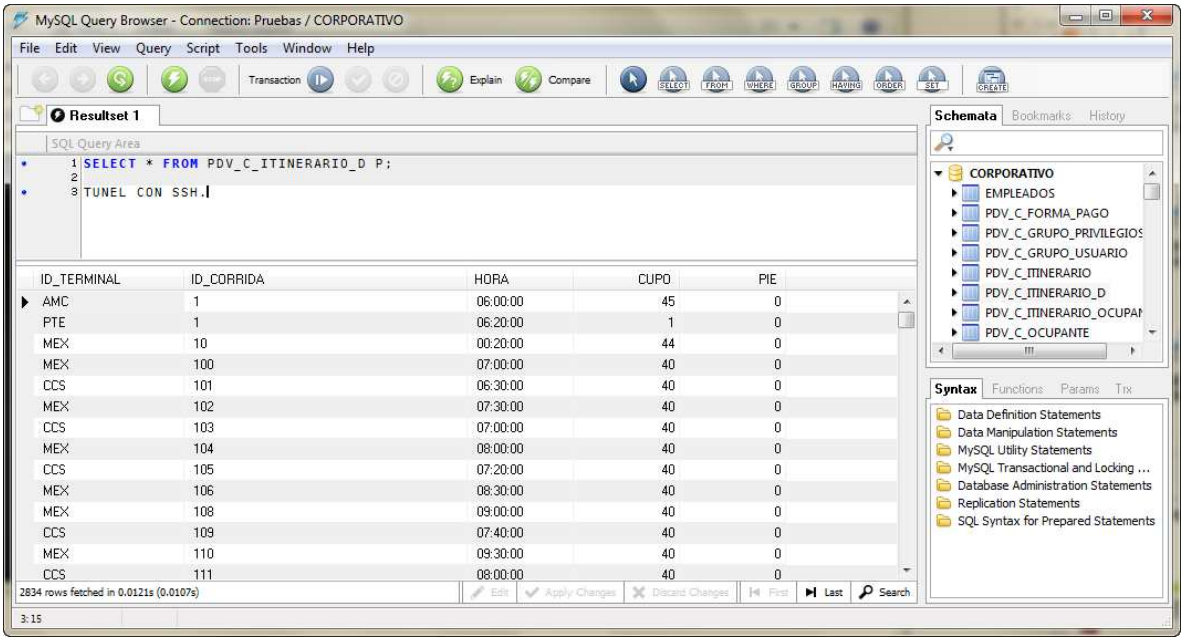

Figura 3.16 Solicitud realizada al servidor através del túnel SSH.

En el equipo cliente la solicitud es enviada al puerto 3306 (Figura 3.17) esta solicitud es re direccionada al puerto 22 el del SSH, aquí el SSH realiza el establecimiento del túnel, primero el cliente y el servidor comparten la clave de host, si la maquina cliente no encontró una clave pública previa, entonces se le pregunta al usuario si acepta la clave no confiable. Después, utilizan están claves públicas para negociar una clave de sesión, que se utilizará para cifrar todos los datos posteriores de la sesión mediante un bloque de cifrado tipo Triple-Des (3DES), blowfish o IDEA.

Es típico para este tipo de conexión, que en esta ronda de negociación de intercambio de clave sea completamente transparente para el usuario final. Solo después de que esté preparada la sesión para cifrarse es cuando se solicitaran las credenciales de acceso.

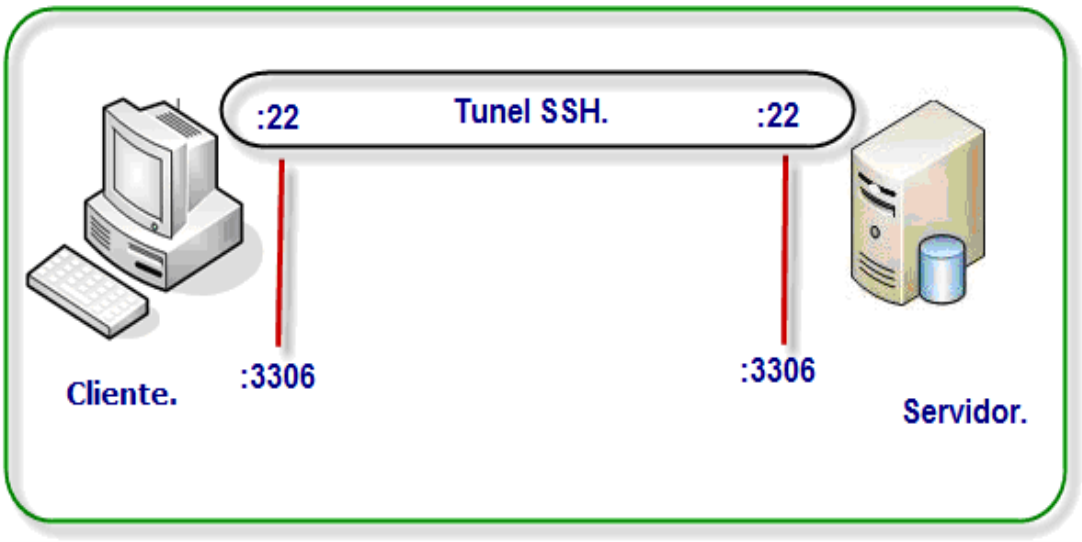

Figura 3.17 Diagrama del túnel SSH.

Por defecto, un servidor intenta autenticar al cliente utilizando certificados RSA o DSA (pares de clave). Si el cliente tiene un certificado reconocido por el servidor, este solicitará el certificado de la clave privada del software cliente; si es satisfactorio, el certificado es utilizado por el cliente y el servidor SSH para completar la autenticación mediante pregunta-respuesta, que le indica al servidor que el cliente posee la clave privada que corresponde a la clave pública que tiene registrada.

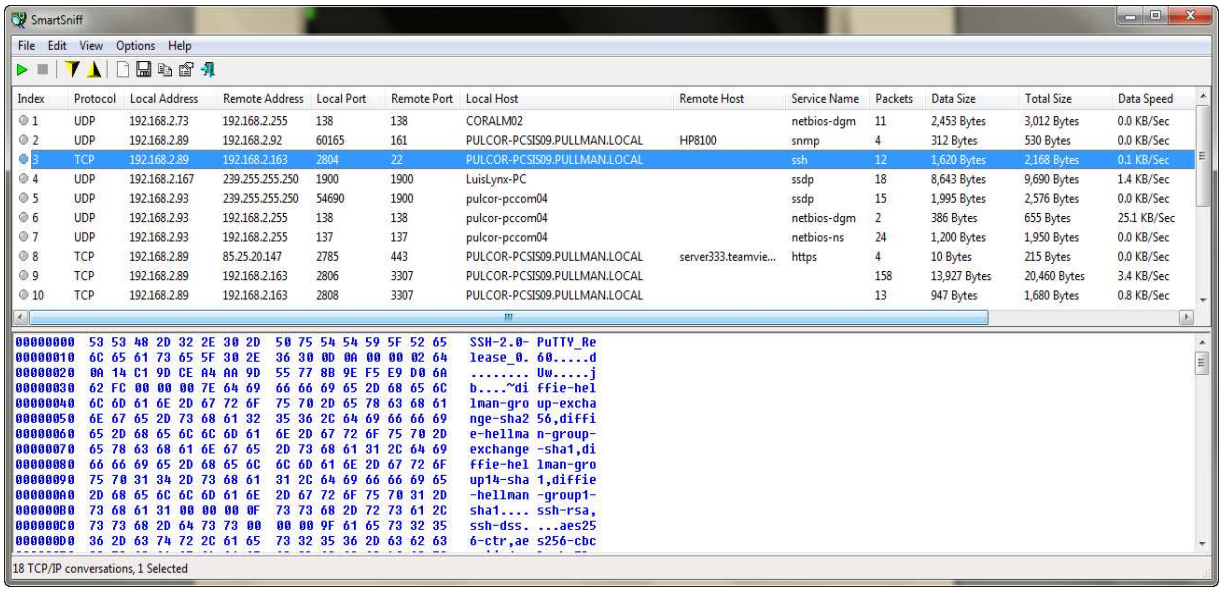

Figura 3.18 Captura de datos en el túnel SSH.

También por defecto, si la autenticación RSA/DSA falla o no hay certificado cliente, el servidor remoto le solicitara al usuario una combinación estándar Unix de usuario/contraseña que sea válida para el sistema remoto, Recuerde, ya se ha establecido una sesión cifrada entre el cliente y el servidor, de ahí que el usuario/contraseña no se podrán manipular al ir cifrados. Finalmente, después de lograr la autenticación comienza la sesión propiamente dicha.

En el cliente se cifra (Figura 3.18) la información recibida del puerto 3306 para ser enviada al puerto 22 del equipo servidor, una vez que llega al puerto 22 del equipo servidor, es descifrada la información y reenviada a su servicio origen en este caso el puerto 3306, para que el MySQL procese la consulta, cuando el servidor tenga el resultado los envía al puerto 3306, que es re direccionado al puerto 22 del SSH, entonces aquí se cifra la información y es enviada por el túnel al puerto 22 del equipo cliente.

De esta manera cuando llegan al equipo cliente, la información es descifrada y re direccionada al puerto 3306 para que la respuesta sea entregada. De esta forma la conexión remota se realizo a través del túnel y la información cuando viajo, fue de forma cifrada. Así la configuración del servidor no permitirá el acceso remoto al MySQL por que el acceso se establece al equipo servidor, en especifico al puerto 22, y no al 3306 de forma directa, así simulamos que el MySQL solo permite conexiones locales.

3.3 Túnel SSL.

La idea principal de crear un túnel es crear un envoltorio de datos o paquetes de un protocolo inseguro dentro de los paquetes de otro cifrado. En esta sección veremos como stunnel, una utilidad de envoltorio SSL, que puede utilizarse para cubrir transacciones desde varias aplicaciones con túneles SSL cifrados.

Además de su relevancia para la seguridad Web, OpenSSL ha llevado a la creación de Stunnel, una de las herramientas de seguridad más versátiles y útiles en el reportorio de software de código abierto. Stunnel hace posible el cifrado de las conexiones enmascarando virtualmente cualquier servicio de puerto simple TCP mediante SSL, sin modificar el servicio. Cuando mencionamos cualquier servicio de puerto simple TCP, nos referimos a los servicios que están escuchando las conexiones de un solo puerto sin utilizar puertos adicionales para otras funciones.

Stunnel se apoya en OpenSSL para todas sus funciones de cifrado. Por tanto, para utilizar Stunnel, primero tiene que obtener e instalar OpenSSL en cada host donde se desee utilizarlo. La versión actual para la mayoria de las distribuciones Linux incluye los paquetes binarios de OpenSSL versión 0.9.7 o superior.

La implementación del túnel mediante el protocolo SSL, se realizó con la herramienta stunnel4 cuya distribución esta disponible tanto para versión Windows y para Linux. La configuración para realizar el túnel consta de dos partes, una del lado del servidor y otra en el cliente, que a continuación se específica. (Figura 3.19).

Configuración en el Servidor.

Se debe recordar que el equipo servidor es un sistema Linux con una distribución Ubuntu 10.10, se debe de abrir una terminal; para instalar stunnel4 ingresar en una terminal el siguiente comando:

```
$ apt-get install stunnel
```

A continuación se deben otorgar permisos de lectura a stunnel.pem que es él certificado que instala por default stunnel y que se recomienda solo para realizar pruebas, no se utilizará otro certificado, porque no está dentro de los alcances de este documento.

```
/etc/stunnel# apt-get install stunnel4
/etc/stunnel# chmod 0400 stunnel.pem
```

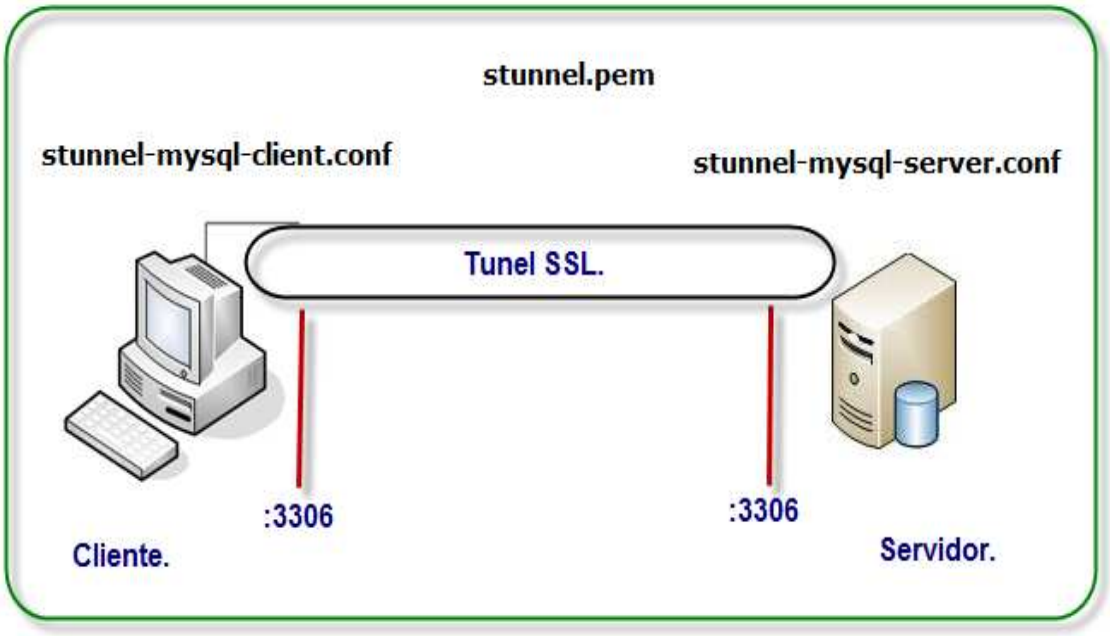

Figura 3.19 Esquemea de archivos de configuración para el túnel SSL.

En las versiones previas de Stunnel (versión 4) aceptaba todas las configuraciones desde la línea de comandos. En la versión en curso (versión 4 y posteriores), sin embargo, Stunnel utiliza un archivo de configuración, stunnel.conf. De hecho, la localización de este archivo de configuración es lo único que se puede específicar con los parámetros del comando en línea. Para lo cual se crea un archivo de configuración para iniciar el túnel:

```
vi stunnel-mysql-server.conf
```

dentro de este archivo, se debe específicar la ruta en donde se encuentra el certificado, además de establecer los puertos de conexión, para este trabajo se utiliza el certificado que instala por default el stunnel, este certificado lo recomiendan tan solo para realizar pruebas, razón por la cual lo utilizaremos, si se necesita mayor seguridad se debe utilizar un certificado más seguro, para fines prácticos y demostrativos no se detallan las características de este certificado, tan solo se utiliza.

`stunnel-mysql-server.conf`

```
cert = /etc/stunnel/stunnel.pem
# chroot = /var/run/stunnel4/
pid = /stunnel.pid
#setuid = nobody
#setgid = nobody
```

Para la configuración de los puertos en donde se va establecer el tunel se utilizo el puerto 3307 local va a hacer el lugar por donde se va a establecer el tunnel SSL. Con esta configuración se coloca al puerto 3307 del servidor a la escucha, con lo cual toda solicitud que llegue a este puerto lo va a redireccionar al puerto 3306 del mismo equipo. Para de esta forma simular que las peticiones realizadas al MySQL son de manera local.

```
[mysql]
accept = 3307
connect = 127.0.0.1:3306
```

Con esto se finaliza la configuración del archivo para establecer el túnel del lado del servidor quedando con la siguiente estructura.

stunnel-mysql-server.conf

```
cert = /etc/stunnel/stunnel.pem
# chroot = /var/run/stunnel4/
pid = /stunnel.pid
#setuid = nobody
#setgid = nobody
[mysql]
accept = 3307
connect = 127.0.0.1:3306
```

Ahora hay que establecer permisos de ejecución al archivo de configuracion stunnel-mysql-server.conf (Figura 3.20).

Establecer permisos: `/etc/stunnel#chmod 755 stunnel-mysql-server.conf`

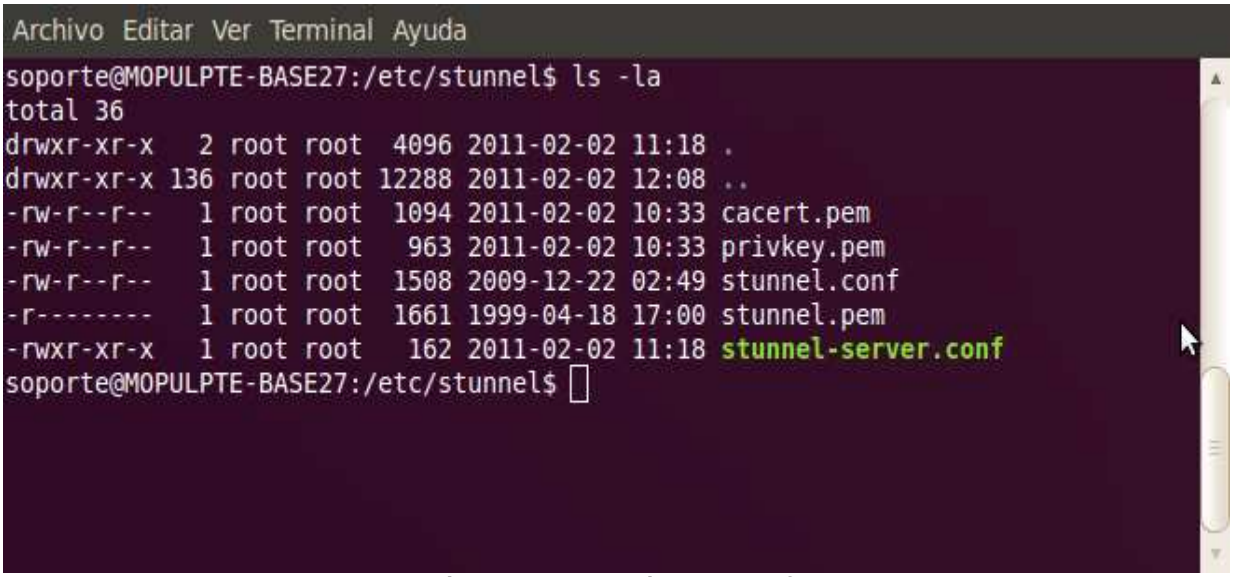

Figura 3.20 Stunnel-server-conf

Para realizar una verificación al archivo de configuración creado, hay que ejecutar el stunnel, pasando como argumento el archivo de configuración, si todo esta bien no indicara error, de forma contraria se debe analizar la alerta para encontrar el error. (Figura 3.21)

Ejecutar la instrucción para iniciar el servicio: **stunnel4 stunnel-mysql-server.conf**

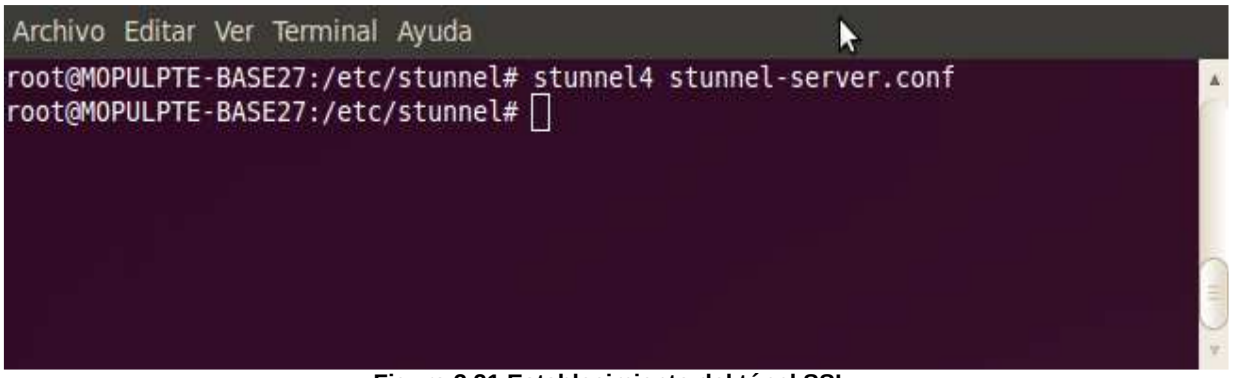

Figura 3.21 Establecimiento del túnel SSL

Finalmente solo falta verificar que el puerto configurado en este caso el 3307, este a la escucha de solicitudes, este paso es necesario realizarlo para verificar que el puerto esta abierto a la espera de peticiones, si no se realiza este proceso, al tratar de establecer el túnel, no se podrá asegurar que la conexión se realice con éxito, si indicara un error y no se realizo la verificación del puerto, será mas difícil deducir el origen del error. (Figura 3.22) Para verificar la conexión se ejecuta la siguiente instrucción:

```
/etc/stunnel# netstat –pln | grep :3307
```

la salida debe de mandar algo parecido:

```
tcp   0   0.0.0.0:3307   0.0.0.0:*   listen    2847/stunnel4
```

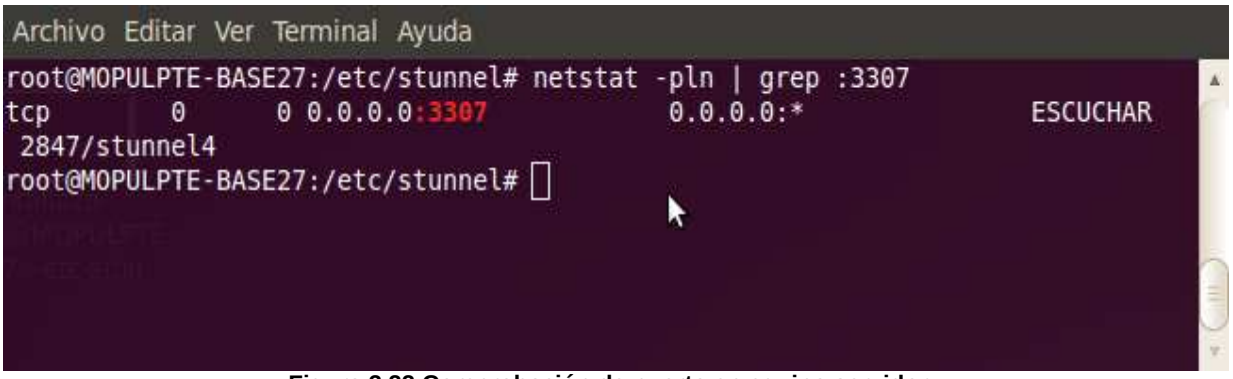

Figura 3.22 Comprobación de puerto en equipo servidor.

Con esta verificación se concluye la configuración de stunnel del lado del servidor.

Configuración en el Cliente.

La configuración del lado de cliente se realizó de la siguiente manera; se instaló el stunnel en su versión para Windows, y se han agregado las librerias (libeay32.dll ,libssl32) en la carpeta en donde se instaló el stunnel.

Instalar el: stunnel-4.34-installer.exe
Agregar a la carpeta de stunnel (C:\Program Files\stunnel) las siguientes dll
- libeay32.dll
- libssl32

Ahora al igual a lo realizado en el servidor, se crea un archivo de configuración, para establecer el túnel, tomando la estructura del archivo por default que instala el stunnel, el punto importante aquí es:

```
[mysqls]
accept  = 127.0.0.1:3306
connect = 192.168.2.163:3307
```

Aquí se indica que el servidor con la `ip x.x.x.x` que escucha el puerto `3307` , va a ser la conexión para todas las peticiones que se realicen al localhost por el puerto `3306`. De esta forma toda petición realizada a través del puerto `3306` del equipo local, será redireccionado al equipo remoto con `IP x.x.x.x` hacia el puerto `3307`. Así toda petición que se realice desde el equipo cliente al puerto local `3306`, será dirigida a la dirección `x.x.x.x` por el puerto `3307,` para de esta forma simular que la petición fue realizada de forma local y no remota.

De esta manera el archivo de configuración del lado del cliente queda con la siguiente estructura:

`stunnel-mysql-client.conf`

```
# Sample stunnel configuration file for securing MySQL
(client side)
socket=r:TCP_NODELAY=1
# Provide the full path to your certificate-key pair file
cert = stunnel.pem

# lock the process into a chroot jail
#chroot = /usr/local/var/run/stunnel/
# and create the PID file in this jail
#pid = /stunnel.pid

# change the UID and GID of the process for security
reasons
#setuid = nobody
#setgid = nobody
```

```
# enable client mode
client = yes

# Configure our secured MySQL client

[mysqls]
accept  = 127.0.0.1:3306
connect = 192.168.2.163:3307
```

Para establecer el túnel se ejecuta el stunnel con el archivo de configuración creado como argumento. Aquí se inicia el establecimiento del túnel de forma automática, la autenticación la realiza a través del certificado especificado en cada archivo de configuración, tanto del cliente como el del servidor, para de esta forma establer el túnel. (Figura 3.23)

Ejecutar la instrucción en una consola de ms-dos: `stunnel.exe stunnel-mysql-client.conf`

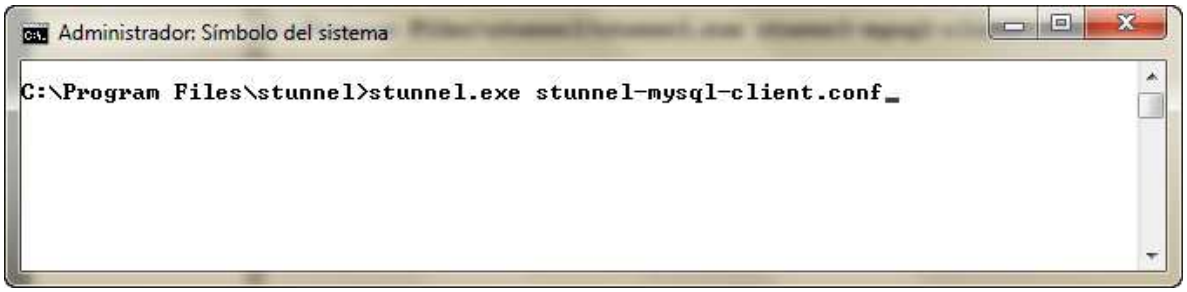

Figura 3.23 Establecimiento del túnel SSL en el cliente.

Se verifica que se este ejecutando el servicio de stunnel debido a que en el paso anterior, no se visualiza ningún mensaje a excepción de que haya ocurrido un error al establecer el túnel. (Figura 3.24)

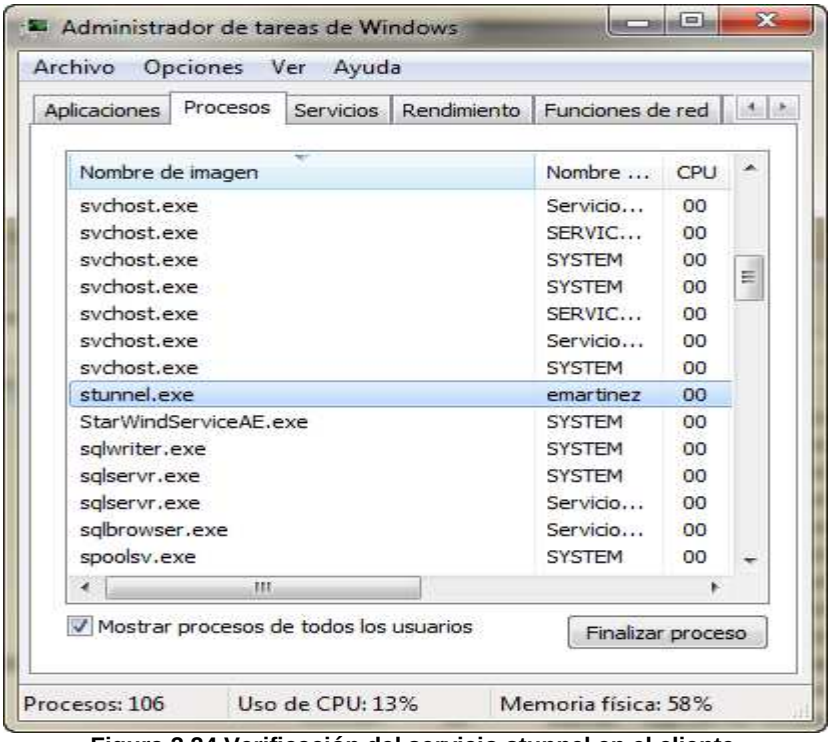

Figura 3.24 Verificación del servicio stunnel en el cliente.

Para realizar una prueba de conexión, se realiza una conexión Telnet de forma local por el puerto 3306 (Figura 3.25), para verificar que la respuesta sea devuelta por el servidor de datos, que viajara por el túnel. La respuesta esperada será el nombre y la versión del MySQL.

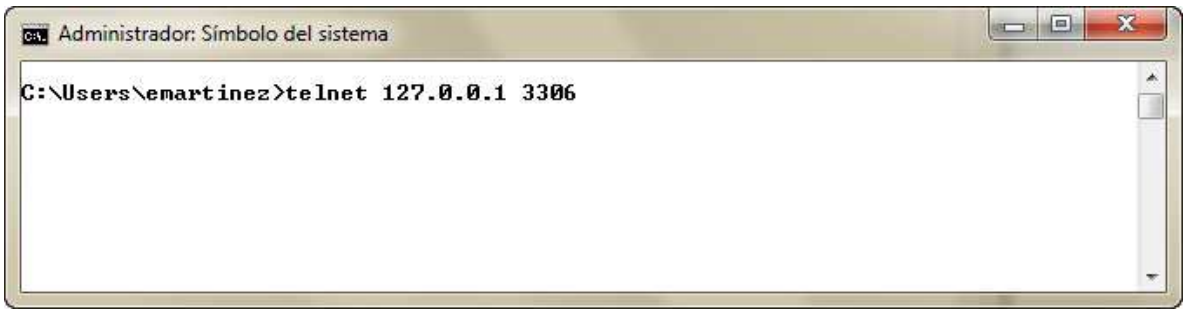

Figura 3.25 Prueba de conexión del túnel SSL en el cliente.

Si se establece la conexión, entonces el servidor envía su respuesta a la solicitud realizada. (Figura 3.26)

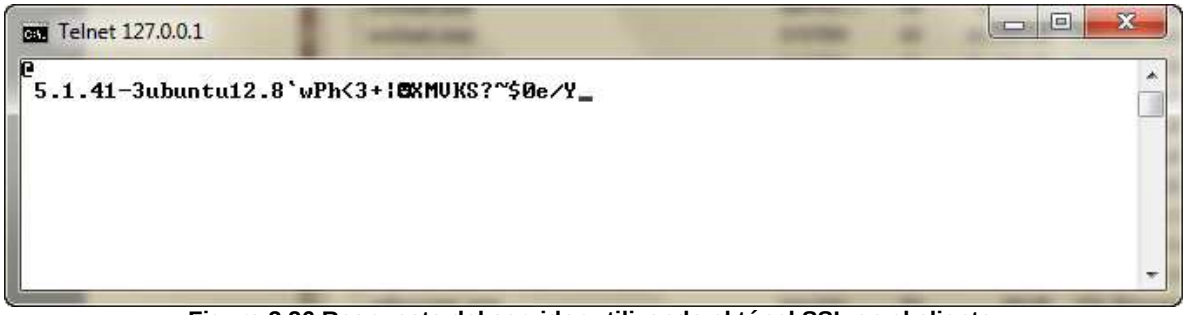

Figura 3.26 Respuesta del servidor utilizando el túnel SSL en el cliente.

Ya que se ha verificado la conexión entonces se puede realizar una conexión al servidor MySQL, por el puerto 3306 de forma local 127.0.0.1. (Figura 3.27)

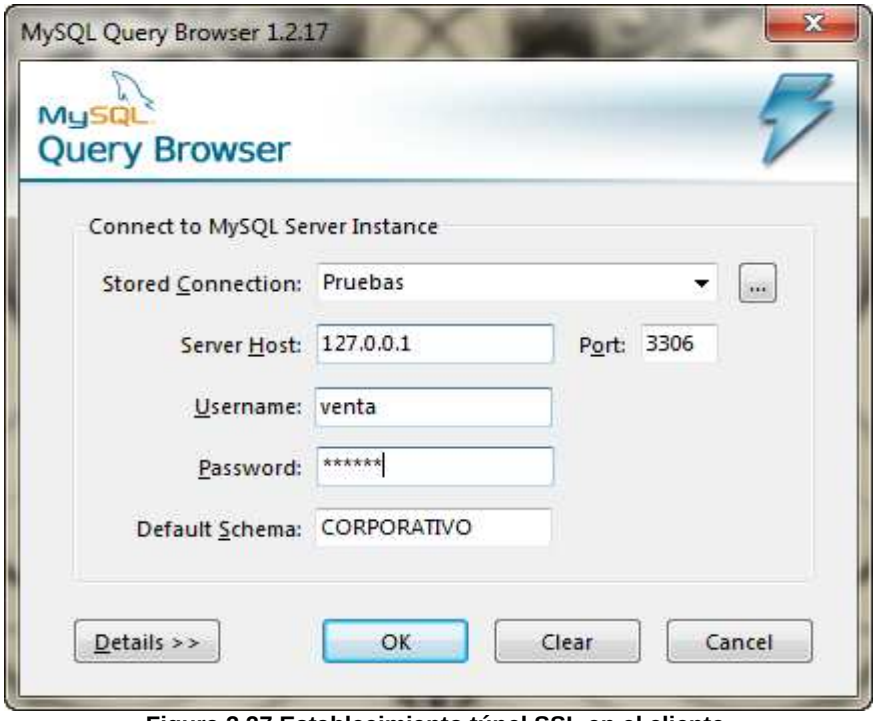

Figura 3.27 Establecimiento túnel SSL en el cliente.

Entrar al servidor MySQL, de forma local y realizamos una consulta a la base de datos para que la solicitud y la respuesta viajen por el túnel SSL, y que estos datos esten cifrados.Para de esta forma tener el sistema de seguridad planteado funcionando, asegurando la integridad de los datos al viajar por el túnel. (Figura 3.28)

Finalmente se ha establecido el túnel SSL através de la herramienta Stunnel, la conexión se ha establecido, debido a que en el archivo de configuración del cliente se indica que sean aceptadas todas las conexiones solicitadas vía localhost al puerto 3306, esto debe se generar un error en primera instancia por que el servidor de datos no esta instalado en la maquina cliente como para poder realizar solicitudes locales, pero como el puerto esta redireccionado al stunnel, este las acepta.

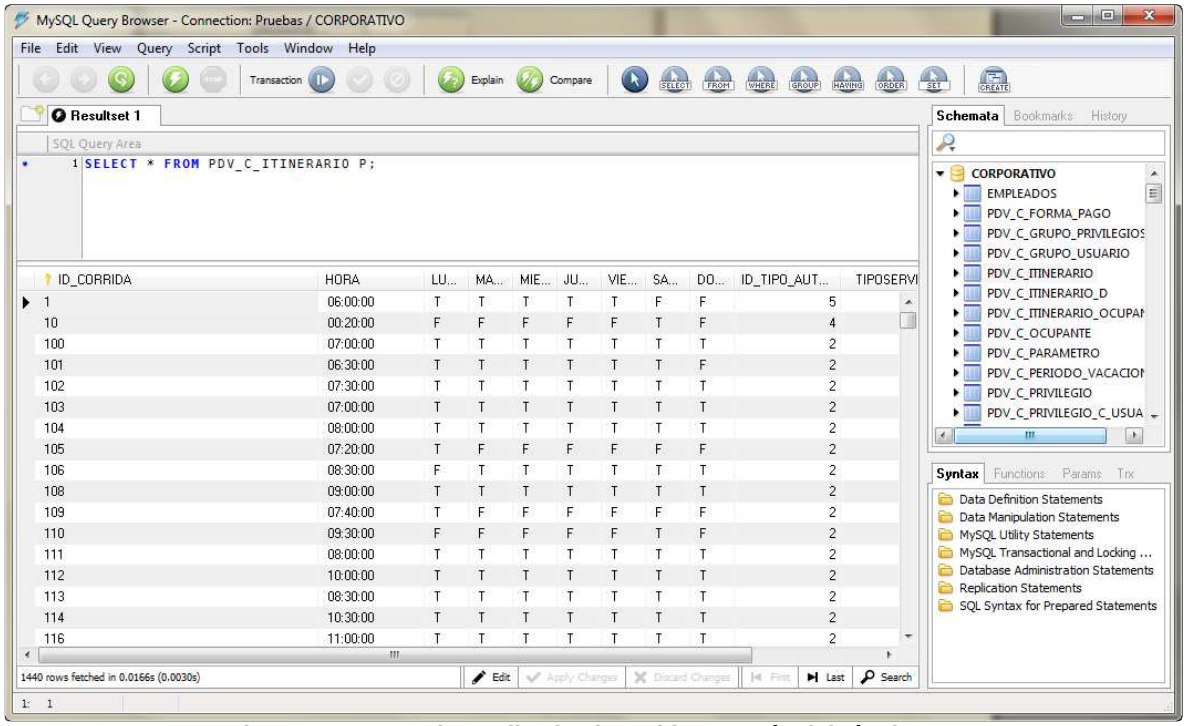

Figura 3.28 Consulta realizada al servidor através del túnel SSL.

Aquí stunnel realiza el trabajo, una vez que tiene una solicitud aceptada, entonces establece, una sesión al servidor por el puerto 3307, aquí se realizá todo el mecanismo de autenticación utilizado por SSL, es decir, utiliza los protocolos de establecimiento de sesión, además válida el certificado utilizado, una vez aceptados, entonces, la autenticación a finalizado, y con ello se inicia la sesión de SSL, mediante la cual viajarán los datos de forma cifrada.

Del lado del servidor, la configuración establece que acepte todo lo que provenga del puerto 3307, y entonces toda esta información una vez que ha llegado al servidor es descifrada, recordar que el cliente cifro los datos antes de enviarlos, entonces una vez que los descifro los redirecciona al puerto local 3306, que es el servidor de datos, así el servidor procesa la solicitud y se vuelve a realizar nuevamente el regreso de forma inversa, con esto la información viaja a través del túnel de forma cifrada y se ha establecido una conexión remota hacia un servidor de datos. (Figura 3.29)

Con el establecimiento de este túnel la información ya no viaja en texto plano, y con ello esta en funcionamiento en sistema de seguridad, que ayuda a que la integridad de los datos se mantenga y no sea expuesta a ataques. Además la autenticación también se ve fortalecida por que primero se realiza una autenticación de un usuario en el servidor y después se establece una autenticación en el servidor de datos pero con la diferencia de que ahora toda esta autenticación esta cifrada.

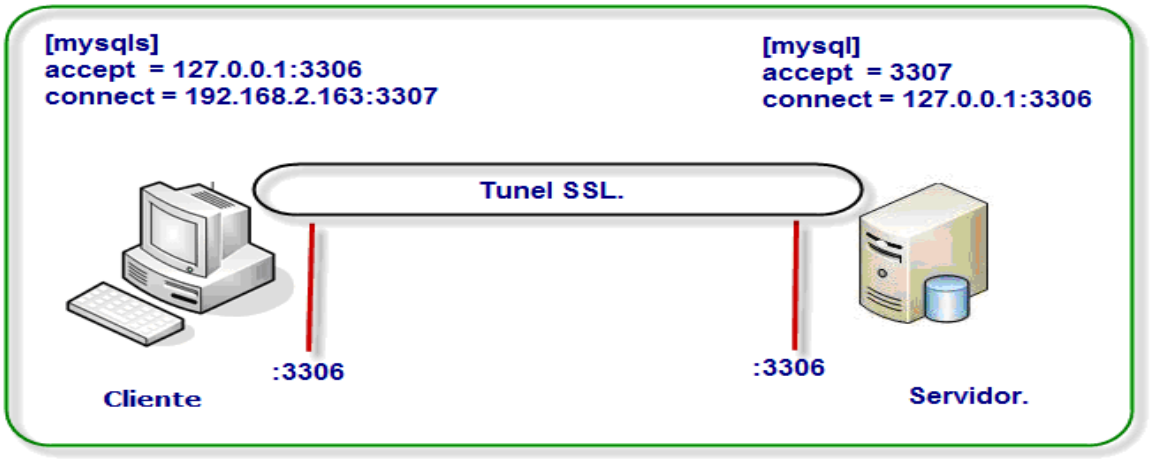

Figura 3.29 Escenario de prueba para el túnel SSL.

Para que de esta manera no exponer los datos, en un medio en el cual alguien los pueda interceptar, y con ello sufrir de un ataque, si en algún momento se pudiera obtener la información, esta en primera instancia seria dirigida al servidor, entonces el que sufre las consecuencias es el sistema operativo y las bases de datos no serian el primer objetivo de ataque, y con ello se puede mantener la integridad y hasta cierto punto la disponibilidad de la información. (Figura 3.30)

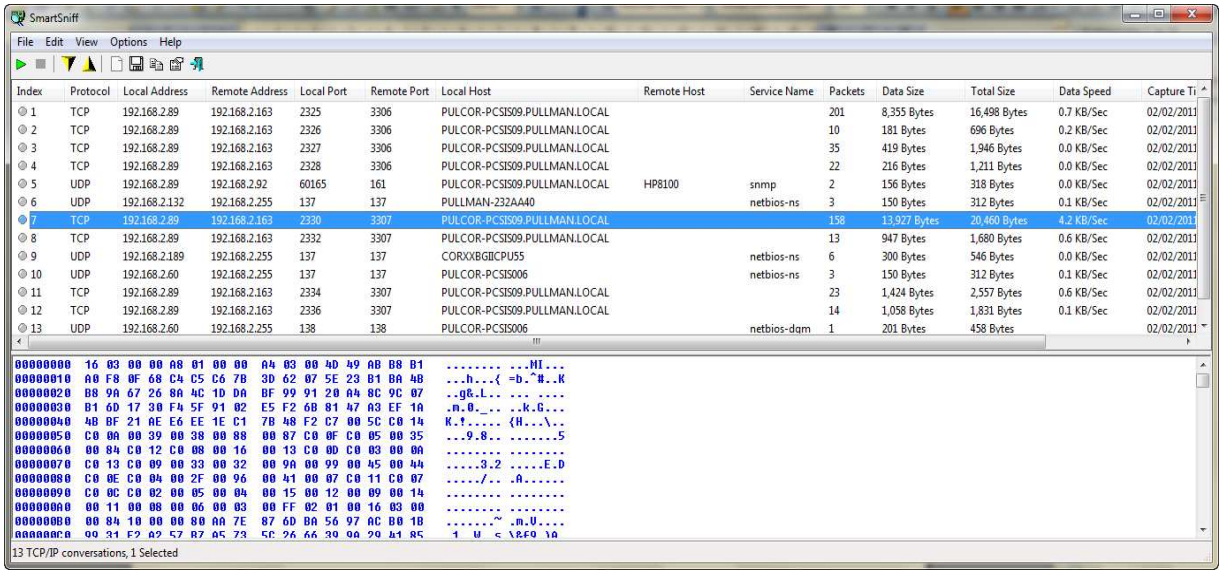

Figura 3.30 Captura de datos en el túnel SSL.

Ademas con stunnel se pueden encapsular conexiones TCP de y hacia puertos arbitrarios entre un servidor y un cliente. Se puede redirigir un puerto local a uno remoto en donde está corriendo el demonio stunnel el cual a su vez redirige ese puerto a uno estándar donde está escuchando algún servicio.

CAPITULO IV. PRUEBAS Y RESULTADOS.

Una vez que se han configurado y establecido los túneles para cada protocolo, se realizaron solicitudes de petición de información al servidor de datos, para obtener un conjunto de datos necesarios para realizar la comparativa de características entre los túneles. Esta comparativa tiene como objetivo presentar datos que determinen cual de los dos túneles establecidos es más rápido en la comunicación establecida entre el cliente y el servidor, además de analizar el rendimiento de cada túnel, para finalizar con las vulnerabilidades que cada túnel puede tener.

El tema de vulnerabilidad es muy importante ya que; como se sabe, no se puede crear un sistemas de seguridad de la información, completamente seguro al grado de garantizar, que con el sistema y la gestión puntual de las políticas no se va a sufrir de algún ataque sin verse afectado el sistema. Por ello se dice que hay que detectar las vulnerabilidades de un sistema para poder minimizar el riesgo de un ataque, y si se sufre de un ataque que las consecuencias sean mínimas y aceptables.

Esta apartado tiene como finalidad mostrar que el estableciemto de los túneles; aumenta la seguridad para mantener la integridad de los datos, sin dejar de lado que también tiene vulnerabilidades, pero el beneficio es mayor ya que los datos no están a disposición de cualquier proceso que pueda interceptar la comunicación, con esto se pretende minimizar el riesgo de ver comprometida la integridad y la disponibilidad, y como ningún sistema de seguridad informática es cien por ciento seguro, es mejor conocer las vulnerabilidades del sistema, para asumir el riesgo o para establecer aun mas, políticas de seguridad que ayuden a minimizar las vulnerabilidades hasta un grado que se aceptable para el sistema.

Las pruebas contempladas, son realizar un análisis de velocidad, una prueba para analizar el rendimiento, y finalmente exponer los casos de alertas de seguridad emitidos por un CERT en este caso el **SSI** (Subdirección de Seguridad de la Información**)/UNAM-CERT** de la Dirección General de Cómputo y de Tecnologías de Información y Comunicación, UNAM, detectados para ambos protocolos. Con los resultados de estas pruebas, se puede conocer el comportamiento que tiene cada túnel en relación a los tiempos de respuesta y el rendimiento del equipo al establecer los túneles, características que dan una idea general, para poder decidir cual utilizar y que puede utilizarse como punto de partida, para la realización de túneles de acuerdo a características especificas y objetivos concretos, velocidad o seguridad.

4.1 Velocidad.

La prueba de velocidad, es solicitar información al servidor de datos, utilizando sentencias predeterminadas, que han sido obtenidas de procedimientos almacenados escritos para la base de datos que se esta utilizando, en este punto el lector puede determinar sus propias sentencias para su base de datos, dejando axial, un análisis para medir el rendimiento de los túneles para una base en especifico.

La recopilación de datos, se realizo de la siguiente forma, se establece conexión con el servidor mediante el túnel, luego a través del MySQL Administrator se realizan las peticiones de datos, esta consulta es ejecutada y muestra los resultados de la sentencia, esta herramienta, despliega los datos así como también los tiempos que ha transcurrido al ejecutar la sentencia, estos tiempos de ejecución son los que se han utilizado y resumido en la Tabla 4.1, dejando al lector construir su propia tabla de datos para realizar su propia grafica de tiempos.

	Sin túnel.	Túnel SSH.	Túnel SSL.
1	**1051 rows fetched in 0.0147s**	**1051 rows fetched in 0.0205s**	**1051 rows fetched in 0.0137s**
2	**1578 rows fetched in 0.0056s**	**1578 rows fetched in 0.0057s**	**1578 rows fetched in 0.0057s**
3	**1623 rows fetched in 0.0075s**	**1623 rows fetched in 0.0111s**	**1623 rows fetched in 0.0077s**
4	**3001 rows fetched in 0.0218s**	**3001 rows fetched in 0.0332s**	**3001 rows fetched in 0.0221s**
5	**8068 rows fetched in 0.0213s**	**8068 rows fetched in 0.0373s**	**8068 rows fetched in 0.0258s**
6	**2015 rows fetched in 0.0063s**	**2015 rows fetched in 0.0083s**	**2015 rows fetched in 0.0067s**
7	**3043 rows fetched in 0.0093s**	**3043 rows fetched in 0.0129s**	**3043 rows fetched in 0.0097s**
8	**3580 rows fetched in 0.0131s**	**3580 rows fetched in 0.0220s**	**3580 rows fetched in 0.0138s**
9	**1 rows fetched in 0.0013s**	**1 rows fetched in 0.0011s**	**1 rows fetched in 0.0012s**
10	**2015 rows fetched in 0.0064s**	**2015 rows fetched in 0.0080s**	**2015 rows fetched in 0.0067s**

Tabla 4.1 Resultados de peticiones al servidor MySQL.

Para realizar la comparativa de velocidades, se toma como punto de partida el tiempo que marca el MySQL Query Browser hacia una petición realizada sin el establecimiento de un túnel. Cada sentencia se ejecuto de igual forma para cada túnel establecido, los tiempos emitidos por la herramienta, son los datos utilizados para la realización de la grafica.

Los tiempos obtenidos, se grafican para poder tener una interpretación visual del comportamiento de peticiones realizadas al servidor de datos, y de esta forma determinar con cual túnel se lleva mas tiempo en obtener una respuesta del servidor, ya que estos datos viajan en dos túneles distintos, con esto se pretende mostrar cual es mas rápido y seguro, es decir si cumple con ambas características o con alguna, y determinar cual utilizar.

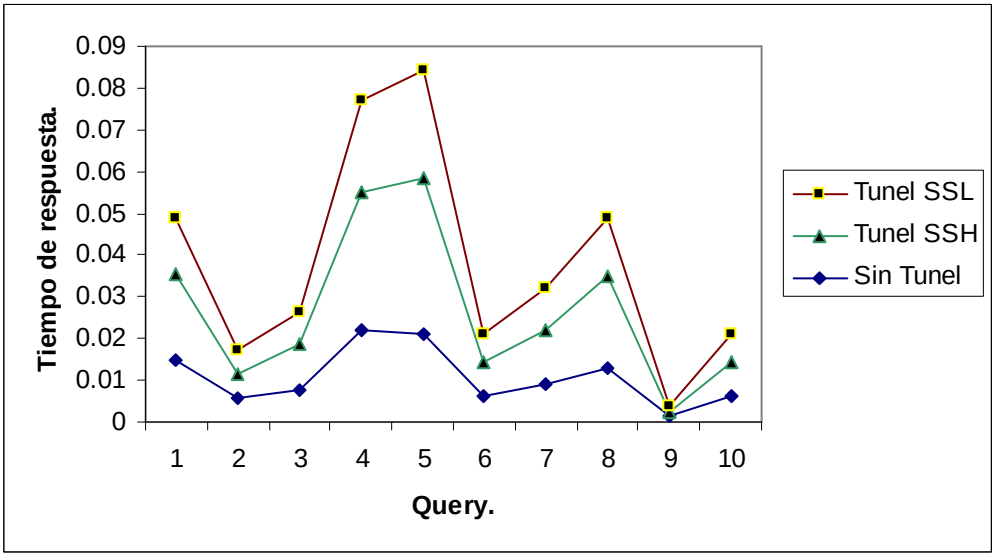

Figura 4.1 Grafica de tiempos de respuesta en los túneles.

De acuerdo a la necesidad o al planteamiento de una política de seguridad que se necesite establecer, es decir, si se necesita velocidad utilizar un túnel SSH, pero no será muy robusto en su sistema de autenticación al no utilizar algún certificado avalado por una CA. Por otro lado si se necesita mayor control de seguridad en los accesos, entonces se puede utilizar un certificado, que se obtiene de una CA, y que además se transfiere el riesgo a esta autoridad, para garantizar la integridad de los datos, utilizando un túnel SSL, pero que será más lento. En resumen de acuerdo al planteamiento de la política de seguridad se decide cual túnel utilizar.

4.2 Rendimiento.

La prueba de rendimiento consiste en medir el uso del CPU, al establecer el túnel y posteriormente realizar consultas a la base de datos; durante este periodo de tiempo, se mide el uso del CPU con la herramienta Monitor de Recursos de un sistema operativo Windows 7. De acuerdo al dato obtenido se realiza la comparativa de los dos túneles y sin túnel, para de esta forma obtener el dato de comparación.

Esta prueba tiene como finalidad, el determinar los recursos que utiliza en cuanto al procesador se refiere, es decir, al establecer los tuneles al iniciar el cifrado y el descifrado. Una vez establecido el túnel y que los datos estan viajando, cada vez que envia una solicitud al servidor el equipo cliente cifra la información y la prepara para enviarla a traves del tunel, cada vez que el servidor envia una respuesta se tiene que descifrar; esto requiere un procesamiento, y este procesamiento es el que se esta monitoreando para obtener la información necesaria para realizar la comparativa.

Para medir el rendimiento sin el establecimiento del túnel, se realiza una solicitud al servidor de datos, y se obtiene la información contenida en la Figura 4.2, la cual indica que el uso medio del CPU es del 3.10, y utiliza el 23% del total del CPU.

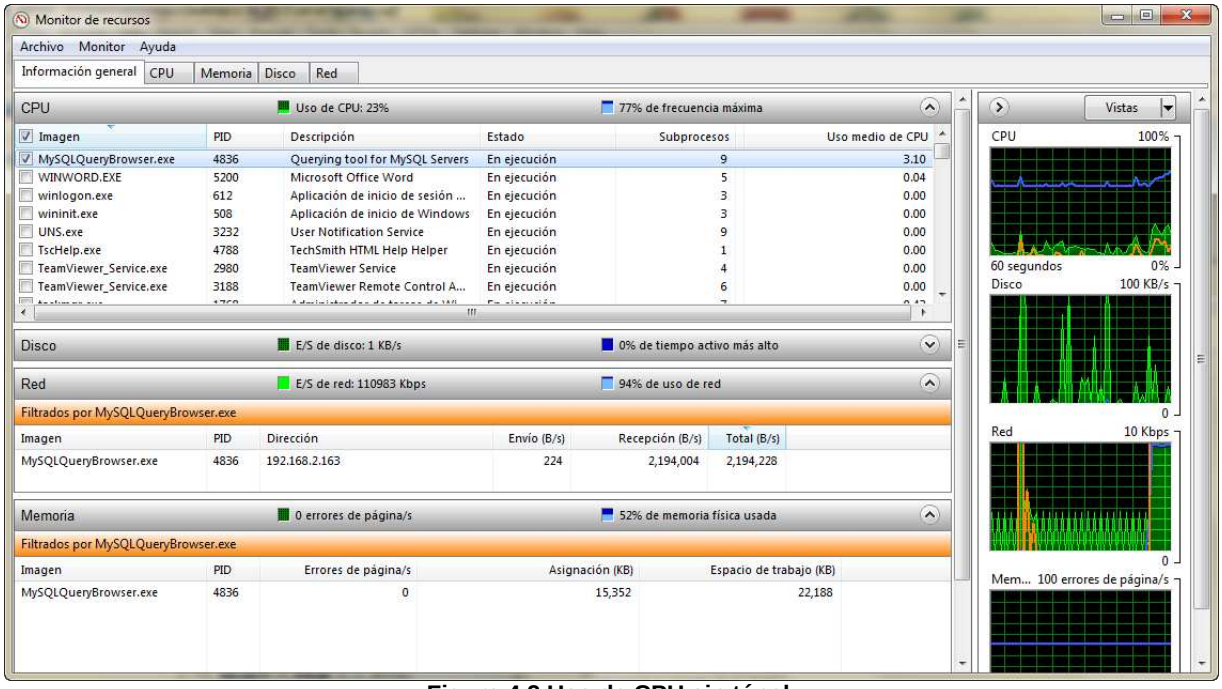

Figura 4.2 Uso de CPU sin túnel.

Para medir el rendimiento a traves del túnel SSH, se realiza una solicitud al servidor de datos, y se obtiene la información contenida en la Figura 4.3, la cual indica que el uso medio del CPU es del 0.28, y utiliza el 46% del total del CPU.

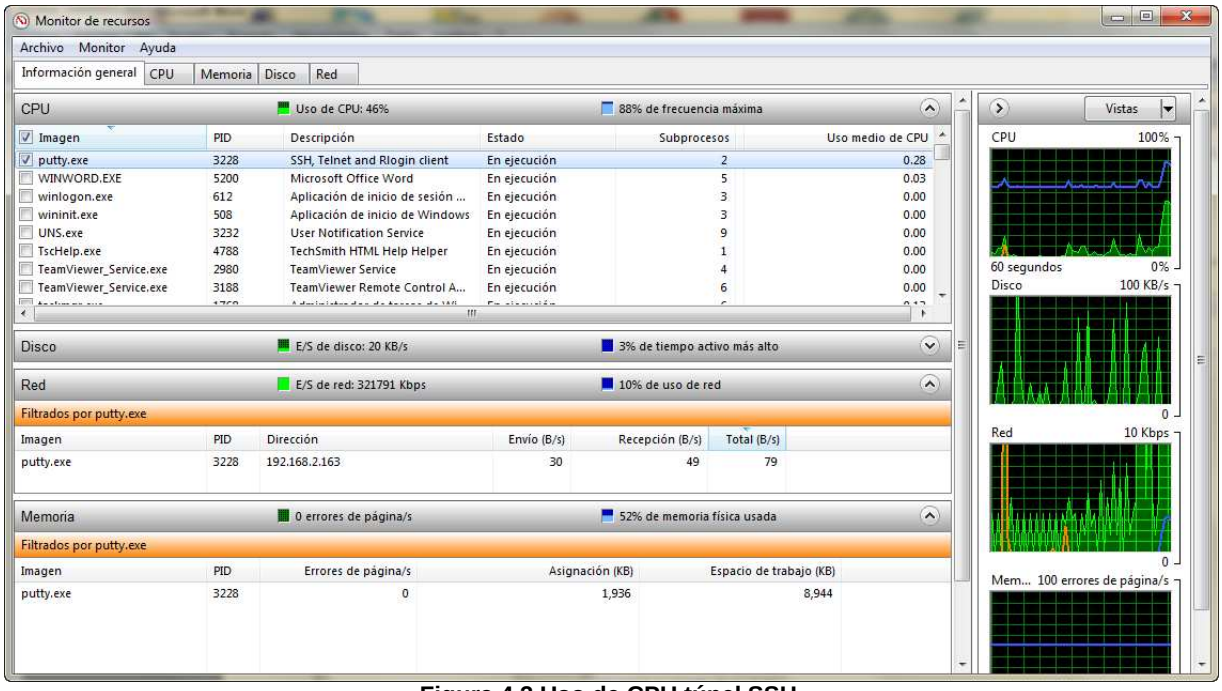

Figura 4.3 Uso de CPU túnel SSH.

Para medir el rendimiento a traves del túnel SSL, se realiza una solicitud al servidor de datos, y se obtiene la información contenida en la Figura 4.4, la cual indica que el uso medio del CPU es del 3.30, y utiliza el 44% del total del CPU.

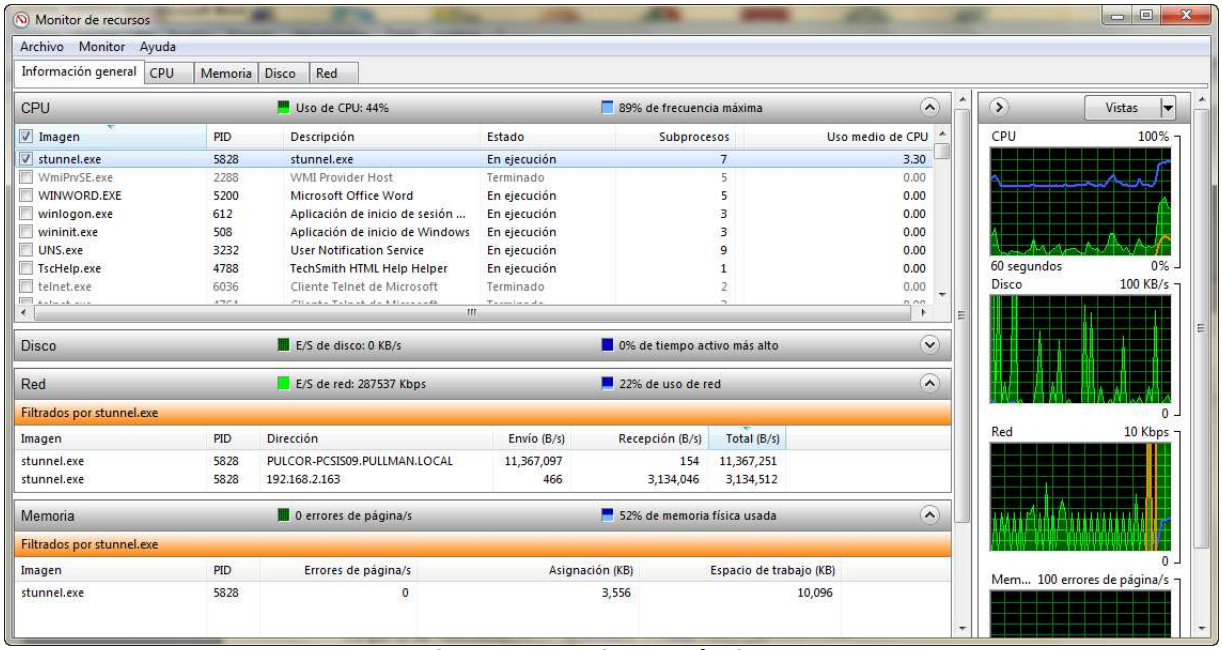

Figura 4.4 Uso de CPU túnel SSL.

Con la obtención de los datos anteriores, se construye la tabla 4.2, se han ordenado primero por tiempo medio del CPU, para finalizar con el uso del CPU. Y con esto se finaliza la pureba de rendimiento, hay que señalar que estas pruebas fueron realizadas en un equipo HP Compaq 8000 Elite Small Form Factor, que tiene un procesador Intel® Core (TM)2 Duo a 3.00GHz con 2 GB en RAM .

Ahora con los datos obtenidos de la tabla 4.2, se obtiene que sin el establecimiento del túnel la petición realizada hacia el servidor requiere del 23% del CPU y que el uso medio es del 3.10, esto indica sin duda alguna que el rendimiento es mejor sin un túnel, pero carece de medidas de seguridad, ya que los datos están viajando el texto plano y que eleva el riesgo de ver comprometida la integridad y la disponibilidad de los datos.

	Sin túnel.	Túnel SSH	Túnel SSL
Uso del CPU	23%	46%	44%

Tabla 4.2 Resultados de medición de uso de CPU.

Con el establecimiento del túnel SSH, el tiempo medio del CPU es de apenas el 0.28, mientras que el uso del CPU se eleva al 46%, esto indica mayor procesamiento, al utilizar este túnel, hay que señalar que de acuerdo a las pruebas de velocidad este túnel es mas rápido, pero como punto en contra, necesita mas procesamiento. Esto es un punto a considerar ya que se puede establecer un túnel con la finalidad de obtener mayor velocidad, pero si se establece con un equipo con características menores al equipo utilizado para esta prueba, el procesamiento puede reducir la velocidad que se pretende obtener.

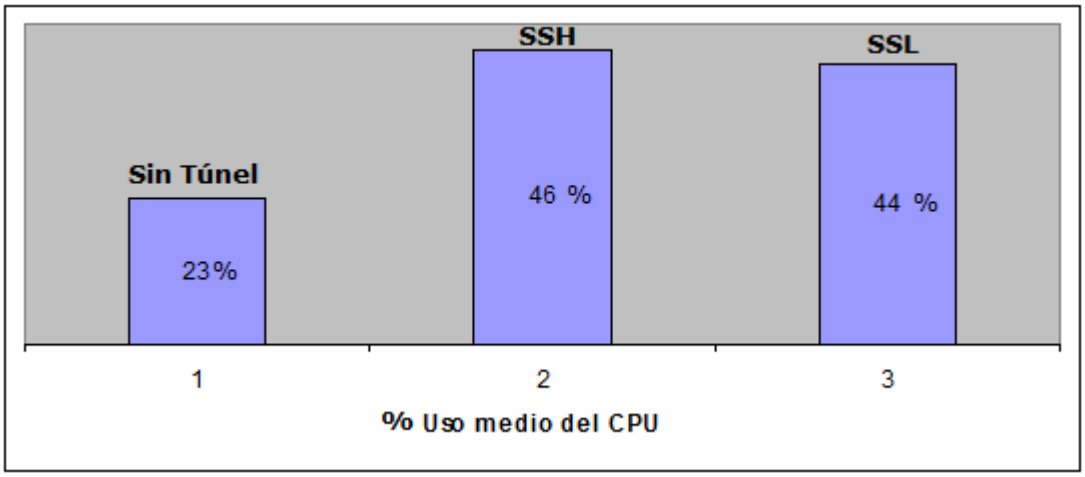

Figura 4.5 Grafica resultados de medición de uso de CPU.

Aun así, con el establecimiento del túnel SSH se deduce que se puede obtener velocidad y seguridad, ya que los datos antes de viajar por la red, son cifrados y tienen un tratamiento para asegurar que los datos no viajen en texto plano, además de reducir el riesgo de que alguien intercepte la comunicación, y pueda visualizar su contenido, además el protocolo SSH, agrega características de seguridad, como lo es una mejor autenticación entre el cliente y el servidor, con ello se cumple dos objetivos de la seguridad informática, como lo son la integridad de los datos y la disponibilidad de la información.

Finalmente para el túnel SSL, la información obtenida muestra que el uso medio es de 3.30, mayor en relación al túnel de SSH y en cuento al uso del CPU es del 44%, un poco mayor al túnel SSH, con esta información obtenemos condiciones semejantes en cuanto al uso del procesamiento, este procesamiento en gran medida es requerido en el momento de cifrar la información, pero la ventaja que tiene SSL sobre SSH, es el certificado que podemos utilizar, es decir obtener uno de una CA.

Este punto el del certificado agrega mayor seguridad, al túnel SSL, ya que la CA, garantiza en cierta medida la integridad de los datos al utilizar su certificado, para este proyecto, se utilizo un certificado de ejemplo para fines demostrativos y observar su funcionamiento, pero para un ambiente de producción es recomendable obtener un certificado expedido por una CA, y de esta manera se crea un estricto control de seguridad, que cuando la política de seguridad requiera seguridad, dejando en un segundo orden de importancia la velocidad es entonces, donde se puede utilizar un túnel SSL.

4.3 Vulnerabilidad.

Las vulnerabilidades de un sistema surgen a partir de errores individuales en un componente, sin embargo nuevas y complejas vulnerabilidades surgen de la interacción entre varios componentes como el kernel del sistema, sistemas de archivos, servidores de procesos, entre otros. Para reducir los riesgos de seguridad por las vulnerabilidades de software y de protocolo, se realiza un esfuerzo para abordar el número de vulnerabilidades existentes tanto en el software como en el protocolo que se está utilizando.

El trabajo de análisis de vulnerabilidad se divide en dos áreas. Identificar y reducir el número de nuevas vulnerabilidades. Se puede por tanto evidenciar, la gran importancia de desarrollar mecanismos de autoprotección contra estos ataques, los cuales deben pasar por una fase de identificación de los potenciales riesgos a los que se está expuesto, luego a una fase de análisis de las debilidades para posteriormente definir acciones de mejora y defensa así como planes de mitigación ante sucesos indeseables.

Para conocer las vulnerabilidades de cada protocolo, se ha consultado en la SSI (Subdirección de Seguridad de la Información)/UNAM-CERT de la Dirección General de Cómputo y de Tecnologías de Información y Comunicación , UNAM , que es un punto de encuentro al cual puede acudir la comunidad de cómputo para obtener información, asesorías y servicios de seguridad; así como para intercambiar experiencias y puntos de vista, logrando con ello, establecer políticas de seguridad adecuadas, disminuir la cantidad y gravedad de los problemas de seguridad y difundir la cultura de la seguridad en cómputo. [21]

En este lugar se han buscado reportes que nos muestren incidentes relacionados con los protocolos SSH y SSL, con el objetivo de descubrir vulnerabilidades de dichos protocolos, la información recabada, menciona que estos dos protocolos pueden sufrir ataques de tipo negación de servicio. Además emite propuestas para evitar tales ataques, este tipo de sitios son importantes ya que estos ofrecen mucha información relacionada a la seguridad informática.

Los reportes aquí presentados son:

- ***Vulnerabilidad de Seguridad UNAM−CERT−2006−106:Negacion de servicio en Open SSH.*** [22]

  En el primer documento se menciona que el openSSH puede sufrir un ataque de negacion de servicio debido a que existe una falla en el demonio openSSH relacionado con la autenticación de los usuarios. Esta advertencia esta contenida en el **CVE−2006−4924 (Common Vulnerabilities and Exposures)** el cual detalla que para el protocolo ssh versión 1 no se encarga adecuadamente del manejo de bloques duplicados al momento de ser ingresados.
  Esto puede permitir a atacantes remotos consumir significativamente recursos del CPU conduciendo a una negación de servicio.

  Otra advertencia esta redacatada en el CVE−2006−5051 esta relacionada con el manejador de señales race condition podría permitir potencialmente a atacantes remotos el detener sshd y podría teóricamente conducir la habilidad de ejecutar código arbitrario.

  Una falla en el demonio de OpenSSH permite a atacantes remotos sin autenticar poder provocar una negación de servicio.

  La vulnerabilidad es causada debido a un error dentro del manejo de múltiples bloques idénticos dentro de un paquete de ssh, si el protocolo ssh es habilitado, esto puede causar una negación de servicio al consumir el procesamiento del CPU, puede ser realizado enviando paquetes específicamente creados.

- ***Boletin de Seguridad UNAM−CERT 2004−004:Múltiples vulnerabilidades en OpenSSL***.[23]

  OpenSSL implemtenta los protocolos Secure Sockets Layer (SSL) y Transport Layer Security (TLS) e incluye una biblioteca criptográfica de propósito general. SSL y TLS son comúnmente usadas para proporcionar servicios de autenticación, encripción, integridad y no repudio para aplicaciones de red como HTTP, IMAP, POP3, STP y LDAP. OpenSSL es ampliamente utilizado entre una diversidad de plataformas y sistemas. En particular, muchos ruteadores y otros tipos de equipo de red usan OpenSSL.

  El National Infrastructure Security Co−ordination Centre (NISCC) del Reino Unido y el OpenSSL Project han reportado tres vulnerabilidades en la biblioteca SSL/TLS de OpenSSL (libssl). Cualquier aplicación o sistema que usa esta biblioteca podría ser afectado.

  Los usuarios que no cuenten con los parches correspondientes son vulnerables a una negación de servicio

Para establecer un sistema de seguridad informática, es necesario conocer que existen protocolos de seguridad, como los utilizados en este proyecto, además las herramientas en este caso el software utilizado, como son OpenSSH, el putty y el Stunnel tienen vulnerabilidades que no garantizan por completo la seguridad al utilizarlos, pero como ningún sistema puede garantizar esta característica, es necesario conocer sus vulnerabilidades para que con este conocimiento establecer los medios de acción y respuesta a incidentes.

Todo el software utilizado en este proyecto cuenta con una sección de incidentes y de posibles problemas encontrados y para los cuales emiten una solución para cubrir la vulnerabilidad, es necesario acudir a los sitios oficiales, para conocer las vulnerabilidades que tienen, este punto es importante para realizar un análisis de vulnerabilidad y decidir si se va a utilizar el software, al conocer las vulnerabilidades, se sabe que amenazas pueden afectar al sistema y de esto establecer mediadas de precaución, de monitoreo y de respuesta a incidentes.

CONCLUSIONES.

En el planteamiento de un sistema de seguridad para la protección de los datos, se tienen que contemplar varios aspectos para estructurar el sistema de tal forma que contemple cubrir varios aspectos que minimicen los efectos de un ataque a alguna vulnerabilidad de nuestro sistema, para ello en la fase de gestión, es necesario establecer políticas adecuadas a las necesidades prioritarias para proteger el activo de la empresa.

La información es un activo que puede llegar a tener gran relevancia para una empresa y su protección llega a ser fundamental, es en esta situación en la cual gestionar sistemas de seguridad es importante, bajo esta circunstancias, se elaboro este proyecto que presenta un sistema para proteger la información, ya que sin un sistema como el que aquí ha sido planteado, la información es muy vulnerable y ve comprometida su integridad.

Los túneles desarrollados y presentados cubren características de seguridad para la autenticación y el envío de la información, el objetivo se ha cumplido, se han creado dos canales de comunicación seguros, esta seguridad esta fundamentada en el tratamiento que tiene la información, con el cifrado, ya no viaja en texto plano y ahora resulta difícil interpretarlos. Con ello aumenta el poder garantizar la integridad de los datos.

También se ha visto los elementos necesarios para poder establecer un túnel de comunicación seguro, sus principales características que lo convierten en un sistema seguro, conocer estos aspectos ayudan para detectar principalmente las vulnerabilidades que pueda tener, es muy importante este punto, ya que forman parte, para la realización de adecuadas políticas de seguridad, para la protección de la información.

Los protocolos que se manejaron en este proyecto, forman parte del conjunto de protocolos de seguridad de red, estos como todo protocolo tienen mecanismos para ayudar a la seguridad, pero al utilizarlos también se estará agregando vulnerabilidades propias del protocolo, aunque son mínimas, la utilización de estos protocolos y su continua mejora, ayudan a fortalecer la seguridad, además de estos protocolos existen otros, que será necesario conocer para poder utilizarlos.

Finalmente del resultado de la comparativa se obtiene que el túnel SSH sea mas rápido, la seguridad es buena y se pueden agregar mas recursos de seguridad, estos recursos pueden ser el manejo de un certificado de seguridad, o el uso de claves para el establecimiento de la sesión. Por otro lado el túnel SSL es más lento pero su fortaleza es la seguridad al manejar certificados, por lo tanto si se necesita velocidad se puede utilizar el protocolo SSL.

Referencias.

[1] http://www.eurologic.es/conceptos/conbasics.htm
[2]http://157.92.26.24/w/Apunte_de_Seguridad_en_Redes_(Teor%C3%ADa_de_las_Comunicaciones)
[3] T. Dierks, E. Rescorla, *RFC 4346,* Network Working Group, Abril 2006.
[4] T. Ylonen, C. Lonvick, *RFC 4253,* Network Working Group, Enero 2006.
[5] Hernández Ortiz, Roberto;Peña Blanco, Nancy;Ramírez Amaya, Carlos Mariano; y Rodríguez Baños, Víctor Fabían. *Implementación de un túnel con cifrado para transporte de datos*, INSTITUTO POLITÉCNICO NACIONAL,México,2009.
[6] O'Reilly - SSH the Secure Shell The Definitive Guide, Introduction to SSH
[7] T. Ylonen,C. Lonvick, *RFC 4251,* Network Working Group, Enero 2006.
[8] T. Ylonen,C. Lonvick, *RFC 4253,* Network Working Group, Enero 2006.
[9] T. Ylonen,C. Lonvick, *RFC 4252,* Network Working Group, Enero 2006.
[10] T. Ylonen,C. Lonvick, *RFC 4254,* Network Working Group, Enero 2006.
[11] Barret, Daniel J., Silverman, Richar E.O'Reilly, *SSH, the Secure Shell: The Definitive Guide,* O'Reilly & Associates, Inc.Frebrero 2001.
[12] http://www.putty.org/
[13] http://es.wikipedia.org/wiki/PuTTY
[14] http://openssh.com/es/
[15] http://www.morales-vazquez.com/pdfs/ssl.pdf
[16] T. Dierks, C. Allen, *RFC 2246,* Network Working Group, Enero 1999.
[17] S. Santesson, *RFC 4680,* Network Working Group, Septiembre 2006.
[18] T.Dierks, E. Rescorla, *RFC 4346,* Network Workiing Group, Abril 2006.
[19] RFC 4346
[20] RFC 4346
[21] http://www.cert.org.mx/index.html
[22] http://www.seguridad.unam.mx/descarga.dsc?arch=1564
[23] http://www.cert.org.mx/pdf/boletin/boletin-UNAMCERT2004-004.pdf

Anexos.

UNAM−CERT
Departamento de Seguridad en Cómputo
DGSCA−UNAM
Vulnerabilidad de Seguridad UNAM−CERT−2006−106
Negación de servicio en OpenSSH

Una falla en el demonio de OpenSSH permite a atacantes remotos sin autenticar poder provocar una negación de servicio.

Fecha de Liberación: 27 de Septiembre de 2006
Ultima Revisión: 29 de Septiembre de 2006
Fuente: GLSA 200609−17 / openssh
CVE ID: **CVE−2006−4924**

Sistemas Afectados

OpenSSH ==3
OpenSSH ==4

Riesgo

Alto

Problema de Vulnerabilidad

Remoto

Tipo de Vulnerabilidad

Negación de servicio

I. Descripción
OpenSSH es una suite de aplicaciones para el protocolo SSH, desarrollado y mantenido por el proyecto OpenBSD.
Se ha descubierto una vulnerabilidad que podría permitir una negación de servicio para el protocolo de SSH versión 1 en el código del detector de ataques de compensación CRC.

La vulnerabilidad es causada debido a un error dentro del manejo de múltiples bloques idénticos dentro de un paquete de ssh, si el protocolo ssh es habilitado, esto puede causar una negación de servicio al consumir el procesamiento del CPU, puede ser realizado enviando paquetes específicamente creados.

II. Impacto
Los usuarios que no cuenten con los parches correspondientes son vulnerables a una negación de servicio.

III. Solución
Se recomienda aplicar los parches que corrigen esta vulnerabilidad en las siguientes direcciones.

http://www.openbsd.org/cgi−bin/cvsweb/src/usr.bin/ssh/deattack.c.diff?r1=1.29ate
http://www.openbsd.org/cgi−bin/cvsweb/src/usr.bin/ssh/packet.c.diff?r1=1.143date
http://www.openbsd.org/cgi−bin/cvsweb/src/usr.bin/ssh/deattack.h.diff?r1=1.9ate

IV. Referencias
http://secunia.com/advisories/22091
http://www.gentoo.org/security/en/glsa/glsa−200609−17.xml
http://www.securityfocus.com/bid/20216/info

El Departamento de Seguridad en Cómputo/UNAM−CERT agradece el apoyo en la traducción, elaboración y revisión de éste Documento a:
Alejandro Nuñez Sandoval (anunez at seguridad dot unam dot mx)
Roberto Sanchez Soledad (rsanchez at seguridad dot unam dot mx)

UNAM−CERT
Equipo de Respuesta a Incidentes UNAM
Departamento de Seguridad en Cómputo
E−Mail: seguridad@seguridad.unam.mx
http://www.cert.org.mx
http://www.seguridad.unam.mx
ftp://ftp.seguridad.unam.mx
Tel: 56 22 81 69
Fax: 56 22 80 43

UNAM−CERT
Departamento de Seguridad en Cómputo
DGSCA−UNAM
boletin de Seguridad UNAM−CERT 2004−004
Múltiples vulnerabilidades en OpenSSL

El CERT/UNAM−CERT, a través de sus equipos de respuesta a incidentes de Seguridad en Cómputo, han emitido éste boletín donde informan sobre varias vulnerabilidades en la biblioteca SSL/TLS de OpenSSL que podrían permitir a un intruso remoto, no autenticado, provocar un ataque del tipo Negación de Servicio.
Fecha de Liberación: 30 de Marzo de 2004
Ultima Revisión:
Fuente:
Sistemas Afectados
SSL/TLS OpenSSL

I.Descripción
OpenSSL implemtenta los protocolos Secure Sockets Layer (SSL) y Transport Layer Security (TLS) e incluye una biblioteca criptográfica de propósito general. SSL y TLS son comúnmente usadas para proporcionar servicios de autenticación, encripción, integridad y no repudio para aplicaciones de red como HTTP, IMAP, POP3, STP y LDAP. OpenSSL es ampliamente utilizado entre una diversidad de plataformas y sistemas. En particular, muchos ruteadores y otros tipos de equipo de red usan OpenSSL.

El National Infrastructure Security Co−ordination Centre (NISCC) del Reino Unido y el OpenSSL Project han reportado tres vulnerabilidades en la biblioteca SSL/TLS de OpenSSL (libssl). Cualquier aplicación o sistema que usa esta biblioteca podría ser afectado.

VU#288574 − OpenSSL contiene asignación a un apuntador nulo en la función do_change_cipher_spec()

Las versiones de OpenSSL desde la 0.9.6c a la 0.9.6k y de la 0.9.7a a la 0.9.7c contienen una asignación a un apuntador nulo en la función do_change_cipher_spec(). Realizando un "handshake" SSL/TLS diseñado especialmente, un intruso podría provocar que OpenSSL falle, lo que puede resultar en una negación de servicio en la aplicación destino.

(Otras fuentes: OpenSSL Security Advisory (1.), CAN−2004−0079, NISCC/224012/OpenSSL/1)

VU#484726 − OpenSSL no valida adecuadamente la longitud de los tickets de Kerberos durante el inicio de negociacion de inicio de sesion (handshake) SSL/TLS.

Las versiones 0.9.7a, 0.9.7b y 0.9.7c de OpenSSL no validan adecuadamente la longitud de los tickets de Kerberos durante el inicio de negociacion de inicio de sesion (handshake) SSL/TLS. OpenSSL no está configurado para usar Kerberos de manera predeterminada. Realizando un handshake SSL/TLS especialmente diseñado con un sistema OpenSSL configurado para usar Kerberos, un intruso podría provocar que falle OpenSSL, lo cual puede resultar en una negación de servicio en la aplicación destino. OpenSSL 0.9.6 no es afectado.

(Otras fuentes: OpenSSL Security Advisory (2.), CAN−2004−0112, NISCC/224012/OpenSSL/2)

VU#465542 − OpenSSL no maneja propiamente el tipo de mensajes desconocidos.

La versión anterior a OpenSSL 0.9.6d no maneja apropiadamente los tipos de mensajes desconocidos de SSL/TLS. Un intruso podría causar que la aplicación entre en un loop infinito, el cual puede resultar en una negación de servicio. La versión OpenSSL 0.9.7 no es afectada.

(Otras fuentes: CAN−2004−0081, NISCC/224012/OpenSSL/3)

Impacto

Un intruso remoto no autenticado podría causar una negación de servicio en cualquier aplicación o sistema que utilice las bibliotecas de OpenSSL SSL/TLS.

Soluciones

Actualizar o aplicar el parche correspondiente

Actualizar a OpenSSL 0.9.6m o 0.9.7d. Alternativamente, actualizar o aplicar un parche que especifique su distribuidor. Es importante mencionar que se necesita recompilar las aplicaciones que estén estáticamente ligadas a las bibliotecas de OpenSSL SSL/TLS.

Apendices

Apéndice A. Información de distribuidores

Varios fabricantes son afectados por diferentes combinaciones de estas vulnerabilidades.

Para información reciente, favor de ver la sección de Sistemas Afectados VU#288574, VU#484726 y VU#465542.

Apéndice B. Referencias US−CERT Technical Cyber Security Alert TA04−078A − http://www.us−cert.gov/cas/techalerts/TA04−078A.html

Nota de vulnerabilidad VU#288574 − http://www.kb.cert.org/vuls/id/288574

Nota de vulnerabilidad VU#484726 − http://www.kb.cert.org/vuls/id/484726

Nota de vulnerabilidad VU#465542 − http://www.kb.cert.org/vuls/id/465542

Boletín de seguridad OpenSSL [17 March 2004] − http://www.openssl.org/news/secadv_20040317.txt

Boletín de seguridad NISCC 224012 − http://www.uniras.gov.uk/vuls/2004/224012/index.htm

RFC 2712 Suites adicionales de cifrado de Kerberos a la capa de transporte Security (TLS) − http://www.ietf.org/rfc/rfc2712.txt

Estas vulnerabilidades fueron investigadas y reportadas por el Proyecto OpenSSL y el Centro de Coordinación Nacional de Infraestructura de Seguridad (NISCC).

UNAM−CERT
Equipo de Respuesta a Incidentes UNAM
Departamento de Seguridad en Computo
E−Mail : seguridad@seguridad.unam.mx
http://www.unam−cert.unam.mx
http://www.seguridad.unam.mx
ftp://ftp.seguridad.unam.mx
Tel : 56 22 81 69
Fax : 56 22 80 43