

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Proyecto Fin de Carrera

Servicio VPN de acceso remoto basado en SSL mediante OpenVPN



AUTOR: Juan José Tomás Cánovas
DIRECTOR: Jose María Malgosa Sanahuja

Octubre / 2008



Autor	Juan José Tomás Cánovas
E-mail del Autor	juanjista@gmail.com
Director(es)	Jose María Malgosa Sanahuja
E-mail del Director	josem.malgosa@upct.es
Codirector(es)	-
Título del PFC	Servicio VPN de acceso remoto basado en SSL mediante OpenVPN
Descriptor(es)	Red Privada Virtual, SSL VPN, OpenVPN
Resumen	
<p>Este proyecto evalúa las posibilidades que ofrece la aplicación OpenVPN para construir conexiones seguras a través de la infraestructura de redes públicas, por medio de conexiones seguras mediante el protocolo SSL. En una primera parte, se presenta el concepto de Red Privada Virtual haciendo más hincapié en las redes privadas virtuales que hacen uso del protocolo SSL (Secure Socket Layer). En esta primera parte también se presentarán los adaptadores virtuales TUN/TAP que permiten la creación de interfaces virtuales con el fin de crear túneles punto a punto. La segunda parte se centra en presentar la herramienta OpenVPN y sus principales características. La tercera y cuarta parte se centra en mostrar diversos escenarios sobre los que se utilizará OpenVPN como herramienta para implementar una VPN y para incorporar seguridad a protocolos y servicios no seguros. Por último, se muestran las conclusiones que se obtienen de este estudio.</p>	
Titulación	Ingeniería Técnica de Telecomunicación
Intensificación	Especialidad Telemática
Departamento	Tecnología de la Información y las Comunicaciones
Fecha de Presentación	Octubre - 2008

INDICE DE CONTENIDOS

	<i>Página</i>
Capítulo 1: Introducción	1
1.1.- Resumen de Contenidos y Objetivos	1
1.2.- Las Redes Privadas Virtuales (VPN).....	2
1.2.1.- Introducción a las Redes Privadas Virtuales	3
1.2.2.- Pasado de las VPN.....	4
1.2.3.- Seguridad en VPN	6
1.2.4.- Arquitecturas y Túneles en VPN	7
1.2.5.- Tecnologías VPN en la Actualidad.....	8
1.2.6.- VPN Basadas en SSL/TLS	13
1.2.7.- La alternativa IPSec	25
1.2.8.- VPN SSL/TLS y VPN IPSec	29
Capítulo 2: OpenVPN	34
2.1.- ¿Qué es OpenVPN? Características Principales.....	34
2.2.- OpenVPN y Paquetes Necesarios	40
2.2.1.- Los Archivos de Configuración de OpenVPN	41
2.2.2.- Los controladores virtuales TUN/TAP y VTUN.....	42
2.2.3.- Seguridad con OpenSSL.....	45
2.2.4.- Compresión con LZO	48
2.2.5.- Scripts y Plugins en OpenVPN.....	50
2.2.6.- Autenticación Extra con PAM.....	51
2.3.- Modo de Funcionamiento de OpenVPN.....	52
2.3.1.- IP Tunneling	54
2.3.2.- Ethernet Bridging.....	54
2.4.- Más Acerca del Funcionamiento de OpenVPN.....	55
2.4.1.- Capa de Cifrado de OpenVPN.....	55
2.4.2.- Protocolo.....	57
2.4.3.- Asignación de Direcciones	66
2.4.4.- UDP y TCP	69
Capítulo 3: Comenzando con OpenVPN	73
3.1.- Instalación del paquete OpenVPN.....	73
3.1.1.- Prerrequisitos en la Instalación de OpenVPN	73
3.1.2.- Instalación de OpenVPN Windows	74
3.1.3.- Instalación de OpenVPN en OpenSuSE.....	78

3.1.4.- Instalación de OpenVPN en sistemas RPM.....	81
3.1.5.- Instalación de OpenVPN en Debian	82
3.1.6.- Instalación de OpenVPN mediante Código Fuente	83
3.1.7.- Otros Aspectos a Tener en Cuenta.....	85
3.1.8.- Ejemplo de un Túnel Sencillo.....	85
3.2.- Implementación de la Seguridad	86
3.2.1.- Creación de una Clave Estática	88
3.2.2.- Creación de Clave/Certificado de la CA	89
3.2.3.- Creación de Clave/Certificado del Servidor	90
3.2.4.- Creación de Clave/Certificado del Cliente	92
3.2.5.- Creación de los Parámetros Diffie Hellman	93
3.2.6.- Creación de una CRL.....	94
3.2.7.- Más Técnicas para Proteger la VPN.....	97
3.2.8.- Técnicas Contra Ataques Man-in-the-middle.....	99
3.3.- Otros Métodos de Autenticación	100
3.3.1.- Login y Password Mediante Scripts Para Plugins	101
3.3.2.- Login y Password Mediante Plugins de Objetos Compartidos	101
3.3.3.- Login y Password Sin Certificado de Cliente.....	102
Capítulo 4: Configuración de los Escenarios	103
4.1.- Breve Descripción del Laboratorio.....	103
4.2.- Túnel Sencillo Seguro LAN to LAN	103
4.2.1.- Seguridad Implementada	106
4.2.2.- Archivos de Configuración.....	107
4.2.3.- Resultados y Conclusiones	110
4.3.- Servidor de Túneles Seguro para Configuración Roadwarrior.....	111
4.3.1.- Seguridad Implementada	115
4.3.2.- Archivos de Configuración.....	116
4.3.3.- Resultados y Conclusiones	119
Capítulo 5: Pruebas en Escenarios Reales	122
5.1.- Breve Descripción de los Escenarios.....	122
5.2.- Servidor de Túneles Seguro con Acceso Mediante Login/Password	122
5.2.1.- Seguridad Implementada	123
5.2.2.- Archivos de Configuración.....	126
5.2.3.- Resultados y Conclusiones	129
5.3.- Acceso a Portal Seguro con Acceso Login/Password	131

5.3.1.- Seguridad Incorporada.....	132
5.3.2.- Configuración del Servidor Apache	132
5.3.3.- Archivos de Configuración.....	132
5.3.4.- Resultados y Conclusiones	135
Capítulo 6: Conclusiones y Líneas Futuras.....	137
6.1.- Conclusiones.....	137
6.2.- Líneas Futuras.....	137
Anexos	139
Apéndice I: GUIs y otros Programas	139
Apéndice II: Archivos de Configuración	140
Bibliografía	156
Referencias.....	156
Glosario.....	156

INDICE DE FIGURAS

	<i>Página</i>
Figura 1. Diagrama Lógico de una VPN.....	4
Figura 2. Acceso VPN LAN to LAN.....	9
Figura 3. Acceso VPN de acceso remoto.....	9
Figura 4. Situación de SSL/TLS en la pila de protocolos OSI	14
Figura 5. Arquitectura SSL/TLS	15
Figura 6. Cifrado y Formato de Datos de Aplicación con Record Protocol	17
Figura 7. Calculo del MAC en Record Protocol de SSL v3	17
Figura 8. Calculo del MAC Record Protocol de TLS v1	17
Figura 9. Algoritmos de Cifrado Soportados por SSL/TLS.....	18
Figura 10. Formato del Protocolo Alert de SSL/TLS	18
Figura 11. Mensaje Handshake en SSL/TLS	19
Figura 12. Fases y Mensajes del Protocolo Handshake de SSL/TLS	20
Figura 13. Cálculo de la Clave Maestra (I).....	23
Figura 14. Cálculo de la Clave Maestra (II).....	23
Figura 15. Calculo de otras Claves a partir de la Clave Maestra (I)	23
Figura 16. Calculo de otras Claves a partir de la Clave Maestra (II).....	24
Figura 17. Cálculo de la Clave Maestra en TLS versión 1 (I)	24
Figura 18. Cálculo de la Clave Maestra en TLS versión 1 (II).....	25
Figura 19. Cálculo de la Clave Maestra en TLS versión 1 (III).....	25
Figura 20. Modo Transporte en IPsec	26
Figura 21. Paquete IP en IPsec para Modo Transporte	26
Figura 22. Modo Transporte en IPsec	26
Figura 23. Paquete IP en IPsec para Modo Túnel	27
Figura 24. Formato de Paquetes con Cabecera AH, según el Modo IPsec	28
Figura 25. Formato de Paquetes ESP, según el Modo de IPsec	28
Figura 26. Formato del Paquete Encapsulado por OpenVPN.....	36
Figura 27. Ejemplo de Archivos de Configuración	42
Figura 28. Multiplexación de las Distintas Capas de Seguridad.....	57
Figura 29. Encapsulación del Canal de Datos en OpenVPN	58
Figura 30. Formato de un Paquete de OpenVPN utilizando TCP o UDP	58
Figura 31. Códigos de Operación de los Paquetes de OpenVPN	59
Figura 32. Formato del Paquete de Control de OpenVPN.....	60

Figura 33. Mensaje de OpenVPN para el Intercambio de Claves (Método 1)	61
Figura 34. Mensaje de OpenVPN para el Intercambio de Claves (Método 2)	63
Figura 35. Operación para el Calculo de la Master Secret en OpenVPN	64
Figura 36. Operación para el Cálculo de las 4 Claves en OpenVPN.....	64
Figura 37. Formato del Paquete de Datos OpenVPN	65
Figura 38. Espacio de Direcciones IP para las Redes Privadas según la IANA ...	67
Figura 39. Ejemplo de Uso de TCP sobre TCP	70
Figura 40. Pasos 1 y 2 en la Instalación en Windows XP de OpenVPN	75
Figura 41. Paso 3: Componentes Instalados en Windows XP.....	75
Figura 42. Aplicaciones Instaladas en Windows XP	76
Figura 43. Paso 4: Directorio de Instalación de OpenVPN en Windows XP	76
Figura 44. Pasos 5 y 6: Progreso y Finalización de la Instalación de OpenVPN .	76
Figura 45. Icono del Proceso de la GUI Instalada en Windows XP	77
Figura 46. Interfaces de Red tras Instalar OpenVPN en Windows XP.....	77
Figura 47. Dirección IP y MAC de la Interfaz Virtual en Windows XP	78
Figura 48. Estado de Conexión OpenVPN utilizando la GUI en Windows XP ...	78
Figura 49. Aspecto Gráfico del Menú Principal de YaST en SuSE	79
Figura 50. Instalador/Desinstalador de Paquetes de YaST en SuSE	80
Figura 51. Progreso de la Instalación de OpenVPN en YaST de SuSE (I).....	80
Figura 52. Progreso de la Instalación de OpenVPN en YaST de SuSE (II)	81
Figura 53. Fin de la Instalación de OpenVPN en YaST de SuSE	81
Figura 54. Archivos Instalados de OpenVPN y su Función en Sistemas RPM....	82
Figura 55. Instalación de OpenVPN mediante APT en un Sistema Debian	83
Figura 56. Archivos Instalados de OpenVPN y su Función en Sistemas Debian.	83
Figura 57. Instalación de OpenVPN mediante el Código Fuente (I).....	84
Figura 58. Instalación de OpenVPN mediante el Código Fuente (II).....	84
Figura 59. Instalación de OpenVPN mediante el Código Fuente (III)	84
Figura 60. Archivos de Configuración de un Túnel Sencillo con OpenVPN	85
Figura 61. Archivos de Configuración de un Túnel Sencillo con OpenVPN	86
Figura 62. Conectividad en un Túnel Sencillo con OpenVPN	87
Figura 63. Conectividad en un Túnel Sencillo en Windows.....	87
Figura 64. Programa para Generar una Clave Estática en Windows	89
Figura 65. Comandos para Crear el Par Certificado/Clave de la CA	90
Figura 66. Creación del Par Certificado/Clave de la CA en Linux.....	90
Figura 67. Comandos para Crear el Par Certificado/Clave del Servidor	91

Figura 68. Creación del Par Certificado/Clave del Servidor en Linux (I)	91
Figura 69. Creación del Par Certificado/Clave del Servidor en Linux (II).....	92
Figura 70. Comandos para Crear el Par Certificado/Clave del Cliente	92
Figura 71. Creación del Par Certificado/Clave del Cliente en Linux (I)	93
Figura 72. Creación del Par Certificado/Clave del Cliente en Linux (II)	94
Figura 73. Comandos para Crear los Parámetros Diffie Hellman	94
Figura 74. Creación de los Parámetros Diffie Hellman.....	95
Figura 75. Comandos para Crear los Parámetros Diffie Hellman	95
Figura 76. Creación de una CRL en Linux	95
Figura 77. Revocación de un Certificado en el lado del Servidor por la CRL	96
Figura 78. Revocación de un Certificado en el lado del Cliente por la CRL.....	96
Figura 79. Switch Utilizado para las Pruebas en el Laboratorio.....	103
Figura 80. Esquema del Escenario 1	104
Figura 81. Esquema del Escenario 2	112
Figura 82. Esquema del Escenario Real 1.....	123
Figura 83. Esquema del Escenario Real 2.....	131

Capítulo 1: Introducción

1.1.- Resumen de Contenidos y Objetivos

A día de hoy, las comunicaciones a través de las redes de información resultan de vital importancia para un gran número de empresas y organizaciones. Para llegar a su destino, ese tráfico debe atravesar, muy a menudo, una infraestructura de redes públicas (como Internet), lo que lo hace vulnerable a los ataques de usuarios mal intencionados. Ante ese peligro potencial, resulta imprescindible poseer herramientas que permitan proteger el contenido de dicho tráfico, para asegurar tanto su privacidad como su integridad, en las comunicaciones extremo a extremo.

La solución más evidente consiste en montar redes privadas, es decir, para el uso exclusivo de los propietarios de la red, garantizándose la seguridad en las comunicaciones. Sin embargo, esta política de seguridad se está abandonando por dos motivos de peso: el coste y la baja escalabilidad. En efecto, las grandes distancias que separan, en ocasiones, las filiales de una organización, hacen que resulte demasiado cara la construcción de enlaces privados. Por otra parte, el uso de redes privadas supone la implantación de un nuevo enlace cada vez que se quiera unir un nuevo miembro a la red de una organización, con los consiguientes problemas de tiempo y coste.

Como solución más eficiente, aparecen las Redes Privadas Virtuales (VPN), un sistema para construir conexiones seguras a través de la infraestructura de redes públicas, tanto para enlaces punto a punto, como para conectar distintas redes locales entre sí o permitir a un tele-trabajador conectarse a la sede de su empresa desde cualquier lugar con acceso a Internet. Este sistema permite aprovechar la infraestructura de comunicaciones existente, aportando los elementos de seguridad necesarios para evitar cualquier intrusión en el contenido del tráfico que protegen. Se consigue así un método de comunicación segura que combina un bajo coste con unos altos niveles de privacidad.

La implementación de VPN suele estar basada en “túneles”, donde la información que atraviesa las redes públicas se encapsula en paquetes de un protocolo (normalmente IP), de forma que el contenido resulta invisible hasta llegar a su destino, donde se desencapsula el paquete. Existen varias tecnologías con las que implementar túneles y a su vez implementar VPN tales como GRE, PPTP, L2TP, IPSEC, SSL/TLS,... Casi todas estas tecnologías pueden implementarse en multitud de medios físicos basados en redes IP tales como X.25, Frame Relay, o tecnología ATM y encapsular paquetes de otros protocolos de red no IP. De estas múltiples arquitecturas para implementación de VPN nos centraremos en el estudio de las VPN basadas en SSL/TLS mediante la herramienta multiplataforma OpenVPN.

Las VPN basadas en SSL/TLS ofrecen confidencialidad en la comunicación por medio del cifrado simétrico, permite el uso de certificados digitales para la autenticación e

intercambio de claves asimétricas y nos brinda integridad en los datos enviados mediante el uso de funciones Hash y Códigos de Autenticación de Mensajes (MAC).

Para realizar este estudio de VPN basada en SSL/TLS se ha elegido la herramienta OpenVPN, una implementación libre multiplataforma para la implementación de este tipo de VPN e implementación de los mecanismos de seguridad.

Este documento consta de 6 capítulos; el capítulo 1 presenta los conceptos teóricos relacionados con las VPN, la seguridad en éstas y las diferentes tecnologías utilizadas en VPN, centrándose sobre todo en la arquitectura SSL/TLS para la implementación de VPN. En concreto, se describe el concepto de VPN basada en SSL/TLS, así como todos los protocolos y algoritmos de seguridad que utiliza SSL/TLS.

En los capítulos 2 y 3 se presentarán las características del paquete OpenVPN. En concreto se presentarán las ventajas de utilizar esta herramienta, los paquetes adicionales de los que depende OpenVPN, los modos de funcionamiento, la manera de implementar los túneles con los adaptadores virtuales TUN/TAP, la manera de implementar los mecanismos de seguridad con OpenSSL, reparto de direcciones IP,...

En el capítulo 4 se implementarán, en el laboratorio, los escenarios típicos para la implementación de una VPN basada en la arquitectura SSL/TLS con OpenVPN, como por ejemplo, escenarios LAN to LAN (para comunicación entre sucursales de empresa, por ejemplo) o escenarios de acceso remoto con clientes Road Warrior (o teletrabajadores). Al final de cada escenario se hará una reflexión sobre los resultados y experiencias obtenidas.

Al final (capítulo 5) se implementarán, con el uso de OpenVPN y otras aplicaciones específicas y plugins, escenarios más reales para añadir seguridad a ciertos protocolos no seguros pero muy extendidos a nivel de usuario. Como en el capítulo anterior se finalizará este capítulo con una breve reflexión sobre los resultados y experiencias obtenidas.

Por último, en el capítulo cinco se expondrán las conclusiones generales que se extraen de este proyecto fin de carrera, resumiendo los conceptos más importantes estudiados, y destacando posibles futuras ampliaciones a este proyecto.

1.2.- Las Redes Privadas Virtuales (VPN)

Este punto se centrará en la presentación de las redes privadas virtuales o VPN como una solución en la política de seguridad económica y de alta escalabilidad. Seguidamente y de manera breve se describirá como surgió esta tecnología. Se describirán las diferentes arquitecturas utilizadas en las redes privadas virtuales y también las diferentes tecnologías que existen en la actualidad. Este apartado sobre las tecnologías presentes en la actualidad para implementar redes privadas virtuales se

centrará en dos tecnologías muy utilizadas en la actualidad: IPSec y las redes privadas virtuales basadas en el protocolo SSL del cual trata este proyecto.

1.2.1.- Introducción a las Redes Privadas Virtuales

Una Red Privada Virtual (en adelante, por comodidad VPN – del inglés Virtual Private Network) es una red, construida sobre la infraestructura de una red pública, que permite a usuarios remotos comunicarse de forma transparente y segura. Tal y como indica su nombre, es privada dado que garantiza la privacidad en sus comunicaciones, y virtual porque se implementa sobre la infraestructura de redes públicas (por ejemplo, Internet).

Las conexiones VPN permiten a los usuarios acceder, desde casa o desde un punto remoto, al servidor de su organización a través de la infraestructura de encaminamiento que proveen las redes públicas. Desde el punto de vista del usuario, la conexión VPN es una conexión punto a punto entre su ordenador y el servidor de la organización. La naturaleza de las redes intermedias es irrelevante para el usuario, ya que, para él, los datos son enviados como si hubiera un enlace dedicado hasta el servidor. La tecnología VPN también permite a las empresas conectar con sus filiales, o con otras empresas, a través de la infraestructura de redes públicas, ofreciendo comunicaciones seguras. También en este caso, la conexión aparece al usuario como si se estableciera en una red privada.

Por tanto, las redes privadas virtuales intentan dar soluciones principalmente a problemas tales como:

- El acceso remoto a la red de la empresa.
- La interconexión de múltiples sitios remotos.
- Establecimiento de conexiones seguras.

Para aportar a sus empleados la posibilidad de conectar con sus servidores, una organización debe desplegar una solución escalable de acceso remoto. Normalmente, tienen dos opciones: una solución es crear un departamento de sistemas de información, que se encargue de comprar, instalar y mantener la infraestructura de redes privadas. La otra opción es contratar los servicios de una empresa para que resuelva estos problemas. En cualquiera de las dos soluciones existe el inconveniente de que el soporte a usuarios móviles, es decir, los que intentan conectarse desde cualquier ubicación, es muy reducido, no siendo así si optamos por la solución de red pública. Además, ninguna de estas soluciones aporta la escalabilidad suficiente, en términos de costes de despliegue y flexibilidad de administración. Resulta pues interesante reemplazar la infraestructura de red privada por una solución de menor coste, basada en la tecnología Internet. Mediante esta solución, unas pocas conexiones a través de proveedores de servicios de Internet y servidores VPN pueden satisfacer las necesidades de enlaces remotos de cientos de clientes remotos o empresas alejadas entre sí.

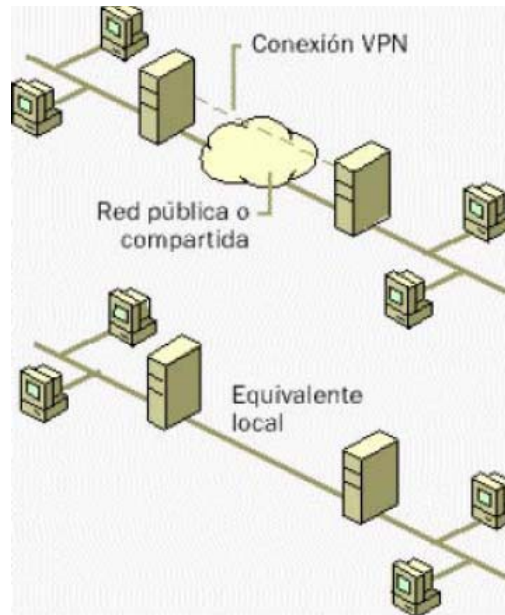


Figura 1. Diagrama Lógico de una VPN

Aunque las VPN presentan ventajas tales como el ahorro de costes, la flexibilidad y la escalabilidad, la implementación de una VPN acompaña consigo los siguientes problemas:

- **Seguridad:** las VPN requieren con mucha frecuencia de un gran esfuerzo de gestión relativa a la seguridad ya que requieren de la gestión de claves, conexiones, control de acceso,...
- **Compatibilidad:** la mayoría de los protocolos que se utilizan para crear túneles y dotar de seguridad a la VPN no son compatibles entre ellos por ahora, por lo que es complicado seleccionar un único protocolo que satisfaga todas las necesidades.
- **Fiabilidad y Rendimiento:** las VPN sobre Internet dependen de la infraestructura pública de Internet y experimentan los mismos problemas que ésta. Además, a menudo las organizaciones necesitan que se les garantice una calidad de servicio (QoS).
- **Cuellos de Botella Potenciales:** el cifrado y descifrado o el encapsulado y desencapsulado de paquetes son acciones que requieren una gran capacidad de procesado y que hacen que aumenten el tamaño de los paquetes.

1.2.2.- Pasado de las VPN

Como ya se ha comentado, una de las necesidades vitales de la empresa moderna es la posibilidad de compartir información, particularmente para aquellas empresas que se encuentran dispersas, con sedes en diferentes zonas y unidades de negocio que no se encuentran en el mismo entorno físico. Hasta el momento, las grandes corporaciones habían solucionado el problema mediante sistemas de comunicación como líneas punto a punto y sofisticadas instalaciones de interconexión. Aunque efectivas, estas soluciones

quedaban fuera del alcance de empresas de menor tamaño y con recursos económicos y técnicos más escasos.

Los años setenta fueron testigos del nacimiento de las redes privadas (redes privadas no virtuales), que permitían a las empresas interconectar sus sedes principales mediante líneas alquiladas independientes para voz y datos, con anchos de banda fijos. Pero la demanda de transmisión de datos a mayores velocidades y rendimientos tan sólo surgió en la década de los 80, propiciada por el cambio del modelo informático basado en mainframe a arquitecturas cliente/servidor, así como el desarrollo de nuevas aplicaciones para estos nuevos entornos.

Aparecieron así nuevos patrones de tráfico, donde el ancho de banda podía permanecer ocioso durante prolongados períodos de tiempo. También en esta época comenzó a detectarse la necesidad de interconectar las distintas redes de área local (LAN) que empezaban a surgir en las organizaciones.

Poco a poco se le asignó el concepto de Intranet a la interconexión de las sedes principales de una empresa mediante líneas alquiladas gastando enormes recursos a fin de configurar redes privadas de alta complejidad. Estas redes eran instaladas usando costosos servicios de líneas dedicadas, Frame Relay y ATM para incorporar usuarios remotos. Como ya se ha comentado con anterioridad, las empresas medianas y pequeñas no podían adquirir servicios tan costosos, y se encontraban condenadas a utilizar servicios muy inferiores.

A finales de los 80 y primeros 90, Frame Relay comenzó a ganar una creciente aceptación, dado que ofrecía una más elevada capacidad de procesamiento de datos que X.25, su tecnología predecesora. Estas mejoras se consiguieron mediante la implementación de un sistema de procesamiento de paquetes simplificado que dividía la información en tramas, cada una de las cuales transportaba una dirección utilizada por los conmutadores para determinar su destino final. El sistema aumentaba la eficiencia en la utilización de los recursos, dado que permitía fragmentar el tráfico en ráfagas y así aprovechar el ancho de banda que antes permanecía ocioso, reduciendo significativamente además los costes de transmisión frente a los de las líneas alquiladas.

Mediados los noventa, la dependencia de las empresas respecto de sus redes aumentó aún más con la creciente utilización del correo electrónico y la implantación de aplicaciones consumidores de grandes anchos de banda. Esta dependencia no ha hecho desde entonces más que crecer ante la expansión cada vez más generalizada del uso de Internet y de las transacciones de comercio electrónico.

Esta expansión, junto a la aparición del concepto de convergencia de las redes de voz y datos en una sola plataforma de networking, condujo al desarrollo de ATM (Asynchronous Transfer Mode), que fue concebida como una tecnología multiservicio de banda ancha capaz de soportar voz, datos, imágenes y videos sobre una misma infraestructura de red.

A diferencia de Frame Relay, en la que el tamaño de los paquetes era variable, ATM se basa en conmutación de celdas de tamaño fijo. Esta característica permitía aprovechar todas las ventajas de la multiplexación estadística y ofrecía un rendimiento determinístico. Las conexiones ATM están típicamente basadas en circuitos virtuales permanentes (PVC) con calidad de servicio (QoS), capaces de soportar transmisiones de extremo a extremo garantizadas y fiables.

1.2.3.- Seguridad en VPN

La privacidad se consigue mediante dos mecanismos: la autenticación, que asegura que sólo los usuarios autorizados accedan a un determinado servicio, y el cifrado de datos, que codifica la información enviada de modo que sólo el/los usuario(s) destino puedan descifrarla y por tanto comprenderla. A éstos dos hay que sumar otro aspecto fundamental en la construcción de una VPN: la integridad de los datos, garantía de que la información no pueda ser modificada en su camino hasta el receptor.

Para emular un enlace punto a punto, los datos son encapsulados, y se les añade una cabecera que provee información sobre cómo atravesar las distintas redes para alcanzar su destino.

Comúnmente estos enlaces privados virtuales son denominados “túneles”, ya que el transporte de datos se realiza sin que nadie pueda ver qué es lo que se transmite. Para construir estos túneles entre los usuarios, éstos deben ponerse de acuerdo sobre el esquema de cifrado y autenticación que implementará la VPN.

Para conseguir conexiones seguras, existen unos requisitos que permiten garantizar privacidad, integridad de la información y gestión de las conexiones. Poner en marcha todos estos mecanismos es un proceso complejo, que exige el uso simultáneo de varias tecnologías. Debe permitirse el acceso de clientes remotos a recursos de redes locales, y las oficinas remotas deben de poder compartir información y recursos. Para asegurar la privacidad y la integridad de los datos, son también necesarias tecnologías de autenticación, cifrado y autorización. La solución VPN deberá aportar:

- **Autenticación del usuario:** debe verificarse la identidad de los clientes de la VPN, permitiendo el acceso únicamente a los usuarios autorizados. En ocasiones se crean unos registros que muestren quién se conectó y por cuánto tiempo.
- **Gestión de direcciones:** deben asignarse direcciones en la red local a los clientes VPN, asegurándose de que no sean visibles fuera de dicha red. También debe darse cierta información a los clientes para que puedan acceder a los recursos protegidos de la red.
- **Cifrado de datos:** los datos que atraviesan las redes públicas deben ser ilegibles para todos, excepto los clientes VPN y su servidor. Esto se consigue mediante tecnologías de cifrado de la información.
- **Gestión de claves:** para cifrar los datos, la solución VPN debe utilizar algún mecanismo de cifrado (basado en claves) que permita crear el túnel. Deben por

tanto generarse, y renovarse cada cierto tiempo, las claves de cifrado, de modo que se mantengan la seguridad y la privacidad en las conexiones VPN.

Así, las VPN constituyen una estupenda combinación entre la seguridad y garantía que ofrecen las costosas redes privadas y el gran alcance, lo asequible y lo escalable del acceso a través de Internet. Esta combinación hace de las Redes Privadas Virtuales una infraestructura confiable y de bajo costo que satisface las necesidades de comunicación de cualquier organización.

1.2.4.- Arquitecturas y Túneles en VPN

Para poder empezar a explicar con más detalle las características de las VPN y poder nombrar las diferentes arquitecturas que puede tomar una VPN se tendrá que definir el término túnel en una VPN.

Muchos de los protocolos utilizados hoy en día para transferir datos de una máquina a otra a través de la red carecen de algún tipo de cifrado o medio de seguridad que evite que nuestras comunicaciones puedan ser interceptadas y espiadas. HTTP, FTP, POP3 y otros muchos protocolos ampliamente usados, utilizan comunicaciones que viajan en claro a través de la red. Esto supone un grave problema, en todas aquellas situaciones en las que queremos transferir entre máquinas información sensible, como pueda ser una cuenta de usuario (nombre de usuario y contraseña), y no tengamos un control absoluto sobre la red, a fin de evitar que alguien pueda interceptar nuestra comunicación por medio de la técnica del hombre en el medio (man in the middle), como es el caso de Internet.

El problema de los protocolos que envían sus datos en claro, es decir, sin cifrarlos, es que cualquier persona que tenga acceso físico a la red en la que se sitúan las máquinas puede ver dichos datos. De este modo, alguien que conecte su máquina a una red y utilice un sniffer recibirá y podrá analizar por tanto todos los paquetes que circulen por dicha red. Si alguno de esos paquetes pertenece a un protocolo que envía sus comunicaciones en claro, y contiene información sensible, dicha información se verá comprometida. Si por el contrario, se cifran las comunicaciones con un sistema que permita entenderse sólo a las dos máquinas que son partícipes de la comunicación, cualquiera que intercepte desde una tercera máquina los paquetes, no podrá hacer nada con ellos, al no poder descifrar los datos.

Una forma de evitar este problema, sin dejar por ello de utilizar todos aquellos protocolos que carezcan de medios de cifrado, es usar una técnica llamada tunneling. Básicamente, esta técnica consiste en abrir conexiones entre dos máquinas por medio de un protocolo seguro, como puede ser SSL (Secure Socket Layer), a través de las cuales realizaremos las transferencias inseguras, que pasarán de este modo a ser seguras. El paquete suele encapsularse (ya cifrado) y se le suele cambiar la cabecera, creando un túnel lógico entre fuente y destino. De esta analogía viene el nombre de la técnica, siendo la conexión segura (en este caso de SSL) el túnel por el cual se envían los datos

para que nadie más aparte de los participantes que se sitúan a cada extremo del túnel, pueda ver dichos datos.

Otra característica de los túneles es que en la mayoría de las tecnologías utilizadas para la creación de estos túneles el espacio de direcciones IP utilizado por la VPN es completamente independiente del espacio de direcciones utilizado por la infraestructura de red subyacente. Por tanto, el enrutamiento de ambos espacios de direcciones se aísla igualmente, es decir, los mecanismos de enrutamiento de la VPN pueden ser diferentes de los de la red subyacente.

Tras una pequeña definición del término tunneling se presentará las diferentes arquitecturas de las que puede partir una implementación VPN. Básicamente existen dos arquitecturas de conexión para las VPN:

- **VPN extremo a extremo:** también conocida como LAN to LAN, este esquema se utiliza para conectar oficinas remotas con la sede central de la organización. El servidor VPN, normalmente un router o un hardware diseñado específicamente para hacer esta función, que posee un vínculo permanente a Internet, acepta las conexiones vía Internet provenientes de los sitios y establece el túnel VPN. Los servidores de las sucursales se conectan a Internet utilizando los servicios de su proveedor local de Internet, típicamente mediante conexiones de banda ancha. Como se ha explicado con anterioridad, esta arquitectura permite eliminar los costosos vínculos tradicionales punto a punto entre sedes, sobre todo en las comunicaciones internacionales.
- **VPN de acceso remoto:** es quizás el modelo más usado actualmente y consiste en usuarios o proveedores que se conectan con la empresa desde sitios remotos (oficinas comerciales, domicilios, hoteles, aviones preparados, ciber-cafés,...) utilizando un acceso a Internet como vínculo de acceso. El cliente, de un acceso a Internet se autentica al servidor de acceso remoto, y el servidor, desde su conexión a Internet, se autentica ante el cliente. Una vez autenticados, los clientes tienen un nivel de acceso muy similar al que tienen en la red local de la empresa.

1.2.5.- Tecnologías VPN en la Actualidad

En la actualidad existen varias tecnologías que permiten implementar una VPN, implementando con técnicas diferentes los túneles y la seguridad que ofrecerá esta VPN. Algunas tecnologías no están abiertas a los desarrolladores porque son propietarias, como pueden ser muchas de las soluciones VPN que Cisco incorpora en sus dispositivos. Como tecnologías VPN más utilizadas en la actualidad se describirán, sin entrar en mucho detalle, PPTP, L2F, L2TP, MPLS, GRE, SSH e IPsec y se describirá con más detalle el protocolo SSL como solución para implementar una VPN. Posteriormente, en otro punto, se hará una comparación entre las tecnologías SSL e IPsec para VPN.

ACCESO VPN LAN TO LAN

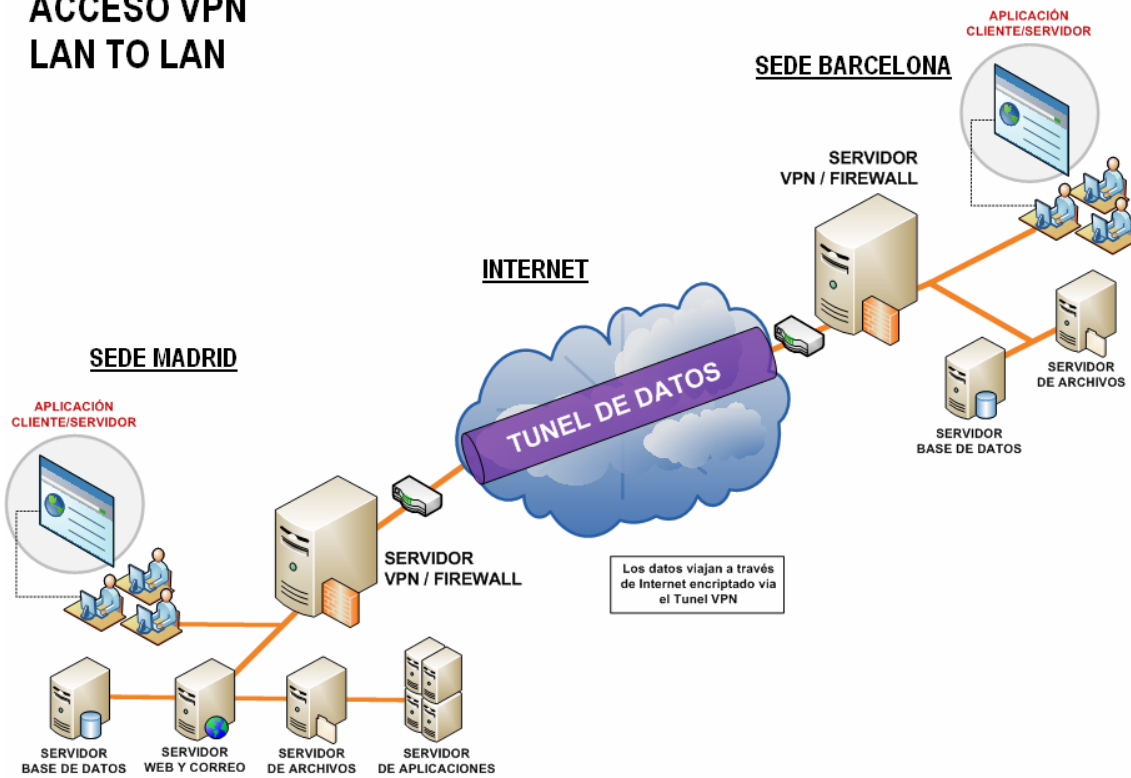


Figura 2. Acceso VPN LAN to LAN

ACCESO VPN DE ACCESO REMOTO

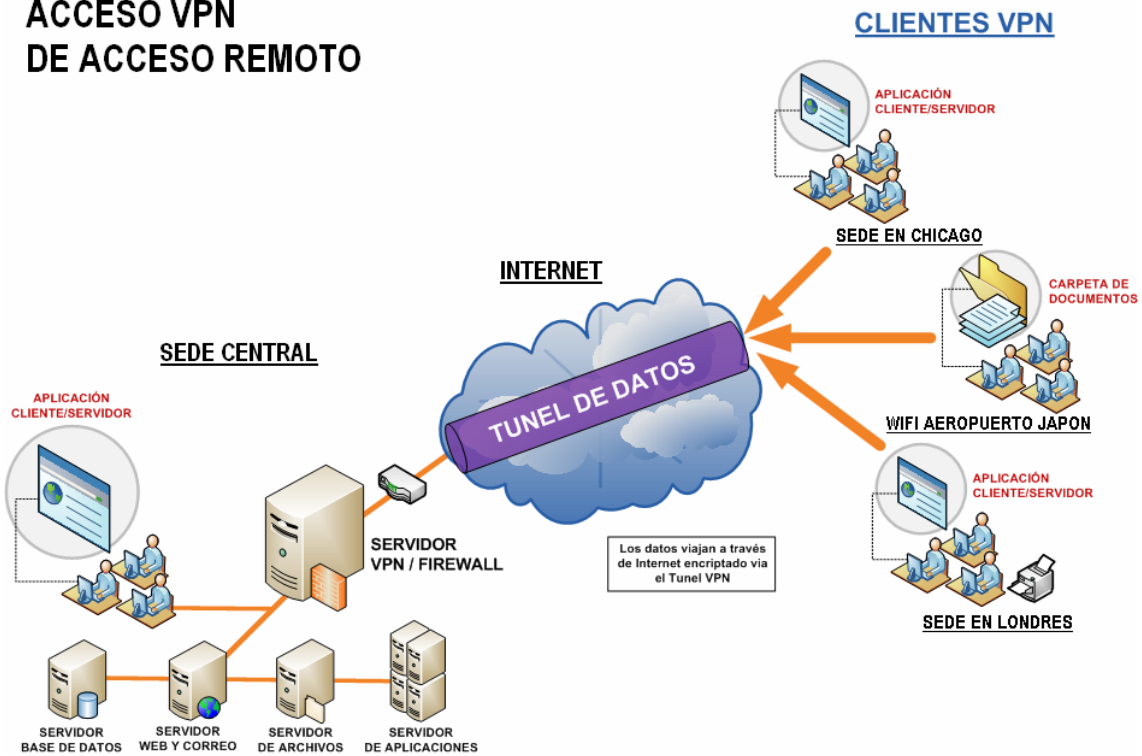


Figura 3. Acceso VPN de acceso remoto

- **GRE (Generic Routing Encapsulation):** es un protocolo diseñado por Cisco para el establecimiento de túneles a través de Internet e implementado en la mayoría de sus productos. GRE soporta hasta 20 protocolos de red pasajeros como por ejemplo, IPv4, IPv6, Novell IPX, etc.

Hay que destacar que el protocolo GRE está diseñado específicamente para establecer túneles y, por lo tanto, tendrá que acompañar otros estándares para conseguir la seguridad necesaria. Aunque en la práctica se podrían utilizar únicamente túneles GRE, se suelen acompañar con otros mecanismos de autenticación y cifrado para conseguir la seguridad exigida, siendo el mecanismo más común IPSec.

El protocolo GRE está definido en la RFC 1701 y RFC 1702 y se han encontrado vulnerabilidades en los productos de Cisco que implementan este mecanismo de creación de túneles por la posibilidad de burlar las listas de control de acceso (ACL) mediante el envío de una serie de paquetes GRE especialmente diseñados.

- **PPTP (Point-to-Point Tunneling Protocol):** es un protocolo propietario de Microsoft diseñado específicamente para implementar VPN. PPTP depende del protocolo de enlace de datos PPP (Point-to-Point-Protocol). En cuanto a seguridad, la autenticación de usuarios se realiza a través de los protocolos PAP (Password Authentication Protocol) y CHAP (Challenge Handshake Authentication Protocol), EAP (Extensible Authentication Protocol) y MS-CHAP (versión de Microsoft de CHAP). PPTP utiliza túneles GRE para implementar el túnel de una VPN, encapsulando tramas PPP y utiliza un cifrado RC-4 bastante débil. Otro inconveniente es que solo permite una conexión por túnel.

La especificación para el protocolo PPTP fue publicada por el RFC 2637, aunque no ha sido ratificada como estándar por el IETF (Internet Engineering Task Force).

La seguridad de PPTP está totalmente rota y los sistemas con PPTP deberían ser sustituidos por otra tecnología de VPN. Existen incluso herramientas para romper las claves de las sesiones PPTP y descifrar el tráfico de una VPN. Por ejemplo, la herramienta ASLEAP permite obtener claves durante el establecimiento de la conexión, en el que se utiliza el protocolo LEAP, debido al mal diseño de éste protocolo.

- **L2F (Layer 2 Forwarding):** como PPTP, L2F fue diseñado, por Cisco, para establecer túneles de tráfico desde usuarios remotos hasta sus sedes corporativas. La especificación para el protocolo L2F fue publicada por el RFC 2341. La principal diferencia entre PPTP y L2F es que, como el establecimiento de

túneles de L2F no depende de IP, es capaz de trabajar directamente con otros medios, como Frame Relay o ATM.

Utiliza el protocolo PPP para la autenticación entre usuarios remotos por lo que implementa los protocolos de autenticación PAP y CHAP. Pero L2F también implementa otras técnicas de autenticación como TACACS+ y RADIUS. Una característica que difiere L2F de PPTP es que L2F permite más de una conexión por túnel.

En L2F se utilizan dos niveles de autenticación, primero por parte del ISP (proveedor de servicio de red), anterior al establecimiento del túnel, y posteriormente, cuando se ha establecido la conexión con la pasarela corporativa. Como L2F es un protocolo de nivel de enlace de datos según el modelo de referencia OSI, ofrece a los usuarios la misma flexibilidad que PPTP para manejar protocolos distintos a IP, como IPX.

- **L2TP (Layer 2 Tunneling Protocol):** éste es un protocolo normalizado por la IETF como RFC 2661 y cabe destacar que presenta las mejores características de los protocolos para implementar túneles L2F y PPTP. Se trata de un estándar abierto y disponible en la mayoría de las plataformas (Windows, UNIX,...).

Al igual que PPTP, L2TP utiliza el protocolo PPP para proporcionar una envoltura inicial de los datos y luego incluir los encabezados adicionales a fin de transportarlos a través de la red. L2TP puede transportar una gran variedad de protocolos y, además, es capaz de trabajar directamente con otros medios, como X.25, Frame Relay o ATM.

L2TP puede usar certificados de seguridad de clave pública para cifrar los datos y garantizar la autenticidad de los usuarios de la VPN. Para potenciar el cifrado y la autenticación, L2TP suele implementarse junto con IPSec, otra tecnología VPN que se explicará en este apartado. Otra característica que hay que destacar es que permite establecer múltiples túneles entre dos puntos finales pudiendo proporcionar diferentes calidades de servicio (QoS).

L2TP no presenta unas características criptográficas especialmente robustas si no es implementado junto a IPSec ya que sólo se realiza la operación de autenticación entre los puntos finales del túnel, pero no para cada uno de los paquetes que viajan por él. Esto puede dar lugar a suplantaciones de identidad en algún punto interior al túnel. Además, sin comprobación de la integridad de cada paquete, sería posible realizar un ataque de denegación del servicio (DoS) por medio de mensajes falsos de control que den por acabado el túnel L2TP o la conexión PPP subyacente. Por otra parte, L2TP no cifra en principio el tráfico de datos de usuario, lo cual puede dar problemas cuando sea importante mantener la confidencialidad de los datos. Por último, a pesar de que la información contenida en los paquetes PPP puede ser cifrada, este protocolo no dispone de

mecanismos para generación automática de claves, o refresco automático de claves. Esto puede hacer que alguien que escuche en la red y descubra una única clave tenga acceso a todos los datos transmitidos.

- **MPLS (MultiProtocol Label Switching):** MPLS es un mecanismo de transporte de datos estándar creado por la IETF y definido en el RFC 3031. Opera entre la capa de enlace de datos y la capa de red del modelo OSI. Fue diseñado para unificar el servicio de transporte de datos para las redes basadas en circuitos y las basadas en paquetes. Puede ser utilizado para transportar diferentes tipos de tráfico, incluyendo tráfico de voz y de paquetes IP.

Una de las tantas aplicaciones de MPLS es la de implementar VPN utilizando MPLS. Las VPN basadas en MPLS son muy flexibles en cuanto a la gestión y provisión del servicio y en cuanto al crecimiento cuando se requieren añadir nuevas sedes o usuarios. Esto es así porque con una arquitectura MPLS no se superpone el modelo topológico sino que se acopla a la red del proveedor ya existente. Por tanto, lo que hay son conexiones IP a una "nube común" en las que solamente pueden entrar los miembros de la misma VPN. Las "nubes" que representan las distintas VPNs se implementan mediante los caminos LSP (Label Switched Path) creados por el mecanismo de intercambio de etiquetas MPLS. Los LSPs son similares a los túneles en cuanto a que la red transporta los paquetes del usuario (incluyendo las cabeceras) sin examinar el contenido, a base de encapsularlos sobre otro protocolo. Aquí está la diferencia: en los túneles se utiliza el encaminamiento convencional IP para transportar la información del usuario, mientras que en MPLS esta información se transporta sobre el mecanismo de intercambio de etiquetas, que no ve para nada el proceso de routing IP. Sin embargo, sí se mantiene en todo momento la visibilidad IP hacia el usuario, que no sabe nada de rutas MPLS sino que ve un Internet privado (intranet) entre los miembros de su VPN. De este modo, se pueden aplicar técnicas QoS basadas en el examen de la cabecera IP, que la red MPLS podrá propagar hasta el destino, pudiendo así reservar ancho de banda, priorizar aplicaciones, establecer CoS (diferentes clases de servicio – Class of Service) y optimizar los recursos de la red con técnicas de ingeniería de tráfico.

La diferencia entre los túneles convencionales y los "túneles MPLS" (realmente son LSP – Label Switched Paths) está en que éstos se crean dentro de la red, a base de LSPs, y no de extremo a extremo a través de la red. Por el contrario, MPLS no fue diseñado como una arquitectura segura, por lo que habría que añadir otros métodos o técnicas de cifrado y autenticación para implementar una VPN basada en MPLS segura.

- **SSH (Secure SHell):** éste es un protocolo normalizado por la IETF como RFC 2451 y RFC 2452. SSH se basa en TELNET pero le añade la seguridad que TELNET carece.

El protocolo SSH se utiliza para tunelizar tráfico confidencial sobre Internet de una manera segura. Por ejemplo, un servidor de ficheros puede compartir archivos usando el protocolo SMB (Server Message Block), cuyos datos no viajan cifrados. Esto permitiría que una tercera parte, que tuviera acceso a la conexión (algo posible si las comunicaciones se realizan en Internet) pudiera examinar a conciencia el contenido de cada fichero transmitido. Para poder montar el sistema de archivo de forma segura, se establece una conexión mediante un túnel SSH que encamina todo el tráfico SMB al servidor de archivos dentro de una conexión cifrada SSH. Aunque el protocolo SMB sigue siendo inseguro, al viajar dentro de una conexión cifrada se impide el acceso al mismo. Por ejemplo, para conectar con un servidor Web de forma segura, utilizando SSH, haríamos que el cliente Web, en vez de conectarse al servidor directamente, se conecte a un cliente SSH. El cliente SSH se conectaría con el servidor tunelizado, el cual a su vez se conectaría con el servidor Web final. Lo atractivo de este sistema es que hemos añadido una capa de cifrado sin necesidad de alterar ni el cliente ni el servidor Web.

Además de la conexión a otras máquinas, SSH nos permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

1.2.6.- VPN Basadas en SSL/TLS

Los protocolos SSL (Secure Socket Layer) y TLS (Transport Layer Security) son protocolos de la capa de transporte que proporcionan comunicaciones seguras en Internet. Los protocolos SSL versión 3.0 y TLS versión 1.0 son idénticos, una de las diferencias, son sus diseñadores. SSL fue diseñado por Netscape en 1996 y, aunque no es un protocolo estandarizado por el IETF, éste lo estandarizó en 1999 con ligeras modificaciones, aunque el protocolo funcionaba de la misma manera. La primera definición de TLS apareció en el RFC 2246, aunque se han ido publicando otras definiciones relacionadas con la compatibilidad de TLS con otros protocolos y técnicas criptográficas. SSL/TLS permite la autenticación tanto de cliente como servidor, usando claves públicas y certificados digitales y proporciona comunicación segura mediante el cifrado de la información entre emisor y receptor. SSL/TLS funciona por encima del protocolo de transporte (normalmente TCP) y por debajo de los protocolos de aplicación. Este protocolo está muy extendido para realizar actividades de comercio electrónico de tal manera que Visa, MasterCard, American Express y muchas de las principales instituciones financieras han aprobado SSL para el comercio sobre Internet.

Hay que destacar que SSL/TLS se compone de cuatro protocolos. Estos protocolos funcionan de manera idéntica en SSL y en TLS pero incorporan algunos detalles en TLS para su mejor funcionamiento. A continuación se definen estos cuatro protocolos sin entrar en mucho detalle:

- **Record Protocol:** encapsula los protocolos de nivel más alto y construye un canal de comunicaciones seguro. Se podría decir que es un protocolo de transporte.
- **Handshake Protocol:** se encarga de gestionar la negociación de los algoritmos de cifrado y la autenticación entre cliente y servidor. Define las claves de sesión utilizadas para cifrar. Se podría decir que es un protocolo de autenticación.
- **Change Cipher Spec Protocol:** es un mensaje de un byte para notificar cambios en la estrategia de cifrado.
- **Alert Protocol:** señala alertas y errores en la sesión establecida.

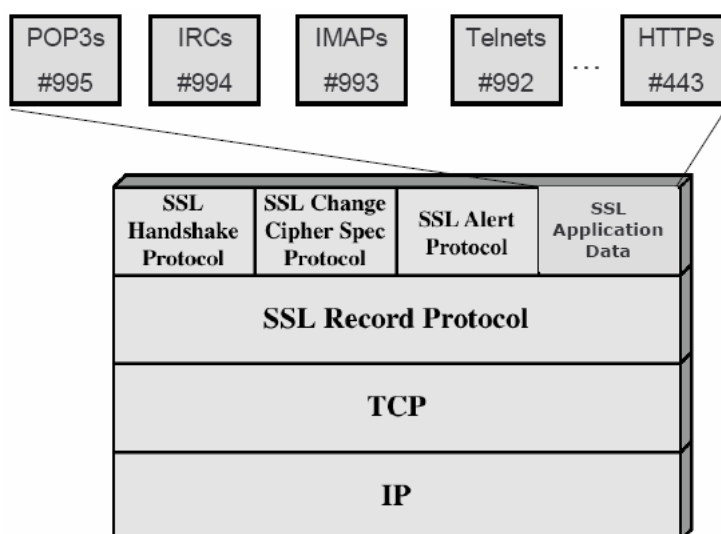


Figura 4. Situación de SSL/TLS en la pila de protocolos OSI

Existen multitud de implementaciones del protocolo, tanto comerciales como de libre distribución siendo una de las más populares la biblioteca OpenSSL de la cual se hablará con más detalle en apartados posteriores.

SSL es capaz de trabajar con la mayoría de protocolos que trabajan sobre TCP de tal manera que el IANA les tiene asignado un número de puerto por defecto, por ejemplo el protocolo HTTP sobre SSL ha sido denominado HTTPS y tiene como puerto el 443.

Sin entrar en mucho detalle (más adelante en este mismo apartado se profundizará mas sobre el funcionamiento de SSL/TLS) SSL se basa en un esquema de clave pública para el intercambio de claves de sesión. En primer lugar cliente y servidor intercambian una clave de longitud suficiente mediante un algoritmo de cifrado asimétrico como RSA o Diffie-Hellman utilizando certificados. Mediante esa clave se establece un canal seguro, utilizando para ello un algoritmo simétrico previamente negociado. Los mensajes a ser transmitidos, se fragmentan en bloques, se comprimen y se les aplica un algoritmo Hash para obtener un resumen (MAC del mensaje) para asegurar la integridad.

Para poder entender bien el funcionamiento de SSL/TLS se tendrán que definir los conceptos de sesión y conexión para estos protocolos. En SSL/TLS una **sesión** es una asociación entre un cliente y un servidor. Las sesiones se crean mediante el protocolo Handshake y coordina los estados del cliente y del servidor. El estado de una sesión incluye la siguiente información:

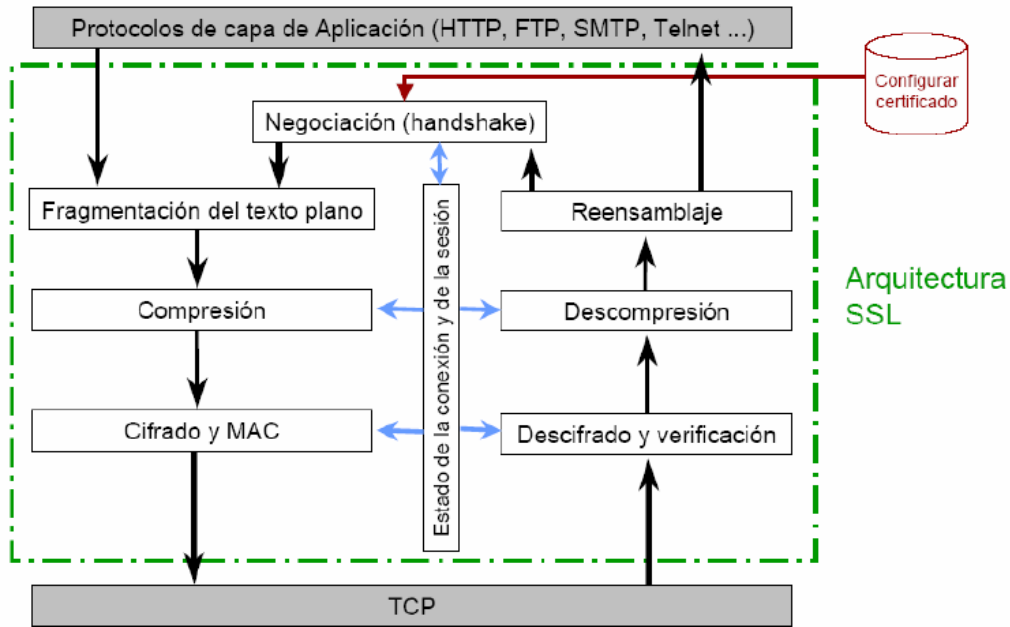


Figura 5. Arquitectura SSL/TLS

- **Identificador de Sesión:** consiste en una secuencia arbitraria de bytes elegida por el servidor para identificar una sesión activa.
- **Certificado de la Entidad Par:** el certificado del otro extremo de la comunicación (puede ser nulo).
- **Método de Compresión:** indica el algoritmo usado para comprimir los datos antes de cifrarlos.
- **Especificación de Cifrado:** especifica el algoritmo de cifrado de datos (DES, AES,...) y el algoritmo MAC (MD5 o SHA-1). También define atributos como el tamaño del Hash.
- **Clave Maestra:** una clave secreta de 48 bytes intercambiado entre cliente y servidor.

En SSL/TLS una **conexión** es transitoria y está asociada solamente a una sesión, mientras que una sesión puede tener múltiples conexiones. En otras palabras, las sesiones se usan para evitar la costosa negociación de los parámetros de cada conexión. El estado de una conexión incluye la siguiente información:

- **Valores Aleatorios del Servidor y del Cliente:** es una secuencia de bytes elegido por el servidor y el cliente para cada conexión.

- **Clave Secreta MAC de Escritura del Servidor:** es el secreto utilizado en operaciones MAC sobre los datos del servidor.
- **Clave Secreta MAC de Escritura del Cliente:** es el secreto utilizado en operaciones MAC sobre los datos del cliente.
- **Clave de Escritura del Servidor:** es la clave secreta para el cifrado de datos por el servidor y descifrado de datos por el cliente.
- **Clave de Escritura del Cliente:** es la clave secreta para el cifrado de datos por el cliente y descifrado de datos por el servidor.
- **Vector de Inicialización (IV) del Cliente y Servidor:** vectores de inicialización utilizados para bloques de cifrado en estado CBC.
- **Número de Secuencia:** cada estado de conexión contiene un número de secuencia que se mantiene independientemente para los estados de lectura y escritura. El número de secuencia debe ser reseteado a cero cada vez que un estado de conexión pasa a estado activo

Tras haber definido los conceptos de conexión y sesión en SSL/TLS se va a explicar el funcionamiento de SSL/TLS explicando, con más detalle, los cuatro protocolos de los que se compone SSL/TLS: Record Protocol, Handshake Protocol, Change Cipher Spec y Alert Protocol.

EL SSL/TLS **Record Protocol** es el protocolo de transporte que proporciona a cada conexión:

- **Confidencialidad:** utilizando una clave compartida generada durante el protocolo Handshake para el cifrado convencional de los datos.
- **Integridad del Mensaje:** el protocolo de Handshake también genera una clave secreta común que se usa para formar el MAC o código de autenticación del mensaje.

En el funcionamiento del SSL Record Protocol intervienen mecanismos criptográficos, para los cuales son necesarios ciertos parámetros como la clave secreta para el cifrado. Estos parámetros se negocian durante el establecimiento del protocolo Handshake, que además permite la autenticación de cliente y servidor.

El proceso que sigue el SSL/TLS Record Protocol es el siguiente:

1. Los mensajes se fragmentan en bloques de 2^{14} bytes o menos.
2. Se aplica compresión opcionalmente.
3. Se calcula un MAC o una función Hash sobre los datos comprimidos.
4. El MAC junto con el mensaje se cifra con un algoritmo simétrico.
5. Finalmente se le añade una cabecera de registro o SSL Record Header

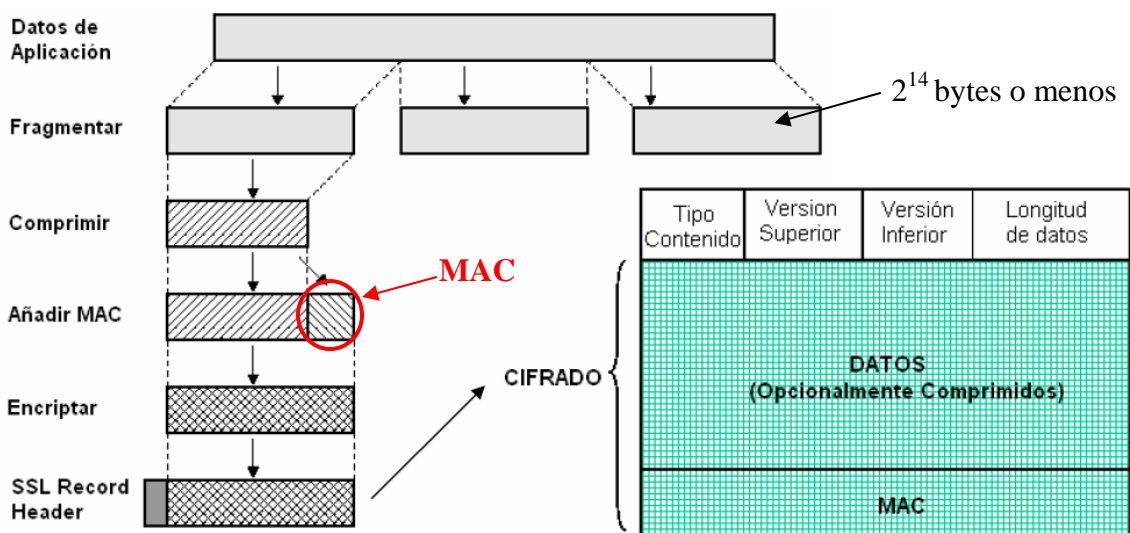


Figura 6. Cifrado y Formato de Datos de Aplicación con Record Protocol

Para calcular la MAC del paquete de datos comprimido SSL y TLS utilizando MAC de clave compartida, es decir, HMAC. El cálculo de dicha MAC es diferente para la versión 3 de SSL y para la versión 1 de TLS, siendo la implementación de TLS versión 1 más segura:

$$\text{Hash}(\text{clave_MAC} \parallel \text{opad} \parallel \text{hash}(\text{clave_MAC} \parallel \text{ipad} \parallel \text{seq_num} \parallel \text{SSLCompressed.type} \parallel \text{SSLCompressed.length} \parallel \text{SSLCompressed.fragment}))$$

Figura 7. Calculo del MAC en Record Protocol de SSL v3

$$\text{HMAC}_K = H[(K^+ \oplus \text{opad}) \parallel H[(K^+ \oplus \text{ipad}) \parallel X]]$$

$$\text{HMAC}(\text{clave_MAC}, \text{seq_num} \parallel \text{TLSCompressed.type} \parallel \text{TLSCompressed.version} \parallel \text{TLSCompressed.length} \parallel \text{SSLCompressed.fragment})$$

Figura 8. Calculo del MAC Record Protocol de TLS v1

Siendo:

- **H:** la función Hash empleada MD5 o SHA-1.
- **X:** el texto plano.
- **K⁺:** la clave secreta con relleno de ceros a la izquierda hasta que iguale la longitud del bloque de entrada de las funciones Hash empleadas.
- **Ipadd:** 00110110 repetido.
- **Opadd:** 01011100 repetido.

Mencionar también que el relleno mínimo que se aplican a los datos de aplicación es diferente en SSL versión 3 y en TLS versión 1. Mientras que en SSL versión 3 el relleno mínimo de los datos que se van a cifrar tiene que ser múltiplo de la longitud del

bloque de cifrado (tamaño según la técnica de cifrado), con TLS el relleno puede ser cualquiera siempre que sea menor de 255 bytes.

Los algoritmos soportados por SSL son los que se muestran a continuación, en los que habría que exceptuar el uso de Fortezza (por ser un algoritmo propietario) en el protocolo TLS versión 1:

Cifrado Bloque		Cifrado Flujo	
Algoritmo	Tamaño K	Algoritmo	Tamaño K
IDEA	128	RC4-40	40
DES	56	RC4-128	128
3DES	112		
RSA	1024		
DSA	1024		
FORTEZZA	80		

Figura 9. Algoritmos de Cifrado Soportados por SSL/TLS

El protocolo **Change Cipher Spec** es muy simple e idéntico tanto para SSL versión 3 como para TLS versión 1. Consiste en un único mensaje de un byte de contenido 1 que tiene como objetivo pasar del modo pendiente, para negociar los parámetros de una conexión, al modo operativo, en el que los parámetros ya se han establecido y la comunicación es totalmente segura.

El tercer protocolo de SSL/TLS que se va a presentar es el **Alert Protocol**. Como su nombre indica el objetivo de este protocolo es el de transmitir alertas mediante mensajes de 2 bytes, siendo su formato el que se muestra a continuación:

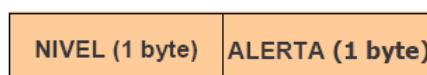


Figura 10. Formato del Protocolo Alert de SSL/TLS

Donde:

- **Nivel:** toma los valores 1 para indicar “Aviso” o 2 para indicar “Fatal”
- **Alerta:** para el nivel de alerta “Aviso” existen las alertas notificación de cierre, no certificado, certificado erróneo, certificado no permitido, certificado revocado, certificado caducado y certificado desconocido. Para el nivel “Fatal” existen las alertas mensaje inesperado, MAC de registro erróneo, fallo de descompresión, fallo de negociación y parámetro ilegal.

Existen muy pocas diferencias en el protocolo de alertas entre SSL versión 3 y TLS versión 1. TLS versión 1 no incluye la alerta “no certificado” y añade nuevas alertas como “fallo de descifrado”, “autoridad de certificación desconocida” y “seguridad insuficiente”.

El cuarto protocolo y más importante que forma SSL/TLS es el **Handshake Protocol**. Mediante este protocolo se generan los parámetros criptográficos que van a definir el estado de una sesión (una sesión SSL/TLS siempre empieza con el Handshake). Este protocolo permite la autenticación entre el cliente y el servidor y la negociación de los algoritmos de cifrado y las claves y subclaves. Por ejemplo, uno de los parámetros a los que deben llegar a acuerdo el cliente y el servidor es la versión de SSL/TLS y método de compresión.

TIPO (1 byte)	LONGITUD (3 bytes)	CONTENIDO (≥ 1 byte)
------------------	-----------------------	-------------------------

Figura 11. Mensaje Handshake en SSL/TLS

El protocolo Handshake opera sobre el Record Protocol de SSL/TLS y se establece antes de enviar los datos de aplicación. También cabe destacar que tiene dos modos de negociación de sesión: el “Full Handshake”, para la primera conexión, y el “Abbreviated Handshake”, para conexiones posteriores.

Este protocolo consta de cuatro fases en los que se negocian los parámetros de una sesión:

- **Fase 1:** aquí se establecen las capacidades de seguridad (versión de protocolo, identificador de sesión, suite de cifrado, método de compresión y números aleatorios iniciales).
- **Fase 2:** en esta fase el servidor puede enviar un certificado, intercambio de clave y solicitud de certificado.
- **Fase 3:** el cliente envía su certificado, en caso de habérselo solicitado, el intercambio de clave y puede que envíe verificación de certificado.
- **Fase 4:** se produce el intercambio de suite de cifrado y finalización del protocolo Handshake. En esta fase se completa el establecimiento de la conexión segura.

En la **fase 1 del protocolo Handshake** el cliente envía al servidor la información necesaria para poder establecer una comunicación segura con SSL/TLS. El servidor decidirá si soporta esos parámetros que el cliente le ha enviado y se lo comunica al cliente. Por ejemplo, el cliente le comunicará la suite de cifrado y el servidor le contestará con la suite de cifrado que va a utilizar junto con otros parámetros.

El mensaje “**Client_hello**” es el primer mensaje que se envía en la fase 1 del protocolo Handshake. Este mensaje contiene los siguientes parámetros a negociar entre cliente y servidor:

- **Versión:** número de la versión más alta de SSL/TLS que el cliente soporta.
- **Valor Aleatorio:** número aleatorio inicial del cliente.
- **Identificador de Sesión:** si el valor del identificador de sesión es diferente de cero, se tendrá que crear una nueva conexión dentro de esa sesión actualizando

los parámetros de la conexión existente, si el valor es cero indica una nueva conexión en una nueva sesión por lo que se actualizarán los valores tanto de la sesión como de la conexión.

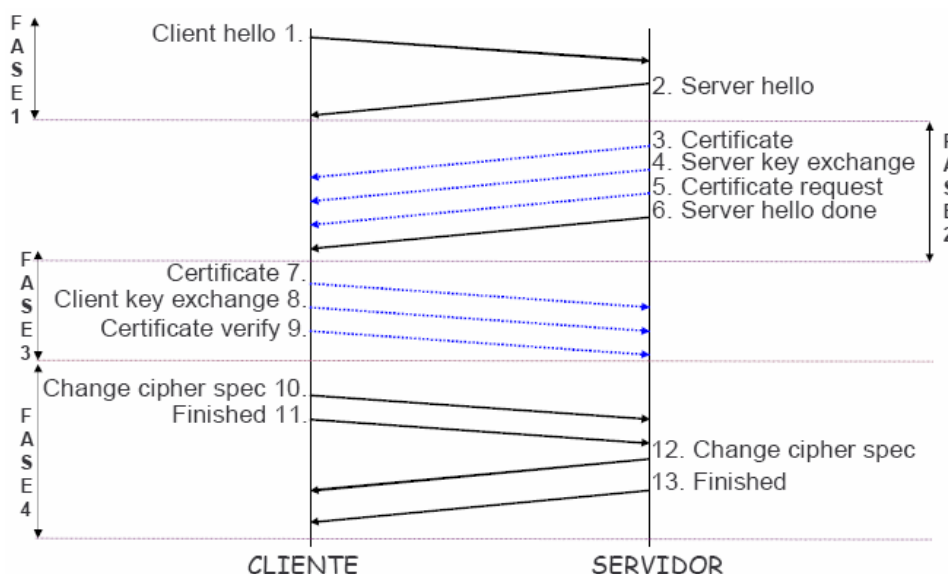


Figura 12. Fases y Mensajes del Protocolo Handshake de SSL/TLS

- **Suite de Cifrado:** contiene una lista de suites de cifrado soportados por el cliente. En esta suite de cifrado deberán aparecer el algoritmo de intercambio de claves, el algoritmo de cifrado, el tipo de cifrado, el tamaño del Hash, parámetros para calcular claves, tamaño de vector de inicialización (IV),...
- **Método de Compresión:** el método o métodos que soporta el cliente para comprimir los datos de aplicación.

El mensaje “**Server_Hello**” es enviado por el servidor tras haber recibido el mensaje “**Client_Hello**” por parte del cliente. En este mensaje el servidor selecciona los parámetros que soporta el cliente:

- **Versión:** el servidor es el que elige la versión de SSL/TLS más alta propuesta por el cliente y que, además, soporta el servidor.
- **Valor aleatorio:** número aleatorio inicial del servidor.
- **Identificador de Sesión:** si el identificador de sesión del cliente, recibido por el servidor, es igual a cero, el identificador de sesión del servidor contendrá un valor distinto de cero, indicando que se ha creado una nueva sesión. Por otro lado, si el identificador de sesión del cliente es diferente de cero, el servidor comprobará en su caché si guarda información sobre esa conexión, y si es así y se puede crear una nueva conexión responde con el mismo identificador de sesión que el del cliente.
- **Suite de Cifrado:** escoge de entre una lista de suites de cifrado soportados por el cliente. En esta suite de cifrado deberán aparecer el algoritmo de intercambio de claves, el algoritmo de cifrado, el tipo de cifrado, el tamaño del Hash, parámetros para calcular claves, tamaño de vector de inicialización (IV),...

- **Método de Compresión:** el servidor escoge el método que soporta el cliente para comprimir los datos de aplicación.

En la **fase 2 del protocolo Handshake** se produce, en pocas palabras, la autenticación del servidor e intercambio de claves entre cliente y servidor. Los mensajes enviados en esta fase son “Certificate”, “Server_Key_Exchange”, “Certificate_Request” y “Server_Hello_Done”, siendo este último el único mensaje que el servidor está obligado a enviar al cliente.

El primer mensaje de la fase 2 es el mensaje “**Certificate**”, que contiene el certificado X.509 (firmado por la CA) del servidor y que permite autenticarse al cliente. Este certificado contiene, además, la clave pública del servidor que será utilizada para intercambiar las claves de sesión.

El segundo mensaje de la fase 2 es el mensaje “**Server_Key_Exchange**”. Este mensaje contiene la clave pública del servidor para el intercambio de claves. Este mensaje no es necesario si el servidor ha enviado su certificado con los parámetros públicos Diffie-Hellman para el intercambio de claves o se utiliza RSA para el intercambio de claves.

El mensaje “**Certificate_Request**” es el tercer mensaje de la fase 2 y tiene la función de pedir al cliente que se autentique, para ello le pide al cliente un determinado tipo de certificado y una lista de autoridades de certificación (CA) aceptables.

En el mensaje “**Server_Hello_Done**” no se envía ningún parámetro, ya que pone fin a los mensajes de la fase 2 asociados al servidor.

Con la información enviada en esta fase el cliente autentica al servidor. En el caso en que se esté utilizando RSA para intercambio de claves, el servidor envía una clave pública para que el cliente cifre con ella la información necesaria para generar una clave secreta común y la devuelva al servidor. De este modo, el servidor la descifrará con la clave privada y podrá generar la misma clave común. Por otra parte, en caso de que el servidor no transmita su certificado al cliente, le debe enviar una clave pública mediante el mensaje “Server_Key_Exchange” utilizando para ello el algoritmo Diffie-Hellman.

La **fase 3 del protocolo Handshake** solo se lleva a cabo en el caso en que haya que autenticar al cliente (pues el servidor envió el mensaje “Certificate_Request”). En tal caso, el cliente debe responder con su certificado X.509, si lo tiene, o con un mensaje de alerta indicando que no lo tiene (alerta “No_Certificate”). Los posibles mensajes que puede enviar el cliente en esta fase del protocolo Handshake son: “Certificate”, “Client_Key_Exchange” y “Certificate_Verify”.

El primer mensaje enviado por el cliente en la fase 3 es el mensaje “**Certificate**”, en el que incluye el certificado X.509 del cliente para autenticarse al servidor.

El segundo mensaje de la fase 3 es el mensaje “**Cliente_Key_Exchange**”, en el que el cliente envía al servidor una clave secreta o número aleatorio generado, también denominada clave pre-master, de 48 bytes, cifrada con la clave pública del servidor. El servidor obtendrá dicha clave pre-master descifrando con su clave privada. El cliente y el servidor utilizarán la clave pre-master para calcular la clave maestra, las claves de sesión y las claves MAC.

El tercer mensaje de la fase 3 es el mensaje “**Certificate_Verify**”, en el que se verifica que el cliente posee la clave privada en concordancia con el certificado del cliente. Este mensaje se envía junto al anterior y consta de una firma Hash que abarca los mensajes anteriores. El cifrado se realiza con la clave privada del cliente.

La **fase 4 del protocolo Handshake** es la última fase de este protocolo, y sin entrar mucho en detalle, el cliente envía un mensaje al servidor diciendo que los siguientes mensajes serán cifrados con la clave de sesión o clave maestra. Además envía un mensaje cifrado comunicando al servidor que la parte del cliente del protocolo Handshake ha finalizado.

Por otro lado, el servidor envía un mensaje al cliente diciendo que los mensajes serán cifrados con la clave de sesión o clave maestra. Por último, al igual que ha hecho el cliente, envía un mensaje cifrado diciendo que su parte del protocolo Handshake ha finalizado.

Esta fase consta de dos mensajes que son iguales tanto para el cliente como para el servidor. El primer mensaje es el “**Change_Cipher_Spec**” que sirve para dar como concluido el intercambio, pasando del estado pendiente al estado operativo. El segundo y último mensaje es el “**Finished**” que sirve para finalizar el protocolo Handshake y para comenzar a transmitir los datos de aplicación protegidos con las claves y algoritmos negociados.

Como se ha comentado en la explicación del protocolo Handshake, se realiza el cálculo de una clave maestra a partir de otros parámetros establecidos en este mismo protocolo. El tamaño de esta clave maestra es de 48 bytes y el uso de esta clave es válido para una única sesión. El cálculo de esta clave maestra se realiza en dos pasos:

1. Cliente genera la clave pre-master (K_{previo}) y se la envía cifrada al servidor mediante el algoritmo de intercambio de claves RSA o por parámetros Diffie-Hellman.
2. Se realiza el cálculo de la clave maestra por parte del cliente y del servidor. En la versión 3 del protocolo SSL las operaciones para calcular la clave maestra se muestran en las dos siguientes figuras:

$$K_{maestra} = MD5 (K_{previo} || SHA ('A' || K_{previo} || clienthello.random || serverhello.random)) || MD5 (K_{previo} || SHA ('BB' || K_{previo} || clienthello.random || serverhello.random)) || MD5 (K_{previo} || SHA ('CCC' || K_{previo} || clienthello.random || serverhello.random))$$

Figura 13. Cálculo de la Clave Maestra (I)

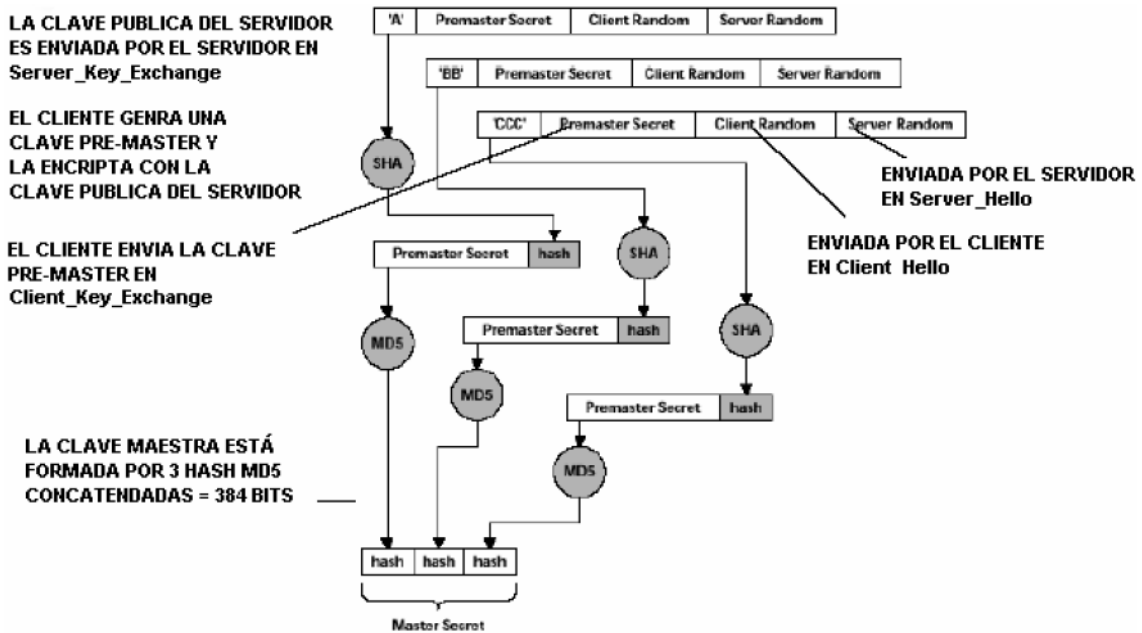


Figura 14. Cálculo de la Clave Maestra (II)

Además, la misma clave maestra se reutiliza para calcular:

- Clave Secreta MAC de Escritura del Servidor.
- Clave Secreta MAC de Escritura del Cliente.
- Clave de Escritura del Servidor.
- Clave de Escritura del Cliente.
- Vector de Inicialización (IV) del Cliente y Servidor.

$$K_{block} = MD5(K_{maestra} || SHA('A' || K_{maestra} || clienthello.random || serverhello.random)) || MD5(K_{maestra} || SHA('BB' || K_{maestra} || clienthello.random || serverhello.random)) || MD5(K_{maestra} || SHA('CCC' || K_{maestra} || clienthello.random || serverhello.random)) \dots\dots$$

Figura 15. Calculo de otras Claves a partir de la Clave Maestra (I)

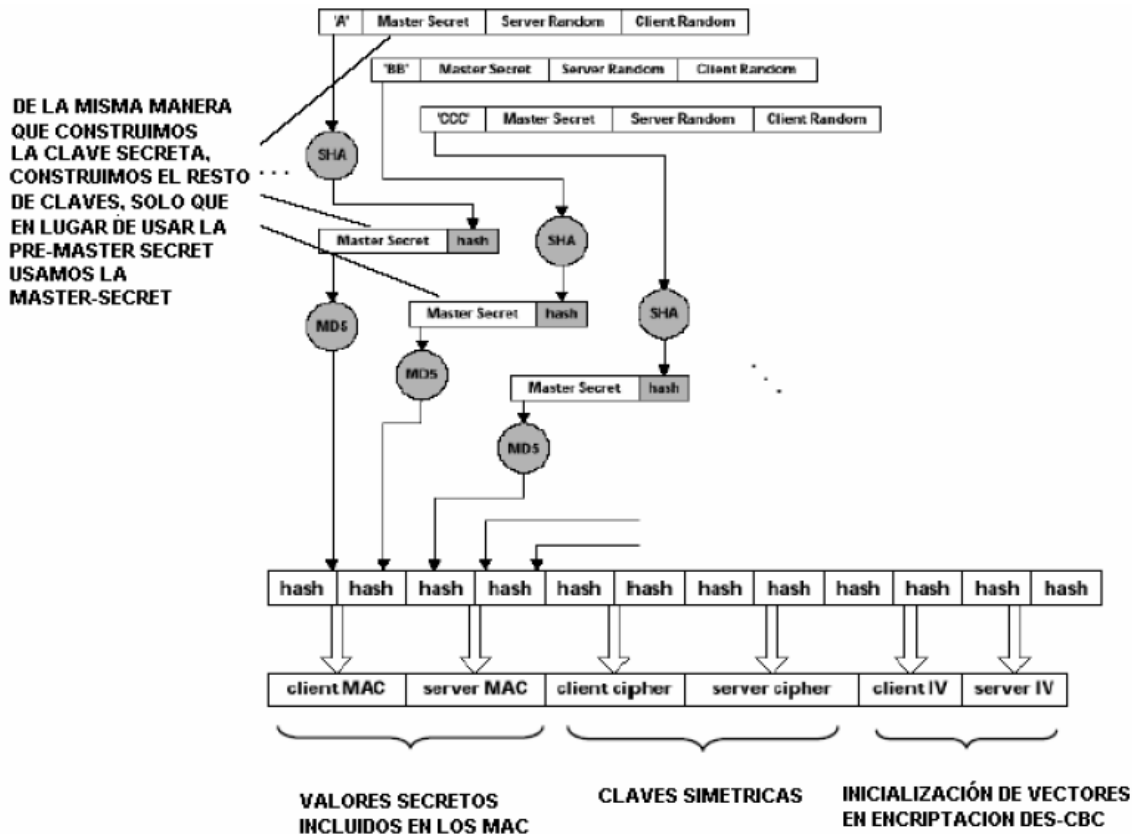


Figura 16. Calculo de otras Claves a partir de la Clave Maestra (II)

Para la versión 1 de TLS el cálculo de la clave maestra de sesión es bastante diferente que en la versión 3 de SSL. Para el cálculo de dicha clave se emplea una función pseudo aleatoria o función PRF (Pseudos Random Function) que tiene como función la expansión de dicha clave maestra. Como sucede en la versión 3 de SSL la clave maestra es de 48 bytes y se necesita generar previamente una clave pre-master (K_{previa}). Para el cálculo de dicha clave maestra en TLS versión 1 se realizan las siguientes operaciones:

$$\text{PRF}(\text{secret}, \text{label}, \text{seed}) = \text{P_MD5}(\text{S1}, \text{label} \parallel \text{seed}) \oplus \text{P_SHA-1}(\text{S2}, \text{label} \parallel \text{seed})$$

$$\text{P_hash} = \text{HMAC_hash}(\text{secret}, \text{A}(1) \parallel \text{seed}) \parallel \text{HMAC_hash}(\text{secret}, \text{A}(2) \parallel \text{seed}) \parallel \text{HMAC_hash}(\text{secret}, \text{A}(1) \parallel \text{seed}) \parallel \dots$$

A(n):

$$\text{A}(0) = \text{seed}$$

$$\text{A}(1) = \text{HMAC_hash}(\text{secret}, \text{A}(0))$$

...

$$\text{A}(i) = \text{HMAC_hash}(\text{secret}, \text{A}(i-1))$$

Figura 17. Cálculo de la Clave Maestra en TLS versión 1 (I)

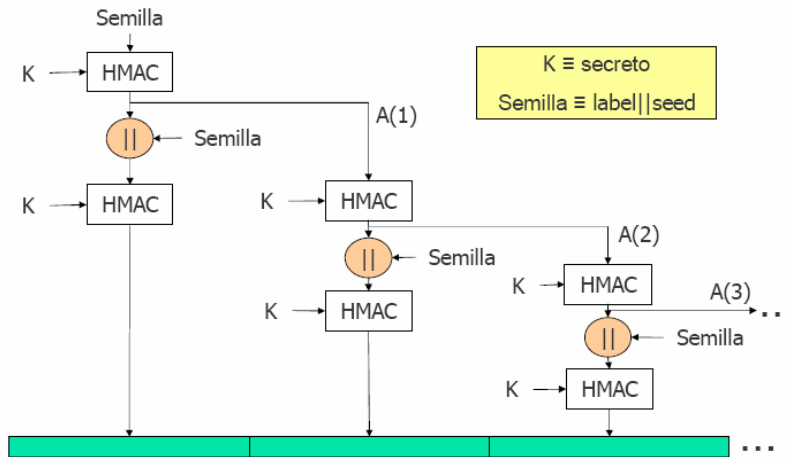


Figura 18. Cálculo de la Clave Maestra en TLS versión 1 (II)

$$K_{\text{maestra}} = \text{PRF}(K_{\text{previa}}, \text{"master secret"}, \text{clienteHello.random} \parallel \text{servidorHello.random})$$

Figura 19. Cálculo de la Clave Maestra en TLS versión 1 (III)

1.2.7.- La alternativa IPsec

IPsec (de las siglas en inglés Internet Protocol Security) es un conjunto de estándares abiertos que trabajan de forma conjunta para garantizar entre entidades pares en el nivel de red confidencialidad, integridad y autenticación independientemente de cual sea el medio de transporte (Frame Relay, PPP, xDSL, ATM,...). El objetivo para el que fue diseñado IPsec fue para las comunicaciones sobre el protocolo de Internet (IP). IPsec y los protocolos que implementa se han especificado en numerosos RFCs entre los que se encuentran 1826, 1827, 2401, 2402, 2406, 2408, 4301 y 4309.

El protocolo IPsec permite definir un túnel entre dos pasarelas. Una pasarela IPsec consistiría normalmente en un router de acceso o un cortafuegos en el que esté implementado el protocolo IPsec. Las pasarelas IPsec están situadas entre la red privada del usuario y la red compartida del operador.

Los túneles IPsec se establecen dinámicamente y se liberan cuando no están en uso. Para establecer un túnel IPsec, dos pasarelas deben autenticarse y definir los algoritmos de seguridad y las claves que utilizarán para el túnel. El paquete IP original es cifrado en su totalidad e incorporado en encabezamientos de autenticación y encriptación IPsec. Se obtiene así la carga útil de un nuevo paquete IP cuyas direcciones IP de origen y destino son las direcciones IP de red pública de las pasarelas IPsec. Se establece así la separación lógica entre los flujos de tráfico de la VPN en una red IP compartida. Seguidamente, se utiliza un encaminamiento IP tradicional entre los extremos del túnel.

IPsec tiene dos modos de funcionamiento:

- **Modo Transporte:** en el modo transporte de IPsec se establece una seguridad extremo a extremo entre cliente a servidor, servidor a servidor o cliente a cliente. En este modo solo se protege la carga útil (capa de transporte UDP o TCP). En cada extremo de la comunicación se requiere la implementación de IPsec.

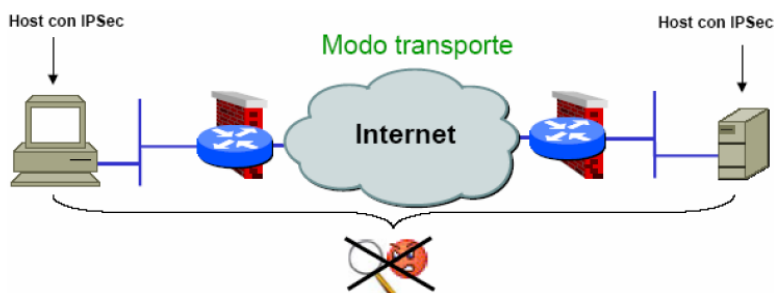


Figura 20. Modo Transporte en IPsec

En el modo transporte, IPsec sólo maneja la carga del paquete IP, insertándose la cabecera de IPsec entre la cabecera IP y la cabecera del protocolo de capas superiores.

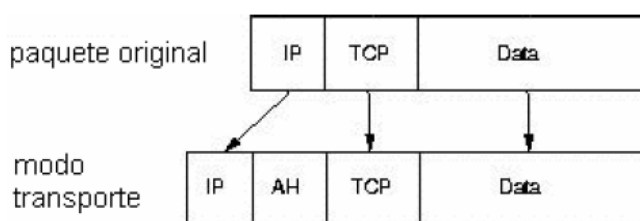


Figura 21. Paquete IP en IPsec para Modo Transporte

- **Modo Túnel:** en el modo túnel se establece una seguridad entre los routers de dos redes siempre que éstos implementen IPsec. El tráfico existente entre host y router no es seguro excepto si se acompaña con otra tecnología de seguridad como puede ser L2TP. Una ventaja es que los host no deben implementar IPsec por lo que será más eficiente a la hora de incrementar el número de host de la red de la empresa. Este modo de IPsec es el más utilizado para implementar VPN entre sedes separadas por Internet.

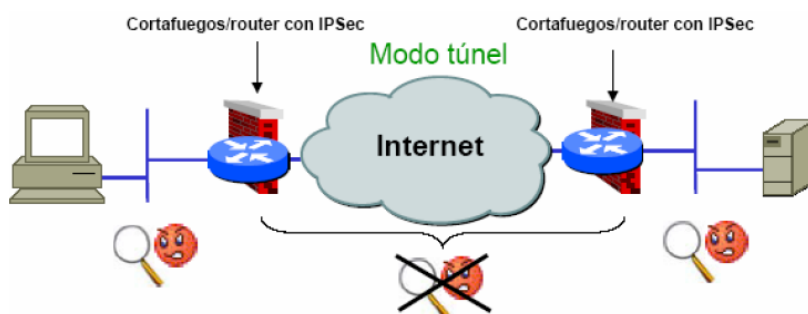


Figura 22. Modo Transporte en IPsec

En el modo túnel, todo el paquete IP (datos más cabeceras del mensaje) es cifrado y/o autenticado. El paquete IP se encapsula completamente dentro de un nuevo paquete IP que emplea el protocolo IPSec y al que se le añade una copia de la cabecera del paquete IP original.

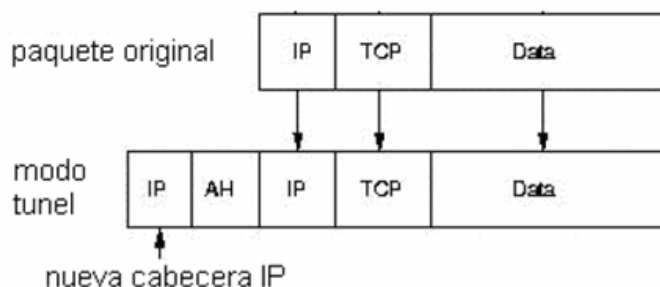


Figura 23. Paquete IP en IPSec para Modo Túnel

Antes de establecer una comunicación segura IPSec establece una asociación de seguridad (SA) entre los dos extremos de la comunicación. Una asociación de seguridad es un acuerdo entre dos o más entidades sobre los mecanismos de seguridad (desde la elección del protocolo de autenticación hasta el intercambio de claves) que emplearán para mantener una comunicación segura. Estas entidades intercambian mensajes para determinar los parámetros que configurarán su enlace seguro; estos parámetros son necesarios para la puesta en marcha de los mecanismos de seguridad que construyen la asociación de seguridad. Las asociaciones de seguridad, a su vez, se almacenan en bases de datos de asociaciones de seguridad (SAD - Security Association Databases).

Esta definición es necesaria para comprender en qué consisten los protocolos IKE (protocolo de Intercambio de Claves, del inglés Internet Key Exchange) e ISAKMP (Protocolo de Asociación de Seguridad y Gestión de Claves, del inglés Internet Security Association and Key Management Protocol). Ambos han sido diseñados para la puesta en marcha de Asociaciones de Seguridad, y podríamos decir que son complementarios. En efecto, ISAKMP provee un marco para negociar la autenticación y el intercambio de claves, mientras que IKE es un método concreto de intercambio de claves.

Una vez establecida la asociación de seguridad entre ambos participantes IPSec consigue la integridad, confidencialidad y autenticación mediante otros dos protocolos definidos por el IETF:

- **Authentication Header (AH):** proporciona integridad, autenticación y no repudio si se eligen los algoritmos criptográficos apropiados. El formato de un paquete con la cabecera AH según el modo de funcionamiento de IPSec es el siguiente:

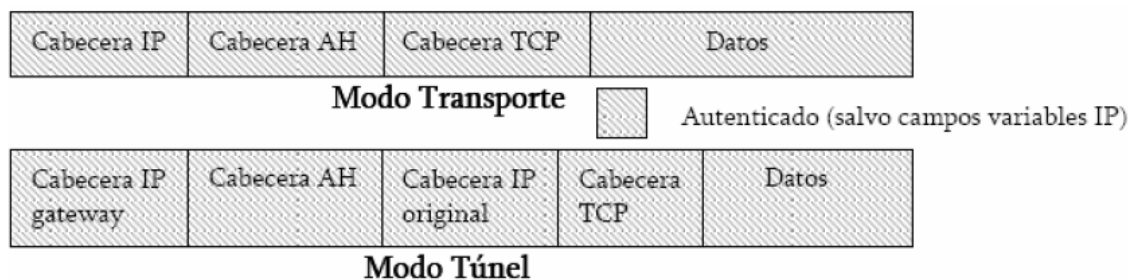


Figura 24. Formato de Paquetes con Cabecera AH, según el Modo IPsec

- **Encapsulating Security Payload (ESP):** proporciona confidencialidad y la opción, altamente recomendable, de autenticación y protección de integridad. El formato de un paquete con cifrado y autenticado con ESP según el modo de funcionamiento de IPsec es el siguiente:

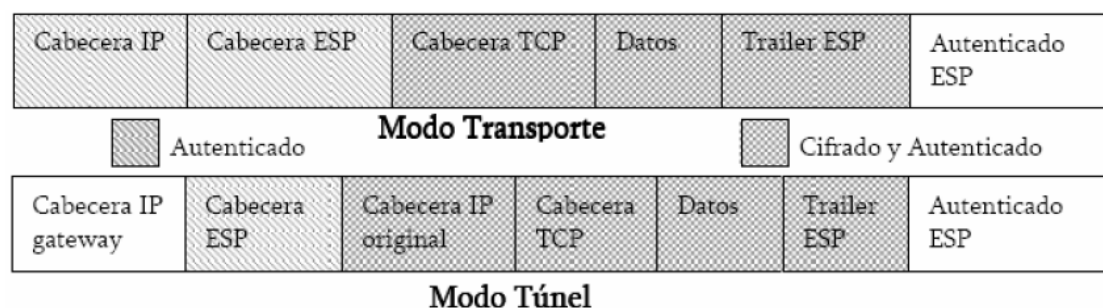


Figura 25. Formato de Paquetes ESP, según el Modo de IPsec

Para proteger la integridad de los paquetes IP, los protocolos IPsec emplean códigos de autenticación de mensaje basados en resúmenes (HMAC - Hash Message Authentication Codes). Para el cálculo de estos HMAC los protocolos HMAC emplean algoritmos de resumen como MD5 y SHA para calcular un resumen basado en una clave secreta y en los contenidos del paquete IP. El HMAC se incluye en la cabecera del protocolo IPsec y el receptor del paquete puede comprobar el HMAC si tiene acceso a la clave secreta.

Para proteger la confidencialidad de los paquetes IP, los protocolos IPsec emplean algoritmos estándar de cifrado simétrico. El estándar IPsec exige la implementación de NULL y DES. En la actualidad se suelen emplear algoritmos más fuertes: DES, 3DES, AES y Blowfish.

Para protegerse contra ataques por denegación de servicio (DoS), los protocolos IPsec emplean ventanas deslizantes. Cada paquete recibe un número de secuencia y sólo se acepta su recepción si el número de paquete se encuentra dentro de la ventana o es posterior. Los paquetes anteriores son descartados inmediatamente. Esta es una medida de protección eficaz contra ataques por repetición de mensajes en los que el atacante almacena los paquetes originales y los reproduce posteriormente.

1.2.8.- VPN SSL/TLS y VPN IPSec

En este punto de la memoria se hará una comparación entre las dos tecnologías más utilizadas en Internet para la implementación de VPN: SSL/TLS e IPSec. Para ello se describirán las ventajas y desventajas que se pueden obtener utilizando una implementación u otra en nuestra VPN, destacando sobre todo las ventajas que ofrecen respecto a seguridad y como evitan ciertos ataques. Además, se describirán posibles aplicaciones específicas que ofrece una y otra tecnología y como se pueden compensar las debilidades de IPSec con las ventajas de SSL/TLS y viceversa.

Sin embargo, actualmente, la mayoría de los expertos y también de suministradores, tienden a considerar que IPSec y SSL, más que de tecnologías en competencia, se trata de propuestas complementarias. De hecho, según argumentan los que adoptan esta actitud conciliadora, la creciente popularidad de las VPN SSL tendría como motivo su capacidad para cubrir de una forma sencilla y económica una necesidad que IPSec nunca ha podido satisfacer adecuadamente: el acceso remoto, específicamente en el área de las aplicaciones extranet. Según reconocen los expertos, a la hora de brindar este acceso universal, el protocolo IPSec, tradicionalmente utilizado en las conexiones LAN to LAN vía VPN, presentaba importantes inconvenientes.

La principal ventaja de las SSL VPN es el hecho de que, a diferencia de las implementaciones IPSec VPN, no exigen el despliegue de agentes software permanente sobre los dispositivos a los que se facilita acceso a las aplicaciones corporativas. De esta forma, se reduce significativamente la carga de soporte asociada a las implementaciones IPSec, simplificando la gestión y el mantenimiento, y reduciendo al mismo tiempo el coste de la solución. También se hace posible extender el acceso remoto a través de Internet a un número mucho mayor de usuarios, dado que soporta la conexión a los recursos corporativos desde cualquier sistema dotado de navegador Web, incluido en la práctica totalidad de los dispositivos de usuario. Los agentes VPN SSL son descargados a los PC remotos, sean estos cuales sean y estén ubicados donde estén, una vez que el usuario se haya autenticado en una aplicación SSL, instalada de forma similar a un proxy entre Internet y la red corporativa. El acceso se independiza así de un determinado dispositivo o ubicación, pasando a estar vinculado al usuario.

Con SSL, el acceso puede distribuirse, de una forma tremendamente sencilla, a más personas, lugares, y dispositivos que con IPSec, reduciendo al mismo tiempo los costes de despliegue y soporte. Una solución mágica para muchas empresas. Además, según han subrayado repetidamente los promotores de SSL VPN, al trabajar a nivel de aplicación, esta alternativa permite un control más preciso de los recursos a los que un determinado usuario está autorizado a acceder, proporcionándole a través del proxy SSL, los privilegios corporativos según su perfil.

Frente a toda esta sencillez, el acceso remoto IPSec resulta, en el mejor de los casos, difícil de desplegar cuando existen grandes cantidades de usuarios o si hay múltiples empresas y gateways implicados, algo inevitable en entornos extranet. No obstante, IPSec, cuyo funcionamiento se produce en el nivel de red, presenta importantes ventajas

cuando de lo que se trata es de realizar conexiones LAN to LAN a este nivel, permitiendo que la experiencia del usuario sea la misma que si estuviera ubicado en la propia LAN a la que accede. Además, sus restricciones respecto de los dispositivos cliente soportados, un inconveniente para proporcionar un acceso remoto más abierto, supone una ventaja en determinados aplicaciones, al elevar en cierto modo la seguridad del acceso. Así, el hecho de que sobre los dispositivos sea necesario desplegar un agente específico de forma permanente, limita los accesos a aquellos sistemas gestionados corporativamente, evitando de antemano potenciales brechas en la seguridad de la red corporativa, que pueden abrirse, por ejemplo, cuando los usuarios trabajan desde estaciones de uso público.

Entre las ventajas de IPSec también se encuentra su capacidad de trabajar con todo tipo de aplicaciones y recursos de la empresa, incluidos los heredados, sin necesidad de instalar soluciones adicionales. SSL VPN, por sí mismo, sólo brinda acceso a aquellos dotados de soporte Web, aunque puede extenderse a cualquier recurso a través de productos software añadidos. Tampoco permite el acceso a estaciones de trabajo, ni soporta tráfico de voz ni streaming. Teniendo en cuenta estas limitaciones de SSL, IPSec, debido a su capacidad de distribuir conectividad completa a nivel de red, se convierte, según los expertos, en la mejor alternativa para conectar múltiples redes privadas. Resulta también especialmente indicada cuando se trata de programas que requieren una comunicación automatizada en ambos sentidos.

Como se ha dicho, existe ya un cierto consenso en conceder a cada alternativa un hueco en la empresa por derecho propio, dejando en manos de las IPSec VPN las conexiones sitio a sitio y en las de SSL VPN el acceso remoto a través de Internet. Zanjando de alguna forma la polémica, dada la influencia de sus acciones, marcas importantes como Cisco han asumido enfoques estratégicos que demuestran su coincidencia con la idea de que en la empresa existe un espacio para cada una de estas tecnologías incorporando ambas alternativas en su oferta.

Resumiendo este apartado, se van a mostrar las principales ventajas y desventajas que ofrecen IPSec y SSL/TLS, así como sus aplicaciones y uso en otras tecnologías en la actualidad:

– **Ventajas de SSL/TLS:**

- Ofrece confidencialidad (cifrado simétrico), autenticación del servidor y del cliente (este último opcional) e integridad de los mensajes brindando unos niveles de seguridad excelentes que permiten el establecimiento de extranets con confianza y tranquilidad
- SSL constituye la solución de seguridad implantada en la mayoría de los servidores Web que ofrecen servicios de comercio electrónico ya que ofrece un canal seguro para el envío de números de tarjeta de crédito.
- Bajos costes de mantenimiento y no requiere mantenimiento en los clientes además de tener una buena interoperabilidad
- Se pueden encontrar en Internet numerosas implementaciones de libre distribución para implementar redes privadas virtuales basadas en SSL/TLS.

- Muchas implementaciones de VPN basadas en SSL/TLS ofrecen mecanismos para defenderse frente a ataques del tipo “man in the middle” y ataques de denegación de servicio (DoS)
- **Desventajas de SSL/TLS:**
- Protección parcial, ya que garantiza la integridad y confidencialidad de los datos únicamente durante el tránsito de los mismos, pero no los protege una vez recibidos por el servidor. Por tanto, un hacker podría manipular tranquilamente un servidor por lo expuesto anteriormente.
 - No es una solución totalmente transparente para el usuario final.
 - En transacciones electrónicas SSL/TLS garantiza la confidencialidad extremo a extremo pero una vez finalizada la conexión, el vendedor posee todos los datos del comprador, así como su número de tarjeta de crédito. El vendedor podría almacenar esos datos y el cliente estaría expuesto a cualquier tipo de fraude por parte de toda persona que tuviera acceso a dicha información..
 - En transacciones electrónicas SSL/TLS no garantiza la integridad de la información una vez finalizada la conexión, por lo que el vendedor podría modificar esos datos, por ejemplo, cobrando más al cliente.
 - En ciertas transacciones electrónicas SSL/TLS el cliente no necesita autenticarse, por lo que una persona con acceso a números de tarjeta de crédito robados podría realizar cualquier tipo de compra por Internet. Este es precisamente el tipo de fraude más común y que causa mayores pérdidas a las compañías de crédito.
 - En transacciones electrónicas del tipo SSL/TLS, una vez finalizada la compra, no existe ningún tipo de comprobante de compra por lo que cualquier protesta posterior carecerá de medios para su confirmación. Tampoco existe ningún documento firmado por lo que tanto el cliente como el vendedor o el banco podrían negar su participación en la compra sin que existiera la posibilidad de probar lo contrario.
 - Tiene problemas con algunos protocolos de la capa de transporte y con ciertas aplicaciones, sobre todo con protocolos no orientados a conexión.
 - Interceptando los mensajes de “Client_Hello” y “Server_Hello”, es relativamente sencillo interceptar los primeros mensajes intercambiados en el mecanismo de Handshake y manipularlos para lograr que la sesión SSL se establezca en condiciones óptimas para su posterior escucha, haciendo creer al cliente que, por ejemplo, el servidor sólo soporta la versión 2.0 del protocolo, longitudes de clave de 40 bits o que el único algoritmo común es el DES.
 - No soporta tráfico de voz ni streaming.
 - No soporta multicast

– **Aplicaciones en la Actualidad de SSL/TLS:**

- Se usa en la mayoría de los casos junto a HTTP para formar HTTPS. HTTPS es usado para asegurar páginas Web para aplicaciones de comercio electrónico, utilizando certificados de clave pública para verificar la identidad de los extremos.
- La mayoría de las aplicaciones son versiones seguras de programas que emplean protocolos que no lo son. Hay versiones seguras de servidores y clientes de protocolos como el http, nntp, ldap, imap, pop3.
- SSL/TLS se puede utilizar en aplicaciones como: Applets de Java, controles ActiveX, Microsoft FrontPage o Adobe PageMill, o FileMaker Pro de Claris...
- Librerías multiplataforma como OpenSSL.

– **Ventajas de IPSec:**

- IPSec ofrece confidencialidad (cifrado), autenticación e integridad.
- Basado en estándares y muy adecuado para tráfico totalmente IP.
- IPSec está debajo de la capa de transporte, por lo que resulta transparente para las aplicaciones.
- IPSec puede ser transparente a los usuarios finales.
- Compatible con la infraestructura de claves públicas.
- Provee un alto grado de encriptación a bajo nivel.
- Estándar abierto del sector. IPSec proporciona una alternativa de estándar industrial abierto ante las tecnologías de cifrado IP patentadas. Los administradores de la red aprovechan la interoperabilidad resultante.

– **Desventajas de IPSec:**

- En la mayoría de los casos, su implementación necesita modificaciones críticas al kernel.
- IPSec no es seguro si el sistema no lo es. Los gateways de seguridad deben estar en perfectas condiciones para poder confiar en el buen funcionamiento de IPSec.
- Puede ser vulnerable a ataques del tipo “man in the middle” y de denegación de servicio (DoS).
- Es un protocolo complejo de entender, su configuración es complicada y además requiere una configuración minuciosa en el cliente. Su administración suele ser lenta y complicada.
- Tiene un alto coste de implementación y de mantenimiento.
- IPSec autentica máquinas, no usuarios: el concepto de identificación y contraseña de usuarios no es entendido por IPSec, si lo que se necesita es limitar el acceso a recursos dependiendo del usuario que quiere ingresar,

entonces habrá que utilizar otros mecanismos de autenticación en combinación con IPSec.

- Problemas con traducción de direcciones NAT (Network Address Translation).
- Diferentes implementaciones de distintos proveedores pueden ser incompatibles entre si.
- Necesita del uso de muchos puertos y protocolos en el sistema hardware que lo implemente (router, firewall,...).

– **Aplicaciones en la Actualidad de IPSec:**

- Desarrolla y depura aplicaciones web ASP.NET en Delphi 2005 mediante Cassini, Delphi, JBuilder 2006, Beta Visual Studio 2006 Desarrollo de aplicaciones Web y Windows con Visual Basic 2005.
- Comercio electrónico de negocio a negocio pero con menos influencia que SSL/TLS.
- El proyecto FreeS/WAN es la primera implementación completa y de código abierto de IPsec para Linux.
- En las redes IPv6 será obligatoria la implementación de IPSec.

Capítulo 2: OpenVPN

2.1.- ¿Qué es OpenVPN? Características Principales

OpenVPN es una solución de conectividad basada en software, una utilidad de código abierto (está publicado bajo la licencia GPL, de software libre) para soluciones SSL/TLS VPN. Fue creado por James Yonan en el año 2001 y ha estado siendo mejorado desde entonces. OpenVPN ofrece conectividad punto a punto con validación jerárquica de usuarios y host conectados remotamente, además de una amplia gama de configuraciones VPN basadas en SSL/TLS, incluyendo acceso remoto, LAN to LAN VPN, seguridad para Wi-Fi (redes inalámbricas bajo estándar IEEE 802.11), soluciones de balanceo de carga, respuesta ante fallos y diferentes técnicas de control de acceso. Partiendo de una premisa muy importante en los sistemas de seguridad: **“la complejidad es el enemigo de la seguridad”**, OpenVPN ofrece, con un bajo coste efectivo, una alternativa a otras tecnologías VPN orientadas para las PYMEs y empresas del mercado. OpenVPN tiene asignado y reservado el **puerto 1194** de manera oficial por la IANA.

La facilidad de uso de OpenVPN ha simplificado mucho la configuración de las VPN y arroja a la basura muchas de las complejidades que caracterizan a otras implementaciones de VPN, como por ejemplo, la que más se va a mencionar por su importancia e influencia en el mercado, IPSec, y ha hecho más accesible para la gente inexperta este tipo de tecnología. El modelo de seguridad de OpenVPN está basado en la arquitectura SSL/TLS, que es el estándar escogido actualmente por la industria para establecer comunicaciones seguras a través de Internet.

Un aspecto que hay que tener en cuenta durante todo el estudio con OpenVPN es que esta herramienta implementa redes seguras en la capa 2 o 3 (según el modo que utilice OpenVPN: Tunnel o Bridge) de la pila de protocolos OSI utilizando como extensión el protocolo SSL/TLS, soportando métodos de autenticación del cliente de manera flexible. Las implementaciones de SSL/TLS más conocidas hoy en día operan a través de un navegador Web (lo que se conoce como HTTPS), ya que se han implementado aproximaciones de SSL/TLS para aplicaciones en la capa de aplicación de la pila OSI (capa 7). Pero hay que tener muy en cuenta que este tipo de implementación Web con SSL/TLS no es una VPN y que OpenVPN no es una aplicación Web Proxy ni opera a través de un navegador Web, ya que opera sobre la capa 2 o 3 de la pila OSI. Por tanto, un cliente de OpenVPN no podrá utilizar un navegador Web para conectarse al servidor de OpenVPN y mantener una comunicación segura a través de la VPN.

Además, OpenVPN es una herramienta **multiplataforma**, soportadas en sistemas operativos como Linux, Windows 2000/XP y Vista, OpenBSD, FreeBSD, NetBSD, Mac OS X y Solaris. Además, soporta autenticación del cliente mediante dos factores, permite utilizar políticas de acceso a usuarios y grupos específicos y reglas de firewall aplicadas a las interfaces virtuales utilizadas por OpenVPN para el filtrado de paquetes IP.

Otra de las ventajas de utilizar OpenVPN es la compatibilidad que ofrece con la infraestructura de clave pública (PKI) mediante el uso de certificados X.509 y la técnica de intercambio de claves RSA, compatible con NAT, DHCP y con los dispositivos de red virtuales TUN/TAP. Por el contrario, OpenVPN no ofrece compatibilidad con estándares tales como IPSec, IKE, PPTP o L2TP.

Aunque hasta ahora se ha utilizado la anotación “SSL/TLS”, OpenVPN utiliza el protocolo TLS versión 1, estandarizado por el IETF, como protocolo subyacente de autenticación y de negociación de claves. La razón por la que OpenVPN utiliza este protocolo, es porque TLS es la última evolución de la familia de protocolos SSL desarrollados en un principio, en 1996, por Netscape para asegurar el primer navegador Web de dicha firma. TLS y las versiones antiguas y actuales de SSL han visto generalizado el uso de la Web durante muchos años y han sido ampliamente analizados por las deficiencias que tenían en estas implementaciones. A su vez, este análisis ha dado lugar a un consecuente fortalecimiento del protocolo de tal manera que hoy, el protocolo SSL/TLS se considera uno de los más fuertes y más maduros protocolos seguros disponibles. Por estos motivos y por otros tantos, TLS es una excelente elección para implementar los mecanismos de autenticación y mecanismos de intercambio de claves en una VPN.

OpenVPN no opera en el kernel, sino que **opera en el espacio de usuario** incrementando de esta manera la seguridad y la escalabilidad. Según el autor, James Yonan, uno de los mayores frenos de IPSec es que añade una gran complejidad en kernel. Como ya se ha comentado la complejidad es el enemigo de la seguridad. El problema de añadir dicha complejidad de seguridad software en el kernel es que se ignora un importante principio de los sistemas de seguridad: nunca se ha de diseñar un sistema en el que si uno de los componentes cae pueda poner en peligro todo el sistema. Un simple desbordamiento de un buffer en el espacio del kernel provocaría un compromiso total en la seguridad del sistema. Es por esta razón que OpenVPN ubica su complejidad y ejecuta su código dentro del espacio de usuario pudiendo contener los fallos en este espacio más rápidamente sin comprometer la seguridad del sistema.

Por el contrario, IPSec se aleja bastante de la arquitectura de sistema operativo segura basada en anillos, que es el principio de no interferencia con el espacio de usuario. Este principio divide el sistema operativo en anillos numerados con diferentes grados de privilegios. El anillo 0 está reservado para el núcleo en sí y procesos esenciales. El anillo 1 está pensado para otros procesos de sistema que necesitan un acceso a bajo nivel de hardware. Conforme incrementamos el número del anillo los privilegios van decreciendo. El anillo 3 es el lugar donde se encuentran la mayoría de los procesos. Esta arquitectura no permite que los procesos con una numeración mayor interfieran en la ejecución de procesos con una numeración menor. De esta manera se mejora la estabilidad y la seguridad en las aplicaciones y permiten la implementación de sistemas multiproceso y multiusuario.

Cuando se habla de que OpenVPN trabaja en el espacio de usuario se quiere decir que no necesita intervenir de manera especial en las funciones del kernel del sistema operativo. OpenVPN opera en el anillo número 3 de la arquitectura segura en anillos del

sistema operativo, el cual es el nivel que realmente se busca. En ocasiones, para proveer de encriptación al enlace, las aplicaciones necesitan intervenir con el kernel del sistema operativo para ganar acceso a bajo nivel en la interfaz de red del enlace. Para estos casos, OpenVPN emplea “**interfaces virtuales**” del espacio de usuario para controlar y acceder sin la necesidad de depender del kernel. Estas interfaces virtuales ofrecen, a las VPN que usan el espacio de usuario, un punto extra de seguridad respecto a otras tecnologías VPN como IPSec, además de ofrecer más flexibilidad a la hora de exportar dicha herramienta a otros sistemas operativos y facilidad de instalación y uso en dichos sistemas operativos.

Por otra parte, OpenVPN permite encapsular el tráfico en paquetes que utilicen como protocolo de transporte **TCP o UDP**. En el apartado 2.4.4 de esta memoria, se describen las razones de por qué es mejor utilizar UDP como protocolo encapsulador que TCP. Además otra característica muy interesante en las versiones más recientes de OpenVPN es la posibilidad de utilizar un único puerto en el servidor para todas las conexiones VPN o de aguantar más de una conexión TCP.

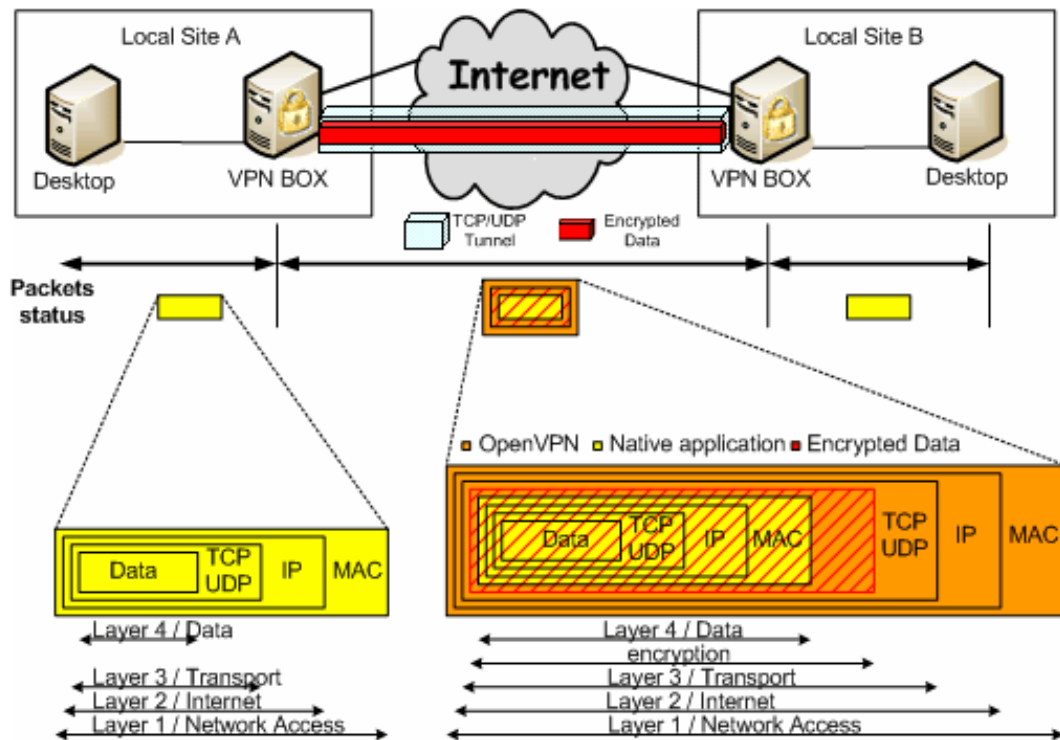


Figura 26. Formato del Paquete Encapsulado por OpenVPN.

De entre todas las aplicaciones y características que OpenVPN ofrece se pueden citar las siguientes:

- Tunelizar cualquier subred IP o adaptador de red virtual a través de un solo puerto TCP o UDP.
- Posibilidad de implementar dos modos básicos, “bridge” o “tunnel”, en la capa 2 o capa 3 respectivamente, con lo que se logran túneles capaces de enviar información en otros protocolos no IP como IPX o broadcast (Netbios).

- Configurar granjas de servidores VPN escalables y con balanceo de carga. Cada servidor puede mantener miles de conexiones dinámicas de clientes VPN.
- Uso de todos los protocolos de cifrado, autenticación y certificación que ofrece la librería OpenSSL para proteger el tráfico de la VPN durante el tránsito por Internet. Esta librería nos ofrecerá muchos algoritmos de cifrado, tamaños de clave, o resúmenes HMAC (para la integridad de los mensajes).
- OpenVPN permite escoger entre el mecanismo de encriptación convencional mediante claves compartidas o el mecanismo de encriptación de clave pública basado en certificados mediante el intercambio de claves dinámica de TLS.
- Uso de compresión en tiempo real y gestión del tráfico para manejar la utilización del ancho de banda.
- Soporte para proxy. Funciona a través de proxy y puede ser configurado para ejecutar como un servicio TCP o UDP y además como servidor (simplemente esperando conexiones entrantes) o como cliente (iniciando conexiones).
- Establecer túneles donde los extremos utilizan direcciones IP dinámicas con técnicas como DHCP.
- Ningún problema con NAT. Tanto los clientes como el servidor pueden estar en la red usando solamente direccionamiento IP privado.
- Crear puentes ethernet seguros utilizando para ello adaptadores ethernet virtuales (o “tap”).
- Controlar y monitorizar conexiones OpenVPN mediante interfaces gráficas de usuario (GUI) para diferentes plataformas (Windows, Linux, MAC OS X,...).
- Permite su instalación en casi cualquier plataforma. Tanto la instalación como su uso son increíblemente simples.
- Una ventaja es que su diseño es modular: se basa en un excelente diseño modular con un alto grado de simplicidad tanto en seguridad como en red.
- Ofrece una alta flexibilidad y posibilidades de extensión mediante scripting. OpenVPN ofrece numerosos puntos para ejecutar scripts individuales durante su arranque.
- Las interfaces virtuales (tun0, tun1,...) permiten la implementación de reglas de firewall muy específicas.
- Solo se ha de abrir un puerto para permitir conexiones, dado que desde OpenVPN 2.0 se permiten múltiples conexiones en el mismo puerto TCP o UDP.

Tras describir las funcionalidades y características de OpenVPN se va a hacer una pequeña reflexión sobre lo que distingue a OpenVPN de otros paquetes y herramientas para crear VPN e implementar su seguridad:

- Los principales puntos fuertes de OpenVPN son la portabilidad multiplataforma para una gran variedad de sistemas operativos del mundo de la computación, sin olvidar su excelente estabilidad, escalabilidad para cientos o miles de clientes, su relativamente fácil instalación y su soporte para IP dinámicas y NAT.

- OpenVPN, a diferencia de otras herramientas, provee de un marco extensible para la implementación de VPN que ha sido diseñado para facilitar su personalización, de manera que proporciona, entre otros recursos, la capacidad de distribuir un paquete de instalación personalizada a cada cliente, o la capacidad de implementar más de un método de autenticación a través de plugins, como por ejemplo, autenticación con certificados X.509 combinada con autenticación basada en el paquete PAM (Pluggable Authentication Method).
- Ofrece una interfaz de manejo, el cual, puede ser utilizada de manera remota permitiendo el control o manejo centralizado del proceso OpenVPN. Además dicha interfaz de manejo puede ser utilizada para el desarrollo de interfaces gráficas de usuario (GUI) o de aplicaciones basadas en Web (Webmin).
- OpenVPN utiliza un modelo de seguridad robusto apoyado por el sector para proteger contra ataques tanto pasivos como activos. El modelo de seguridad de OpenVPN se basa en la utilización de SSL/TLS para la sesión de autenticación y un modelo muy similar al del protocolo ESP de IPSec para cifrar el túnel de transporte sobre UDP (por defecto en OpenVPN). Además OpenVPN utiliza para la infraestructura de clave pública (PKI) certificados X509 para la autenticación, el protocolo TLS para el intercambio de claves, la interfaz EVP de funciones de OpenSSL para el cifrado de los datos del túnel y utiliza algoritmos HMAC para la autenticación de los datos del túnel.
- Al contrario de otras herramientas o tecnologías para la implementación de VPN, OpenVPN se diseñó pensando en la portabilidad para muchos sistemas operativos. En la actualidad OpenVPN se puede ejecutar en entornos Linux, Solaris, OpenBSD, FreeBSD, NetBSD, Mac OS X y Windows 2000/XP y Vista (existe incluso una versión de OpenVPN para Symbian OS en desarrollo). Además, debido a que OpenVPN se ha escrito en el espacio de usuario bastante más que en el núcleo y no se ha modificado la capa IP, los esfuerzos para el diseño de su portabilidad a otras plataformas están bastante simplificados.
- A diferencia de otras tecnologías, como el caso de IPSec, OpenVPN es fácil de usar. En general, se puede crear y configurar un túnel mediante un solo comando y, opcionalmente, sin ningún tipo de archivo necesario. Además OpenVPN está muy bien documentado.
- OpenVPN se ha construido con un diseño firmemente modular. Todo lo relacionado con la encriptación está soportado por la librería OpenSSL y todo lo relacionado con la funcionalidad de los túneles IP está proporcionado por el adaptador virtual TUN/TAP. Los beneficios de esta modularidad pueden ser vistos, por ejemplo, en el caso de que OpenVPN puede enlazarse dinámicamente con una nueva versión de la librería OpenSSL e, inmediatamente, tener acceso a cualquier función ofrecida por esta nueva versión. Por ejemplo, cuando OpenVPN es implementada junto con la última versión de la librería OpenSSL (0.9.7), se puede acceder automáticamente a nuevos algoritmos de cifrado como AES-256 (Advanced Encryption Standard con 256 bits de clave) y, también, a las nuevas capacidades de OpenSSL que permiten la utilización de nuevos aceleradores hardware para optimizar el cifrado, descifrado y los métodos de autenticación. La portabilidad de OpenVPN hacia otros entornos también se ve facilitada gracias a esta modularidad.

- OpenVPN es rápido, ejecutándose en una distribución de Linux Redhat 7.2 en una máquina Pentium II a 266 Mhz, utilizando una sesión de autenticación basada en TLS, el algoritmo de cifrado Blowfish con 128 bits de tamaño de clave, el método SHA1 para la autenticación de los datos del túnel, un túnel mediante el adaptador virtual TUN/TAP y una sesión FTP con un gran número de archivos precomprimidos, OpenVPN puede lograr una velocidad de transferencia de 1,455 megabytes por segundo de tiempo de CPU.
- Mientras que OpenVPN ofrece multitud de opciones para el control de los parámetros de seguridad del túnel VPN, también proporciona opciones para dar protección en la seguridad a nivel de usuario, con comandos como "--chroot" para restringir la parte del sistema de ficheros al que OpenVPN puede acceder, "--user" y "--group" para disminuir los privilegios al arrancar el sistema, y "--mlock" para asegurar que las claves y los datos del túnel nunca se guardarán en el disco donde después puedan ser recuperados.

En la tabla que se muestra a continuación se muestra una comparativa entre OpenVPN y su más fuerte tecnología de VPN competidora, IPSec:

IPsec	OpenVPN
Estándar de la tecnología VPN	Aun desconocida y no compatible con IPSec
Plataformas de hardware (dispositivos, aparatos)	Solo en computadoras, pero en todos los sistemas operativos disponibles
Tecnología conocida y probada	Tecnología nueva y aun en crecimiento
Muchas interfaces gráficas disponibles	Sin interfaces gráficas profesionales, aunque ya existen algunos proyectos prometedores
Modificación compleja de la pila IP	Tecnología sencilla
Necesidad de modificaciones críticas al kernel	Interfaces de red y paquetes estandarizados
Necesidad de permisos de administrador	Ejecuta en el espacio del usuario y puede ser chroot-ed
Diferentes implementaciones de distintos proveedores pueden ser incompatibles entre si	Tecnologías de cifrado estandarizadas
Configuración compleja y tecnología compleja	Facilidad, buena estructuración, tecnología modular y facilidad de configuración

Curva de aprendizaje muy pronunciada	Fácil de aprender y éxito rápido para principiantes
Necesidad de uso de muchos puertos y protocolos en el firewall	Utiliza solo un puerto del firewall
Problemas con direcciones dinámicas en ambas puntas	Trabaja con servidores de nombres dinámicos como DynDNS o No-IP con reconexiones rápidas y transparentes
Problemas de seguridad de las tecnologías IPsec	SSL/TLS como estándar de criptografía
	Control de tráfico (Traffic shaping)
	Velocidad (más de 20 Mbps en máquinas de 1Ghz)
	Compatibilidad con firewall y proxies
	Ningún problema con NAT (ambos lados puede ser redes NATeadas)
	Posibilidades para hackers y road warriors

2.2.- OpenVPN y Paquetes Necesarios

Como se ha comentado en el apartado anterior la instalación, configuración y puesta en marcha de OpenVPN es bastante fácil en comparación con otras herramientas, como por ejemplo FreeS/Wan. Aunque OpenVPN puede utilizarse en los sistemas operativos más conocidos en el mundo de la informática, en este apartado se va a comentar el sencillo proceso de instalación de OpenVPN en entornos Linux y Windows y la dependencia de OpenVPN con otros paquetes y herramientas que también tendrán que estar instaladas en la misma máquina. Estas herramientas serán detalladas en apartados posteriores para su mejor comprensión.

La herramienta OpenVPN debe ser instalada tanto en las máquinas que hagan de servidor como en las que hagan de clientes. No existe una versión para clientes y otra para servidores, desde que OpenVPN se instala en el PC, proporciona tanto las funciones del cliente como las del servidor con sus respectivos comandos y directivas.

OpenVPN corre casi completamente en el espacio de usuario no exigiendo ninguna modificación ni componente en el kernel del sistema operativo que no sean los

controladores virtuales “TUN/TAP drivers” disponibles para muchas plataformas tales como Windows, Linux, MAC OS X y variantes de BSD.

Para instalar OpenVPN en una máquina primero hay que saber que OpenVPN puede instalarse:

- Con las librerías “crypto” y “SSL” de OpenSSL (versión 0.9.6 o mayor), ofreciendo autenticación mediante certificados, cifrado con clave pública e intercambio dinámica de claves basada en el protocolo TLS.
- Con la librería “crypto” de OpenSSL únicamente, pudiendo utilizar cifrado convencional basado en claves estáticas compartidas por ambos extremos de la comunicación.
- Sin ninguna librería de seguridad OpenSSL, pudiendo implementar túneles no seguros, sin cifrado de los datos del túnel.

OpenVPN también puede ser vinculado con la librería de compresión de datos en tiempo real LZO (Lempel Ziv Oberhumer). De esta manera OpenVPN soporta compresión con adaptación en tiempo real, que significa que permitirá realizar la compresión sólo cuando flujo de datos del túnel pueda ser comprimido.

2.2.1.- Los Archivos de Configuración de OpenVPN

Una vez instalada la herramienta, OpenVPN se puede ejecutar mediante directivas por consola, donde cada directiva o comando debe de ir con dos guiones (“--comando y parámetros del comando”) o mediante un fichero de configuración con la directiva “--config ruta_del_fichero”.

Una de las pocas diferencias que existen entre OpenVPN en Windows y OpenVPN en cualquier otra plataforma son las extensiones de los ficheros de configuración. Mientras que en Windows, la extensión de un fichero de configuración de OpenVPN es “ovpn” en cualquier otra plataforma como puede ser Linux, por ejemplo, la extensión del fichero de configuración de OpenVPN es “conf”. Existen otras pequeñas diferencias, que no entraremos en detalle, a la hora de utilizar OpenVPN en plataformas Windows más relacionadas con las diferencias que existen entre la consola de una plataforma Windows y cualquier consola de una plataforma basada en UNIX o BSD. Por ejemplo en Windows, a diferencia de las otras plataformas, para ejecutar un fichero de configuración de OpenVPN no es necesario poner la directiva “--config”, simplemente la ruta donde se encuentra el fichero de configuración con extensión “ovpn”.

Por ejemplo, para ejecutar un fichero de configuración en Linux, una vez situado en el directorio donde se encuentra dicho fichero, se escribe la siguiente línea en consola:

```
openvpn --config nombre_del_fichero
```

Por otro lado, si utilizamos Windows, una vez situados en el directorio donde se encuentra dicho fichero, se escribe la siguiente línea en consola:

`openvpn nombre_del_fichero`

En ambos casos, en el fichero de configuración se debe encontrar, línea por línea, los comandos de OpenVPN sin utilizar los dos guiones delante.

Según la arquitectura de la VPN que se quiera implementar con OpenVPN se tendrá que hacer un fichero de configuración en modo cliente, en modo servidor de túneles o prescindiendo de ninguno de estos modos, como un simple extremo de la comunicación. En cualquiera de estos casos, cada máquina ha de ejecutar un fichero de configuración independientemente de los scripts o plugins que también deba ejecutar.

Un ejemplo muy sencillo para implementar un túnel en ambos sentidos sin cifrado ni autenticación, mediante comandos en consola (sin utilizar ficheros de configuración) sería el que se muestra a continuación:

```
openvpn --remote dominio1.com --dev tun0 --ifconfig 10.8.0.1 10.8.0.2
openvpn --remote dominio2.com --dev tun0 --ifconfig 10.8.0.2 10.8.0.1
```

Como se puede ver los extremos del túnel tienen las direcciones IP 10.8.0.1 y 10.8.0.2 y las interfaces virtuales TUN/TAP trabajan en el modo túnel con el nombre “tun0” en ambas máquinas.

El mismo ejemplo, pasado a ficheros de configuración, con nombres Ejemplo1A.conf (.ovpn en Windows) y Ejemplo1B.conf (.ovpn en Windows) tendría la siguiente forma en un fichero de configuración:

Ejemplo1A.conf	Ejemplo1B.conf
<code>remote dominio1.com</code>	<code>remote dominio2.com</code>
<code>dev tun0</code>	<code>dev tun0</code>
<code>ifconfig 10.8.0.1 10.8.0.2</code>	<code>ifconfig 10.8.0.2 10.8.0.1</code>

Figura 27. Ejemplo de Archivos de Configuración

Una vez creados los archivos de configuración hay que ejecutarlos mediante el programa OpenVPN en un terminal de consola en ambos extremos del túnel:

```
openvpn --config Ejemplo1A.conf
openvpn --config Ejemplo1B.conf
```

2.2.2.- Los controladores virtuales TUN/TAP y VTUN

Como bien se ha comentado en esta memoria OpenVPN depende de los **controladores virtuales TUN/TAP** para poder establecer túneles punto a punto entre los dos extremos de la comunicación. Estos túneles fueron desarrollados por Maxim Krasnyansky en

1999 contenidos en la herramienta para creación de adaptadores virtuales **VTUN**, de libre distribución. Los túneles virtuales TUN/TAP han sido incorporados en el kernel de Linux a partir de las **versiones 2.4.x** de estos kernels. Para poder implementar estos túneles en versiones anteriores de kernel de Linux o en otros sistemas operativos (como MAC OS X, o variantes BSD), sin incluir plataformas Windows, hace falta instalar la herramienta que es capaz de implementar estos túneles, VTUN, creada, como ya se ha comentado, por Maxim Krasnyansky. El caso de las plataformas Windows ha sido un tanto especial, ya que OpenVPN en sus primeras versiones no era compatible con ningún sistema operativo tipo Windows, debido a su incapacidad para implementar estos túneles virtuales TUN/TAP (en concreto a su incapacidad de implementar enlaces punto a punto reales), además la herramienta VTUN, no se había diseñado para entornos Windows. Sin embargo, en el 2003 el equipo de desarrolladores de OpenVPN en colaboración con su autor, James Yonan, desarrolló una herramienta para plataformas Windows (2000/XP/Vista) que permitía crear, en dicho sistema operativo, interfaces virtuales TUN/TAP, con el fin de hacer compatible con la plataforma Windows la herramienta OpenVPN y conocidos en dicha plataforma como TAP-Win32. Desde entonces, los drivers TAP-Win32 ha sido desarrollados y depurados por el equipo de desarrolladores de dispositivos y drivers de Microsoft y el paquete auto instalable de OpenVPN para plataformas Windows incorpora, además de OpenVPN, los túneles TAP-Win32 para Windows 2000, XP o Vista.

La idea de diseñar una utilidad como son las interfaces virtuales e integrarlos en el kernel de ciertos sistemas operativos viene en concordancia con la aparición de las VPN. IPSec y otras tecnologías como SSL/TLS, PPTP o L2TP permiten la implementación de VPN, pero no si el tiempo para crearlas es muy reducido. Por ejemplo, para un tele-trabajador que se desplace muy a menudo de país a país le supone un beneficio en tiempo la implementación de un túnel con la sede central mediante estos adaptadores virtuales ya que en el caso de optar por otra opción, como IPSec, el tiempo de aprendizaje e implementación se incrementaría considerablemente.

El modelo que, básicamente, utiliza OpenVPN para implementar una VPN se basada en utilizar el espacio de usuario y enlazar una interfaz de red virtual punto a punto llamada “tun” (lo típico, dependiendo del sistema operativo, es llamarlo tunX, donde X corresponde a un número) con otra interfaz de red virtual punto a punto (“tun”) remota. Un adaptador de red virtual “tun” a su misma vez se podría ver como un enlace punto a punto entre la tarjeta de red del PC y el sistema operativo. La interfaz “tun” en lugar de entregar los bits al medio físico los entrega al espacio de usuario del sistema operativo y éste abre, lee y/o escribe datos de paquete IP en la interfaz “tun” como si de un fichero se tratase.

Cuando una interfaz virtual TUN/TAP recibe el nombre “tun” significa que está trabajando en modo túnel (o tunnel) punto a punto, enlazando con otra interfaz virtual del mismo tipo. Por otro lado, si una interfaz virtual TUN/TAP recibe el nombre de “tap”, significa que está trabajando en modo bridge, emulando de esta manera una interfaz de red Ethernet en lugar de una interfaz punto a punto. En la mayoría de los casos, y en todos los escenarios de este proyecto se implementarán túneles con el modo tunnel, utilizando interfaces virtuales “tun”.

Por ejemplo, supongamos que se tienen dos interfaces virtuales “tun”, una en una máquina A y otra en otra máquina B. Para poder transferir bits de una máquina a otra se tendrían que copiar, en la máquina A, desde el dispositivo virtual al socket de red del sistema operativo, y una vez enviados a la máquina B, en ésta, desde el socket del sistema operativo al dispositivo virtual “tun”. Sin embargo, el cifrado y la autenticación no se han incorporado aún. La manera en que OpenVPN añade la seguridad a los túneles se describe, más adelante, en esta misma memoria, en el apartado 3.2.

Una de las ventajas de utilizar estos dispositivos de red virtuales es que permiten aplicarles rutas y reglas de firewall como si de una tarjeta de red física Ethernet se tratase. Por ejemplo, si en un PC se tiene una tarjeta de red física con el nombre de “eth0” y se ha implementado un dispositivo de red virtual con el nombre de “tun0”, se podrán aplicar las mismas o diferentes rutas y reglas de firewall a uno u otro dispositivo de red.

Otra ventaja de utilizar el concepto de dispositivo de red virtual TUN/TAP es que se desplaza la complejidad de la VPN al espacio de usuario, separando lógicamente los componentes de red de los componentes de cifrado y seguridad, obteniendo, de esta manera, un código que pueda ser exportado a diferentes plataformas, y obteniendo también una interfaz intuitiva al usuario final.

VTun es la herramienta que permite implementar estos adaptadores de red virtuales para casi cualquier sistema operativo en caso de no disponer de un kernel con una versión igual o mayor a la 2.4.x. De momento solo se han comentado las características de esta herramienta en relación con la creación de túneles del tipo “tun” o de adaptadores virtuales Ethernet tipo “tap”, pero VTUN o su implementación en el kernel 2.4.x tienen otras características igual de importantes:

- Herramienta de fácil uso para crear túneles virtuales en redes TCP/IP, pudiendo implementar varios tipos de estos túneles con diferentes características de cifrado y compresión.
- Permite gran variedad de configuraciones y es muy utilizado para crear VPN.
- Soporta compresión de datos mediante las librerías “Zlib” (solo para TCP) y “LZO” (para TCP y UDP).
- Permite implementar diferentes tipos de autenticación, cifrado mediante Blowfish con clave de 128 bits y funciones Hash MD5.
- Entre los tipos de túneles que se pueden implementar se encuentran:
 - **Túneles IP** (tun): para crear túneles IP punto a punto.
 - **Túneles Ethernet** (ether o tap): muy utilizado, ya que soporta multitud de protocolos de red, como IP, IPX, Appletalk,...
 - **Túneles serie** (tty): soportan todos los protocolos que trabajan en líneas serie como PPP, SLIP,...
 - **Tuberías** (pipe): soporta todos los programas que trabajan sobre tuberías de UNIX, como SSH.

- Permite crear túneles utilizando como protocolos de transporte tanto TCP como UDP.
- Está disponible para varios sistemas operativos entre los que se encuentran Linux, FreeBSD (y otros clones de BSD), OpenBSD, MAC OS X, Solaris,...

2.2.3.- Seguridad con OpenSSL

OpenSSL es un proyecto de software de libre distribución desarrollado por los miembros de la comunidad Open Source para libre descarga y está basado en SSLeay y desarrollado por Eric Young y Tim Hudson. Consiste en un robusto paquete de herramientas de administración y librerías relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenVPN, OpenSSH y navegadores web (para acceso seguro a sitios HTTPS). Estas herramientas ayudan al sistema a implementar el protocolo Secure Socket Layer (SSL versiones 2 o 3), así como otros protocolos relacionados con la seguridad, como el protocolo Transport Layer Security (TLS versión 1). Este paquete de software es importante para cualquiera que esté planeando usar cierto nivel de seguridad en su máquina con un sistema operativo libre basado en GNU/Linux. OpenSSL también nos permite crear certificados digitales que podremos aplicar a nuestro servidor, por ejemplo Apache.

OpenSSL soporta un gran número de algoritmos criptográficos diferentes según la finalidad:

- Algoritmos de cifrado: Blowfish, AES, DES, RC2, RC4, RC5, IDEA, Camellia.
- Algoritmos para funciones hash: MD5, SHA, MD2, MDC-2.
- Algoritmos de intercambio de clave publica: RSA, Diffie-Hellman, DSA, curva elíptica.

Además OpenSSL proporciona las herramientas y funciones para:

- Generar distintas claves para distintos algoritmos de cifrado.
- Generar claves asimétricas para los algoritmos habituales (RSA, Diffie-Hellman, DSA).
- Generar números aleatorios y pseudos aleatorios.
- Utilizar los algoritmos para firmar, certificar y revocar claves.
- Manejar formatos de certificados existentes en el mundo (X.509, PEM, PKCS7, PKCS8, PKCS12).

OpenVPN y otras muchas aplicaciones (como OpenSSH) y protocolos (HTTPS o sTelnet) utilizan OpenSSL para implementar su modelo de seguridad y proteger, así, de ataques activos y pasivos. Dicho modelo de seguridad esta basado, como ya se ha mencionado a lo largo de esta memoria, en la utilización del protocolo de seguridad SSL/TLS.

OpenSSL es un conjunto de herramientas de criptografía para la aplicación de Secure Socket Layer (SSL v2/v3) y Transport Layer Security (TLS v1) y para los protocolos de red relacionados con las normas de criptografía exigidas por ellos. OpenSSL se compone de la herramienta de línea de comandos, “OpenSSL”, y de dos librerías, la librería “SSL” y la librería “Crypto”.

El programa “openssl” es una herramienta de línea de comandos para el uso de diversas funciones de criptografía de OpenSSL la biblioteca criptográfica del kernel. El programa “openssl” ofrece una rica variedad de comandos, cada uno de los cuales tiene a su vez una gran variedad de opciones y argumentos que pasarles. Se puede utilizar para realizar los siguientes propósitos:

- Creación y gestión de claves privadas, claves públicas y los parámetros de clave pública.
- Operaciones de criptografía de clave pública.
- Creación de certificados X.509, peticiones de firma de certificados (CSR, Certificate Signing Request) y listas de revocación de certificados (CRL, Certificate Revocation List).
- Cálculo de resúmenes de mensajes.
- Cifrado y descifrado mediante algoritmos de cifrado.
- Tests en clientes y servidores que utilicen SSL/TLS.
- Manejo de correo S/MIME firmado o cifrado.
- Marcas de tiempo en solicitudes, generaciones y validaciones

Una de las librerías de OpenSSL es la librería “ssl” (archivo libssl.a), la cual implementa los protocolos Secure Socket Layer (SSL versiones 2 y 3) y Transport Layer Security (TLS versión 1) y el código necesario para poder proporcionar SSL (versión 2 o 3) y TLS (versión 1) en un cliente o en un servidor. En una primera instancia, la biblioteca “ssl” debe ser inicializada mediante la función `SSL_library_init()`, la cual registra los algoritmos de cifrado y de resumen permitidos. Tras inicializar la librería “ssl”, una serie de objetos `SSL_CTX` son creados como frameworks para establecer conexiones TLS/SSL mediante la función `SSL_CTX_new()`. A estos objetos `SSL_CTX` se le pueden pasar varias opciones relacionadas con certificados, algoritmos, etc. Cuando una conexión SSL/TLS ha sido creada, esta conexión puede ser asignada a un objeto SSL. Estos objetos SSL son creados tras utilizar las funciones `SSL_new()`, `SSL_set_fd()` o `SSL_set_bio()` y, tras su creación, pueden asociarse dichos objetos con la conexión de una red. Una vez se ha establecido la conexión, el protocolo handshake, de TLS/SSL, comienza a funcionar utilizando las funciones `SSL_accept()` o `SSL_connect()` respectivamente. Otras funciones de gran utilidad que proporciona la librería “ssl” son la función `SSL_read()` y la función `SSL_write()` para leer o escribir datos en la conexión TLS/SSL establecida. Además se puede utilizar la función `SSL_shutdown()` para terminar la conexión TLS/SSL establecida.

La otra librería utilizada por OpenSSL es la librería “crypto” (archivo libcrypto.a), que implementa un gran número de algoritmos criptográficos utilizados en muchos estándares de Internet. Los servicios proporcionados por esta librería son utilizados por las implementaciones de SSL/TLS de OpenSSL y también son utilizados por otras aplicaciones criptográficas como, por ejemplo, OpenSSH, OpenPGP y OpenVPN.

La librería “crypto” consiste en varias sub-librerías que implementan los algoritmos criptográficos de manera individual. La funcionalidad de esta librería incluye cifrado simétrico, criptografía de clave pública y negociación de claves, manejo de certificados, funciones criptográficas hash y generación de números pseudo-aleatorios. Algunas de las funciones que proporciona la librería “crypto” son:

- Cifrado simétrico: blowfish(), cast(), des(), idea(), rc2(), rc5()...
- Criptografía de clave pública y negociación: dsa(), dh(), rsa()...
- Certificados: x509(), x509v3()...
- Códigos de autenticación y funciones hash: hmac(), md2(), md4(), md5(), mdc2(), ripemd(), sha()...
- Funciones auxiliares: err(), threads(), rand(), openssl_version_number()...
- Codificación de datos y entrada/salida: asn1(), bio(), evp(), pem(), pkcs7(), pkcs12()...
- Funciones internas: bn(), buffer(), lhash(), objetcs(), snack(), txt_db()...

Como ya se ha dicho en esta memoria, OpenVPN, puede incluir o no dichas librerías de OpenSSL, según las prestaciones que se quiera ofrecer de OpenVPN. Por tanto, OpenVPN puede instalarse:

- Con las librerías “crypto” y “ssl” de OpenSSL, ofreciendo autenticación mediante certificados, cifrado con clave pública e intercambio dinámica de claves basada en el protocolo TLS.
- Con la librería “crypto” de OpenSSL únicamente, pudiendo utilizar cifrado convencional basado en claves estáticas compartidas por ambos extremos de la comunicación.
- Sin ninguna librería de seguridad OpenSSL, pudiendo implementar túneles no seguros, sin cifrado de los datos del túnel.

OpenSSL es compatible con un gran número de sistemas operativos, entre los que se encuentran sistemas operativos tipo Windows, UNIX/Linux o también MAC OS X. Según el sistema operativo en el que se instale OpenVPN será necesario o no una instalación a parte de la herramienta OpenSSL. Por ejemplo, en sistemas operativos Windows, el ejecutable que se utiliza para instalar OpenVPN también instala las librerías y programas de OpenSSL necesarios para OpenVPN. Sin embargo, en sistemas operativos tipo UNIX para poder instalar OpenVPN es necesaria una instalación previa de OpenSSL, en el caso de que se quiera incluir alguna de las dos librerías indicadas con anterioridad de OpenSSL. En cualquier distribución de Linux se puede instalar en cuatro pasos:

- `./config`
- `make`
- `make test`
- `make install`

2.2.4.- Compresión con LZO

LZO es una librería multiplataforma de compresión de datos sin pérdidas escrito en ANSI C. LZO ofrece bastante velocidad en la compresión de datos y mucha más velocidad en la descompresión de los datos, que es donde más destaca por su gran velocidad. Además no requiere memoria para la descompresión de los datos. Esta librería, sus algoritmos e implementaciones que la componen se han creado bajo la GNU (General Public License), por lo que es una fuente de código abierto. LZO es el acrónimo de Lempel-Ziv-Oberhumer.

LZO es una librería de compresión de datos, la cual es muy apropiada para compresión o descompresión de datos en tiempo real. Tanto el código de dicha librería como los datos comprimidos están codificados en ANSI C con la finalidad de poder ser exportados a una gran variedad de plataformas.

Por otro lado, existe un software libre que implementa la librería LZO y se llama lzop. En este proyecto final de carrera se ha tenido que utilizar la librería LZO, ya que es requisito para poder instalar OpenVPN, por lo que no se ha utilizado el software lzop. Dicho software ha sido programado en varios lenguajes de programación, entre los que se encuentran Perl, Python o Java.

LZO implementa varios algoritmos de compresión y descompresión de datos con algunas de las siguientes características:

- La descompresión es simple y muy rápida.
- No requiere memoria para la descompresión.
- La compresión de los datos también es bastante rápida.
- Requiere de algún tipo de buffer de 64 kB de memoria para la compresión. Existen niveles de compresión de los datos más bajos en el que, únicamente, se necesitan 8 kB de memoria.
- No necesita ningún tipo de buffer o memoria para la descompresión además de los buffers fuente y destino.
- Permite al usuario realizar un ajuste entre calidad de compresión y velocidad de compresión sin que la velocidad de compresión se vea afectada.
- Proporciona niveles de compresión para la realización de una pre-compresión de los datos con el que se logra un ratio de compresión totalmente competitivo.

- El algoritmo es sin pérdidas.
- El algoritmo es thread-safe.
- LZO soporta superposición en las compresiones.

LZO utiliza un algoritmo de compresión en bloque, de manera que comprime y descomprime bloques de datos. Cuando LZO trata datos que no pueden comprimirse más, expande los datos de entrada, de cómo máximo 16 bytes, por datos de entrada de 1024 bytes.

Otro aspecto importante es la rapidez y depuración de esta librería. Cuando hay muchos ficheros de objetos, en su mayoría independientes unos de otros, las dimensiones de un ejecutable que esté relacionado con la biblioteca LZO suele ser bastante baja (de unos pocos kB), debido a que el enlazador de solo añade los módulos que se necesitan utilizar realmente, consiguiendo así ejecutables más ligeros y una mayor rapidez.

Existen varios algoritmos que han sido poco a poco sustituidos por otros mas rápidos y eficaces, entre los que podemos encontrar: LZO1, LZO1A, LZO2A, LZO1B, LZO1F, LZO1X, LZO1Y y LZO1Z. De entre todos los algoritmos listados la mejor opción y más utilizada por la mayoría de programas que incluyen LZO para la compresión de los datos es el algoritmo LZO1X, ya que se logra un mejor ratio de compresión en la mayoría de los ficheros y una velocidad muy alta en la descompresión. Por ejemplo, la versión de C del algoritmo LZO1X-1 es cerca de 4 a 5 veces más rápido que el mejor algoritmo utilizado por zlib, otra librería de compresión y descompresión de datos muy extendida.

El funcionamiento básico de LZO, asumiendo que se utilizará el algoritmo LZO1X-1 para la compresión y descompresión de los datos, es muy simple, y se compone de los siguientes pasos:

- Compresión de los datos:
 - Añadir las cabeceras de LZO1X: include <lzo/lzo1x.h>.
 - Llamar a la función lzo_init().
 - Comprimir los datos con la función lzo1x_1_compress().
 - Enlazar nuestra aplicación con la librería LZO.
- Descompresión de los datos:
 - Añadir las cabeceras de LZO1X: include <lzo/lzo1x.h>.
 - Llamar a la función lzo_init().
 - Descomprimir los datos con la función lzo1x_decompress().
 - Enlazar nuestra aplicación con la librería LZO.

2.2.5.- Scripts y Plugins en OpenVPN

OpenVPN permite utilizar **scripts** para conseguir mayor flexibilidad y posibilidades de extensión. En este proyecto final de carrera se han utilizado algunos de estos scripts, escritos en lenguaje C y Perl, y otros scripts de consola, para su desarrollo. Entre los plugins y scripts para OpenVPN más importantes, por su funcionalidad, podemos encontrar los siguientes:

- Scripts relacionados con la ejecución de los procesos de cada túnel establecido y scripts para el arranque y parada del proceso del servidor OpenVPN al iniciar o apagar el sistema operativo de la máquina que hace de servidor.
- Scripts para la configuración de las reglas de firewall con iptables, diferenciando entre las reglas de firewall de la interfaz de red virtual del túnel y las reglas de firewall de la interfaz de red física de la máquina
- Scripts para añadir rutas para cada extremo del túnel, de tal forma que se tendrá una tabla de rutas para cada interfaz de red virtual de ambos extremos del túnel y una tabla de rutas para cada interfaz de red física de cada máquina de los dos extremos del túnel.
- Scripts y ejecutables (para Windows) para generar claves, certificados, parámetros necesarios para el intercambio de claves, listas de revocación de certificados y nuestra propia autoridad certificadora (CA) utilizando, para ello, las librerías y comandos de OpenSSL.
- Scripts para arrancar y parar los túneles mediante el demonio extendido de Internet o xinetd (eXtended Internet Daemon). El servicio xinetd puede usarse para instanciar automáticamente un demonio OpenVPN al recibir un datagrama inicial de un extremo remoto.
- Scripts de servidor de OpenVPN para comprobar los certificados de los clientes OpenVPN por medio del “Common Name” (CN) de dichos certificados.
- Plugin para habilitar el funcionamiento de las secuencias de comandos con privilegios de root incluso si las directivas --user/--group/--chroot se han utilizado para establecer privilegios de root o cambiar el entorno de ejecución.
- Plugin para añadir un factor extra en seguridad mediante la autenticación por login y password utilizando para ello el paquete PAM (Pluggable Authentication Module). Esta opción no está disponible si utilizamos un servidor con sistema operativo Windows. Este plugin ha sido utilizado en este proyecto final de carrera para que los clientes puedan autenticarse al servidor introduciendo su login y su password cuando quieren utilizar un servicio protegido mediante OpenVPN. En esta memoria se ha escrito un apartado para comentar las características del paquete PAM y también los servicios y módulos que este paquete proporciona además del servicio de login y password.
- Plugin para que el cliente de OpenVPN realice la autenticación mediante login y password utilizando para ello protocolo LDAP (Lightweight Directory Access Protocol), en concreto, mediante las librerías que se incluyen en el paquete OpenLDAP.

Existen varias directivas (que pueden verse en el man de OpenVPN) en OpenVPN que utilizan estos scripts para el establecimiento de rutas, arranque de servidores y clientes, cambios de IP, comprobación de certificados,... pero también se pueden cargar los módulos de los plugins mediante la directiva "--plugin", también utilizada en este proyecto.

2.2.6.- Autenticación Extra con PAM

Los programas que permiten a los usuarios acceder a un sistema deben verificar la identidad del usuario a través de un proceso llamado autenticación. Históricamente, cada programa tiene su forma particular de realizar la autenticación. En muchas distribuciones de Linux, muchos de esos programas son configurados para usar un proceso de autenticación centralizado llamado **Pluggable Authentication Modules (PAM)**.

PAM utiliza una arquitectura conectable y modular, que otorga al administrador del sistema una gran flexibilidad al establecer las políticas de autenticación para el sistema.

En la mayoría de los casos, el archivo de configuración por defecto PAM para una aplicación tipo PAM es suficiente. Sin embargo, algunas veces es necesario modificar el archivo de configuración. Debido a que un error en la configuración de PAM puede comprometer la seguridad del sistema, es importante comprender la estructura de estos archivos antes de hacer cualquier modificación.

PAM ofrece las ventajas siguientes:

- Un esquema de autenticación común que se puede usar con una gran variedad de aplicaciones.
- Permite gran flexibilidad y control de la autenticación para el administrador del sistema y el desarrollador de la aplicación.
- Los desarrolladores de aplicaciones no necesitan desarrollar su programa para usar un determinado esquema de autenticación. En su lugar, pueden concentrarse puramente en los detalles de su programa.

En distribuciones Linux, el directorio `/etc/pam.d/` contiene los archivos de configuración de PAM para cada aplicación tipo PAM. En versiones antiguas de PAM se utilizaba `/etc/pam.conf`, pero este archivo ya no se utiliza y `pam.conf` solamente es leído si el directorio `/etc/pam.d/` no existe.

Las aplicaciones tipo PAM o servicios tienen un archivo dentro del directorio `/etc/pam.d/`. Cada uno de estos archivos es llamado después del servicio para el cual controla el acceso.

Depende del programa tipo PAM definir el nombre de su servicio e instalar su archivo de configuración en el directorio `/etc/pam.d/`. Por ejemplo, el programa login define su nombre de servicio como `/etc/pam.d/login`.

Existen cuatro tipos de módulos PAM usados para controlar el acceso a los servicios. Estos tipos establecen una correlación entre los diferentes aspectos del proceso de autorización:

- **auth**: estos módulos autentifican a los usuarios pidiendo y controlando una contraseña. Los módulos con esta interfaz también pueden establecer credenciales, tales como pertenencias de grupo o tickets Kerberos.
- **account**: estos módulos controlan que la autenticación sea permitida (que la cuenta no haya caducado, que el usuario tenga permiso de iniciar sesiones a esa hora del día, etc.).
- **password**: estos módulos se usan para establecer y verificar contraseñas.
- **session**: estos módulos configuran y administran sesiones de usuarios. Los módulos con esta interfaz también pueden realizar tareas adicionales que son necesitadas para permitir acceso, como el montaje de directorios principales de usuarios y hacer el buzón de correo disponible.

Existe una gran cantidad de módulos que han sido creados por usuarios y desarrolladores de PAM. Por ejemplo, existen módulos de autenticación por PAM para RADIUS, LDAP, SSH, Apache y una infinidad de aplicaciones y protocolos en general. Como desventaja, hay que destacar que PAM no está disponible para sistemas operativos tipo Windows.

En este proyecto se ha utilizado un plugin que utiliza PAM para añadir autenticación mediante login y password. Para ello se utiliza un módulo llamado “`openvpn-auth-pam`”, que permite utilizar cualquier autenticación soportada por PAM (como LDAP, RADIUS y otros métodos normalmente utilizados en Linux) y poder ser utilizado por OpenVPN. Mientras que PAM ofrece autenticación mediante login/password, también puede ser combinado con la autenticación mediante certificados, ofreciendo así dos niveles de autenticación. Para ello, se necesita tener instalada una librería de PAM (que se puede encontrar en la mayoría de las distribuciones de Linux o bien se puede descargar de Internet) llamada “`PAM-devel`”.

2.3.- Modo de Funcionamiento de OpenVPN

OpenVPN puede trabajar en dos modos: modo “`tun`” (o modo Tunnel) o modo “`tap`” (o modo Bridge). Ambos modos utilizan el adaptador TUN/TAP detallado en el apartado 2.2.3, en concreto, utilizando el adaptador TUN para transmitir tráfico IP a lo largo del túnel o, por el contrario, utilizando el adaptador TAP para transmitir tráfico Ethernet por el túnel. Como sabemos, la configuración de estos adaptadores es muy fácil y solo requiere de un conjunto de comandos o de líneas introducidas en un fichero de

configuración. Una vez la interfaz TUN/TAP se ha establecido ya se puede hacer una comunicación mediante OpenVPN.

En este apartado se listarán las principales características de funcionamiento de ambos modos, sus principales ventajas y desventajas y las principales diferencias entre ambos modos en términos de configuración.

La principal diferencia entre los adaptadores tun y los adaptadores tap es que un adaptador tun es como un dispositivo virtual IP punto a punto entre los dos extremos de la comunicación (como si de una línea dedicada entre los dos extremos de la comunicación se tratase) y un dispositivo tap es como un dispositivo Ethernet virtual entre los dos extremos de la comunicación (como si se estableciese una red Ethernet los dos dispositivos de la comunicación).

Lo más normal es utilizar el modo tun para establecer un túnel IP entre ambos extremos de la comunicación, pero la gente que ejecuta aplicaciones que necesitan características típicas que puede ofrecer Ethernet (que no trabajan en una red exclusivamente IP) necesitarán entonces realizar un puente entre una red local física Ethernet mediante la utilización de un dispositivo tap virtual proporcionado por OpenVPN. Para ello se necesita un dispositivo tap en cada extremo de la comunicación. De esta manera OpenVPN permite encaminar tráfico Ethernet broadcast y tráfico no IP como Windows NetBios o IPX a través de la VPN establecida mediante los adaptadores. En caso de no necesitar ninguna característica especial de Ethernet lo más usual es utilizar el modo tun mediante los dispositivos virtuales tun de OpenVPN, además es el procedimiento más utilizado por los usuarios por su facilidad de configuración.

Para poder arrancar un dispositivo tun o tap en OpenVPN, lo más frecuente es utilizar la directiva `--dev tun|tap` en el fichero de configuración o en la consola de comandos. A veces puede ser interesante asignarle un número a dicho dispositivo virtual, pero no necesariamente deben coincidir el número de los dispositivos virtuales de ambos extremos de la comunicación, solo es una forma de identificación dentro de la misma máquina. Hay que destacar que no se pueden mezclar ambos dispositivos virtuales TUN/TAP, es decir, no se permite utilizar un dispositivo virtual tun en uno de los extremos y un dispositivo virtual tap en el otro de los extremos.

Generalmente, el modo Tunnel probablemente sea la mejor elección para la mayoría de usuarios de OpenVPN, ya que es más fácil de configurar y es más eficiente que el modo Bridge. Además, el modo Tunnel también proporciona más capacidad para realizar un control en el acceso a usuarios específicos. Normalmente se recomienda utilizar el modo Tunnel a menos que se necesiten características específicas del modo Bridge tales como:

- La VPN debe ser capaz de aguantar protocolos no IP como IPX.
- Se necesitan correr aplicaciones que dependen de redes broadcast sobre la VPN.

- Le interesa permitir la búsqueda de archivos compartidos de Windows a través de la VPN sin la creación de un servidor Samba o un servidor WINS.

2.3.1.- IP Tunneling

Como ya se ha comentado el modo tun o modo Tunnel establece un enlace IP punto a punto virtual mediante la utilización del dispositivo virtual tun que proporciona OpenVPN.

Las principales ventajas de utilizar el modo Tunnel o modo tun son:

- Eficiencia y mayor escalabilidad.
- Permite un establecimiento mejor de la MTU para una mayor eficiencia.

Entre las principales desventajas del modo Tunnel o modo tun podemos encontrar:

- En algunas configuraciones los clientes necesitan tener de un servidor WINS.
- Se deben establecer las rutas que unen cada subred.
- El software que dependa de tráfico broadcast no podrá ver a las máquinas pertenecientes a la red del otro lado de la VPN.
- Solo funciona para los protocolos de nivel de red IPv4 e IPv6 siempre que en ambos extremos del túnel se use el mismo protocolo.

2.3.2.- Ethernet Bridging

El modo Bridge o modo tap es capaz de unir, mediante un puente, dos redes locales Ethernet (separadas por Internet, por ejemplo), mediante el uso del dispositivo virtual tap que ofrece OpenVPN. Es decir, el modo Bridge es una técnica para crear redes de área local (LAN) virtuales.

Las principales ventajas de utilizar el modo Bridge o modo tap son:

- Permite al tráfico broadcast atravesar la VPN. Esto permite a software o protocolos que dependen del tráfico broadcast como NetBIOS poder comunicarse con ambas partes de la VPN.
- No se pueden configurar rutas.
- Trabaja con cualquier protocolo de red que pueda trabajar sobre Ethernet tal como IPv4, IPv6, IPX, AppleTalk, etc.
- Para usuarios “road warrior”, su configuración es relativamente fácil.

Entre las principales desventajas del modo Bridge o modo tap podemos encontrar:

- Menos eficiente y escalable que el modo Tunnel.

2.4.- Más Acerca del Funcionamiento de OpenVPN

En este apartado se va a realizar una visión técnica sobre la capa de cifrado de OpenVPN, para comprender con mayor detalle como se ha implementado la seguridad en OpenVPN y cuando interviene la suite de cifrado de OpenSSL. Además, para complementar, se va a estudiar con detalle como funciona el protocolo que utiliza OpenVPN y el formato de sus mensajes. Por último, se va a explicar como realiza OpenVPN la asignación de direcciones y se demostrará porque es mejor la utilización de UDP que la utilización de TCP como protocolo de transporte.

2.4.1.- Capa de Cifrado de OpenVPN

Como ya se ha dicho a lo largo de esta memoria, uno de los dichos de la seguridad informática es que la complejidad es el enemigo de la seguridad. Una manera de reducir el impacto de la complejidad del software sobre la seguridad del propio software en general, es forzar al tráfico entrante de la red a pasar a través de una especie de pasarela de seguridad que tiene un código más simple que las aplicaciones que corren detrás de dicho software. Un ejemplo de utilización de dicha estrategia de lo sencillo es el uso de firewalls. La clave es reducir el número de líneas de código que tienen que ser leídas cuando llegan paquetes no autenticados. Este menor número de líneas puede ser, después, examinado con mayor detalle para estudiar las vulnerabilidades. OpenVPN amplía el concepto de firewall, utilizando el comando `--tls-auth` para someter a los paquetes entrantes a la VPN a un test de firma digital antes de pasar por el código real de SSL/TLS. De esta manera OpenVPN puede contener tres niveles de seguridad:

- **Nivel 1:** utilizar autenticación HMAC basada en la opción `--tls-auth` para prevenir ataques de inyección de paquetes al subsistema SSL/TLS.
- **Nivel 2:** utilizar las herramientas de seguridad proporcionadas por SSL/TLS para una autenticación bidireccional entre cliente y emisor.
- **Nivel 3:** rebajar el nivel de privilegios del demonio de OpenVPN utilizando los comandos `--user/--group` para ayudar a prevenir la inyección exitosa de código.

Como ya se ha comentado en las características, OpenVPN tiene dos posibles modos de autenticación:

- Utilizar claves estáticas, es decir, utilizar claves pre-compartidas.
- Utilizar TLS, es decir, utilizar los protocolos de SSL/TLS y certificados para la autenticación y el intercambio de claves.

En el modo de claves estáticas, una clave pre-compartida es generada y compartida entre los dos participantes de OpenVPN antes de iniciar el túnel. Por defecto, en el modo de clave pre-compartida, la clave estática consta de 4 claves independientes: una HMAC enviada, una HMAC recibida, una clave para cifrar y una clave para descifrar (aunque también se puede establecer una clave con 2 claves: una HMAC para ambos extremos y una clave para cifrar/descifrar). Con el comando `--secret`, indicamos que se

quiere utilizar el modo de claves estáticas y según el parámetro que le pasemos a dicho comando se utilizarán 2 claves o 4 claves para dicha clave pre-compartida.

La ventaja de utilizar el modo de seguridad mediante claves pre-compartidas, frente al modo SSL/TLS, es su facilidad de configuración, ya que los participantes no necesitan manejar ningún certificado y no hace falta la intervención de una autoridad certificadora (CA). Por el contrario, dichas claves han de estar a buen recaudo, por lo que este modo se considera menos seguro que al utilizar SSL/TLS. Además, dichas claves serán utilizadas durante el tiempo de vida de la VPN creada, por lo que el modo de seguridad mediante claves pre-compartidas es más recomendable utilizarlo para VPN con conexiones de poca duración.

En el modo SSL/TLS, se establece una sesión SSL bidireccional para la autenticación de ambos extremos del túnel, es decir, cada lado de la conexión debe presentar su propio certificado (aunque no siempre el que hace de cliente está obligado a presentarlo). Una vez la autenticación SSL/TLS se ha realizado con éxito, la clave para cifrado/descifrado y la clave para la HMAC es generada aleatoriamente por la función de OpenSSL RAND_bytes y se hace el intercambio mediante la conexión SSL/TLS. En la generación de dichas claves aleatorias se tienen en cuenta parámetros de ambos extremos de la conexión. En este modo nunca se utiliza una clave para ambos sentidos de la comunicación, es decir, cada participante tiene una clave de HMAC para enviar, una clave de HMAC para recibir, una clave para cifrar y una clave para descifrar. Si se utiliza el comando `--key-method` con el parámetro "2", las claves se generan con la función aleatoria dada por la función pseudo aleatoria, PRF, del protocolo TLS. Si el comando `--key-method` tiene el parámetro "1", las claves se generan directamente de la función de OpenSSL RAND_bytes. OpenVPN establece por defecto el comando `--key-method 2`, es decir, si no se incluye en el fichero de configuración este comando con el parámetro que se quiera, OpenVPN utilizará `--key-method 2`.

Durante la regeneración de claves SSL/TLS, existe unos parámetros de transición de ventana que permiten superponer las claves viejas y las claves nuevas en uso, por lo que no existen cuellos de botella durante las renegociaciones SSL/TLS. Una vez cada uno de los participantes tiene su juego de claves, la transmisión de la información puede comenzar en el túnel.

Otra mejora notable en la seguridad que proporciona OpenVPN junto a TLS es que le da al usuario la posibilidad de utilizar una clave pre-compartida, trabajando conjuntamente con la directiva `--tls-auth` (para trabajar en modo TLS), para generar un HMAC que autentifica los paquetes que son parte del protocolo de establecimiento Handshake del propio protocolo TLS. De esta manera se protege a la implementación TLS de OpenSSL de un desbordamiento de buffer por parte de un atacante, ya que dicho atacante no podrá iniciar el Handshake de TLS si no genera paquetes con la HMAC que se está utilizando en ese momento.

OpenVPN multiplexa la sesión SSL/TLS utilizada para la autenticación y para el intercambio de claves con el flujo de datos del túnel cifrado. Además, OpenVPN está

diseñado para poder establecer una conexión SSL/TLS, que se considera fiable, sobre una capa de transporte fiable, como es TCP. Sin embargo, este uso de varias capas fiables puede llevar a cabo un problema de colisiones de capas fiables. Para este caso, OpenVPN ofrece y recomienda la posibilidad de utilizar el protocolo UDP como protocolo encapsulador mediante el comando `--proto udp`. Si utilizamos el comando `--proto-udp`, los paquetes IP no serán tunelizados en un protocolo de transporte fiable, eliminando el problema de las colisiones de capas fiables. Hay que destacar, que si lo que se quiere encapsular es un paquete TCP sobre OpenVPN tomando como opción el protocolo UDP mediante la directiva `--proto-udp`, el protocolo TCP de los datos originales ya proporciona una capa fiable.

En la imagen siguiente se muestra como la sesión TLS y el túnel de datos se multiplexan en el mismo puerto TCP/UDP:

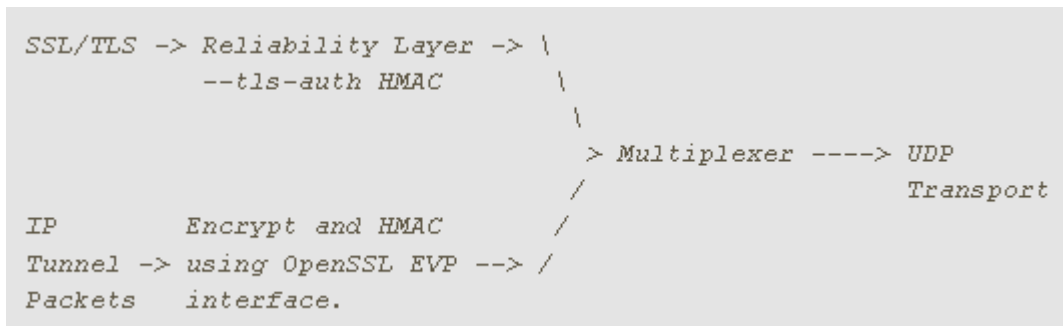


Figura 28. Multiplexación de las Distintas Capas de Seguridad

Este modelo tiene la ventaja de que SSL/TLS ve una capa de transporte fiable, mientras que el paquete IP ve una capa de transporte no fiable, que es exactamente lo que ambos componentes, el protocolo SSL/TLS y el protocolo IP, les conviene ver. Las capas de fiabilidad y autenticación son independientes unas de otras, por ejemplo, el número de secuencia de un paquete está metido dentro de una HMAC y no se utiliza con fines de autenticación.

2.4.2.- Protocolo

A continuación vamos a describir, sin entrar en mucho detalle el **protocolo** utilizado por OpenVPN y el formato de los paquetes de OpenVPN. Dicho protocolo se encuentra definido en el fichero de cabecera **ssl.h**.

OpenVPN encapsula en un mismo datagrama UDP (o TCP, pero no recomendado) los canales de control y datos. Es decir, utiliza el mismo puerto para ambos canales, por lo que un datagrama UDP puede contener un mensaje de control y de datos a la vez. Además, un paquete de OpenVPN puede contener información IP o una trama Ethernet pudiendo operar a nivel IP o a nivel de enlace.

Utilizando UDP como protocolo encapsulador (mediante el comando `--proto-udp`), obtendremos una encapsulación del paquete original como la que se muestra en la siguiente figura:

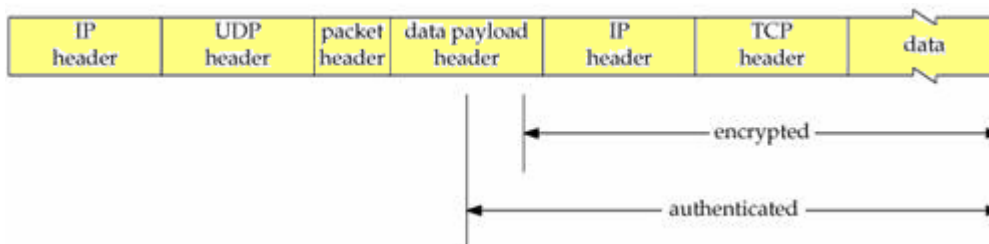


Figura 29. Encapsulación del Canal de Datos en OpenVPN

OpenVPN divide la cabecera del payload en dos partes: la **cabecera del paquete**, que identifica el tipo de paquete y el material o parámetros para las claves, y la **cabecera del paquete de datos**, que contiene parámetros de autenticación, vector de inicialización (IV) y campos de número de secuencia del paquete de datos.

El formato de un paquete en OpenVPN es el siguiente, según se utilice UDP o TCP como protocolo encapsulador (mediante el comando `--proto-udp` o `--proto-tcp`).

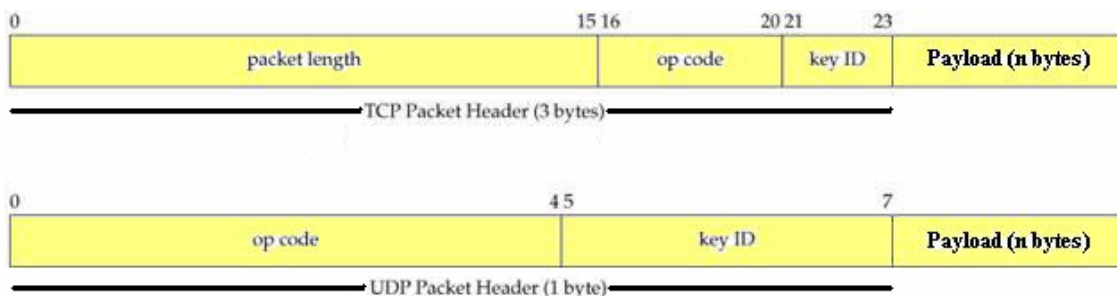


Figura 30. Formato de un Paquete de OpenVPN utilizando TCP o UDP

- **Packet length** (longitud de paquete, 16 bits, sin signo): solo para el caso de usar TCP, siempre se manda como texto plano. Como TCP es un protocolo en flujo, la longitud del paquete define la posición de dicho paquete en el flujo de paquetes.
- **Packet opcode / key_id** (Código de operación / Identificador de clave, 8 bits): solo en caso de utilizar TLS en lugar del modo de seguridad de claves pre-compartidas (o modo de claves estáticas).
 - **Packet message type** (Tipo de mensaje del paquete, 5 bits): define el tipo de mensaje que contiene el paquete. Su nomenclatura es `P_tipoMensaje`.
 - **Key_id** (Identificador de clave, 3 bits): se refiere al identificador de clave de la sesión actual TLS. OpenVPN renegocia sin problemas las sesiones TLS utilizando un nuevo identificador de clave para la nueva sesión. En este caso, se permite el solapamiento entre la sesión TLS vieja y la sesión TLS nueva,

proporcionando una transición estable para que el túnel pueda trabajar sin problemas.

- **Payload** (carga de datos, n bytes): este campo puede ser un mensaje P_CONTROL, P_ACK o P_DATA, lo cuales se describen en la siguiente tabla.

Los tipos de mensaje se muestran a continuación en la siguiente tabla:

Valor	Código de Operación	Significado
1	P_CONTROL_HARD_RESET_CLIENT_V1	Key method 1, clave inicial del cliente, olvida el estado anterior
2	P_CONTROL_HARD_RESET_SERVER_V1	Key method 1, clave inicial del servidor, olvida el estado anterior
3	P_CONTROL_SOFT_RESET_V1	Nueva clave
4	P_CONTROL_V1	Paquete de control (normalmente texto cifrado TLS)
5	P_ACK_V1	Reconocimiento (ACK) tras haber recibido un paquete P_CONTROL
6	P_DATA_V1	Paquete de datos que contiene datos reales cifrados del túnel de datos
7	P_CONTROL_HARD_RESET_CLIENT_V2	Key method 2, clave inicial del cliente, olvida el estado anterior
8	P_CONTROL_HARD_RESET_SERVER_V2	Key method 2, clave inicial del servidor, olvida el estado anterior

Figura 31. Códigos de Operación de los Paquetes de OpenVPN

De la tabla anterior, también hay que destacar que en el mensaje P_CONTROL_SOFT_RESET_V1 se realiza una transición fiable entre la clave vieja y la nueva, en el sentido en que existe una ventana de transición entre la que ambos key_id, el viejo y el nuevo, pueden ser utilizados. Además, OpenVPN utiliza dos formas diferentes de key_id. La primera forma es una key_id de 64 bits, y es la forma de key_id que utilizan los mensajes de control (mensajes P_CONTROL). Por otro lado, los mensajes de datos (mensajes P_DATA) utilizan una key_id más reducida de 3 bits por razones de eficiencia, ya que la mayoría de los paquetes OpenVPN que atravesarán el túnel son paquetes de datos (P_DATA). Además, los 64 bits de la key_id de los paquetes de control se refieren a un identificador de sesión (session_id), mientras que los 3 bits de la key_id de los paquetes de datos se refieren a la key_id en sí.

El mensaje de tipo P_CONTROL contiene un paquete TLS de texto cifrado que ha sido encapsulado en una capa fiable. La fiabilidad de la capa está implementada como un modelo de retransmisión con ACK.

El formato de un **paquete de control** de OpenVPN se muestra en la figura siguiente:

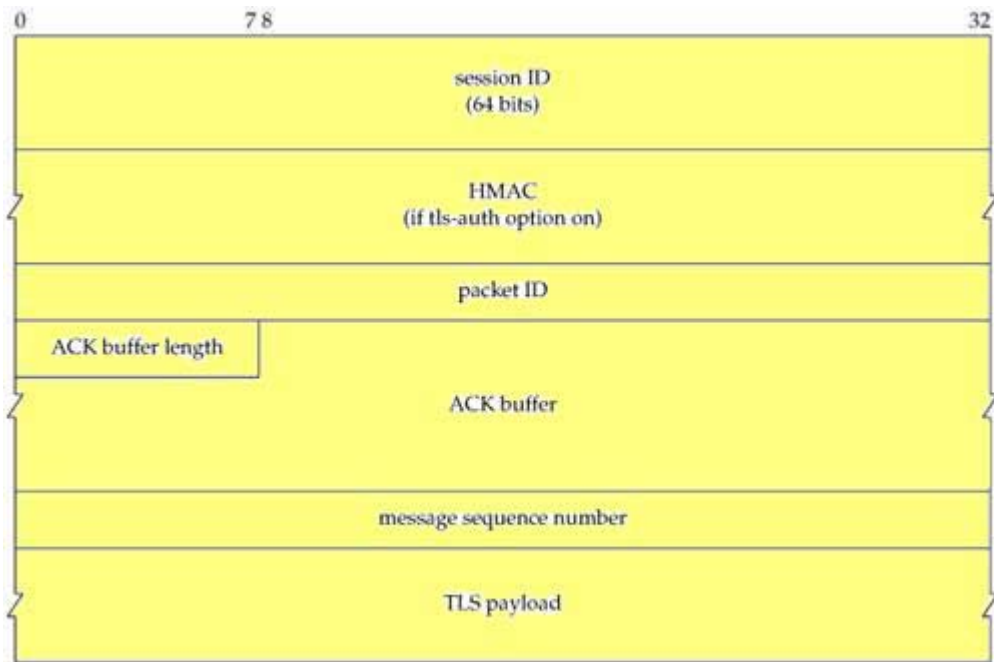


Figura 32. Formato del Paquete de Control de OpenVPN

- **Local session_id** (identificador de sesión): 64 bits aleatorios para identificar la sesión TLS.
- **HMAC** de toda la cabecera de encapsulación para obtener integridad y poder prevenir ataques de denegación de servicio (DoS). Solo se comprueba si se utiliza el comando `--tls-auth`. Normalmente son 16 o 20 bytes. Cuando se habilita esta opción permite a un nodo tirar un paquete si éste fuera falso sin gastar casi ningún recurso en dicho proceso. De esta manera también se previene que un paquete falso pueda alcanzar la capa SSL/TLS, donde podría encontrar puntos débiles o información confidencial.
- **Packet_id** (identificador de paquete): para protección ante ataques de reproducción de paquetes. Este campo realiza la misma función en los paquetes de datos y en los paquetes de control. Cuando se utiliza el modelo de seguridad mediante TLS este campo tiene un tamaño de 32 bits, en otro caso, tiene un tamaño de 64 bits. En este campo se incluyen el número de secuencia y, opcionalmente, una variable `time_t` que marca el instante.
- **P_ACK packet_id length**: es un simple byte que indica el número de reconocimientos (ACK) hay en el buffer de ACK. Cuando este valor es cero, el buffer de reconocimientos ACK que se explicará a continuación no existe. Por el contrario, si no es cero, este campo contiene un número de 32 bits que representa el número, en decimal, de reconocimientos ACK que existen en el buffer.
- **P_ACK packet_id buffer** (si `length > 0`): el buffer de ACK es utilizado por la capa fiable para el reconocimiento de paquetes entre ambos puntos de la comunicación. Si el campo `P_ACK packet_id length` es cero, este campo no existe en el paquete de control de OpenVPN.
- **P_ACK remote session_id** (si `length > 0`): este campo se encuentra al final del buffer de ACK explicado anteriormente y representa el indicador de sesión

(`session_id`) de los pares que componen la VPN. Estos participantes de la VPN utilizan la `session_id` para enlazar los reconocimientos ACK a una sesión VPN particular.

- **Message packet_id** o número de secuencia del paquete: 32 bits.
- **TLS payload ciphertext** o carga TLS cifrada: n bytes, solo para paquetes de control P_CONTROL (no para P_ACK).

Cuando el código de operación (op code) del paquete es P_ACK_V1, el último campo del paquete es el buffer de ACK. Si el código de operación (op code) del paquete es P_CONTROL_V1, los últimos campos del paquete son el número de secuencia del paquete (o `message packet_id`) y la carga TLS cifrada (o TLS payload ciphertext).

Cuando arranca la VPN, ambos extremos del túnel ejecutan la autenticación de cliente de SSL mediante el protocolo Handshake, donde cada uno presenta su certificado al otro extremo. Tras la autenticación, ambos lados saben que se están comunicando con quien dice ser y tienen un canal seguro mediante SSL sobre el cual pueden realizar el intercambio de material aleatorio para generar las claves para el túnel seguro.

Hay dos tipos de mensajes de intercambio de claves, que dependen del método utilizado, `key-method 1` o `key-method 2` (con el comando `--key-method`). Cuando se utiliza el método 1, los tipos de mensaje se caracterizan por tener la nomenclatura `tipoMensaje_V1`. El formato de un mensaje de intercambio de claves utilizando el método 1 se muestra en la figura siguiente:

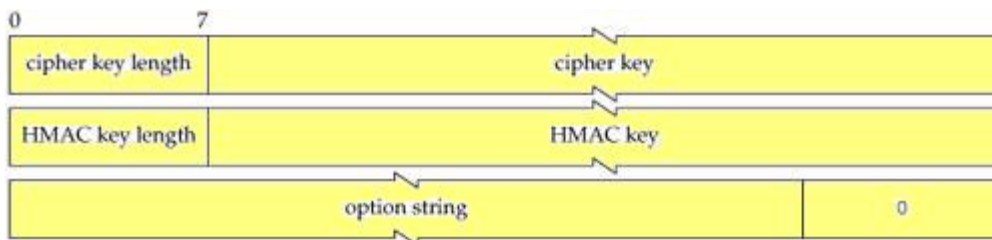


Figura 33. Mensaje de OpenVPN para el Intercambio de Claves (Método 1)

- **Cipher key length** (tamaño de la clave de cifrado): este campo de 1 byte contiene el tamaño de la clave del algoritmo de cifrado.
- **Cipher key** (clave de cifrado): este campo contiene una clave generada aleatoriamente. El extremo que recibe este mensaje deberá de utilizar dicha clave para descifrar.
- **HMAC key length** (tamaño de la clave del HMAC): este campo de 1 byte contiene el tamaño de la clave del HMAC utilizado.
- **HMAC key** (clave del HMAC): este campo contiene una clave generada aleatoriamente. El extremo que recibe este mensaje deberá de utilizar dicha clave para autenticar los paquetes entrantes.

- **Option string** (opciones): este campo contiene opciones que han de coincidir en ambos lados de la VPN con el objetivo de verificar que ambos lados están configurados idénticamente.
- **Campo nulo.**

Por tanto, si nos fijamos en el formato del mensaje anterior hay que destacar que se disponen de 4 claves, 2 por cada participante de la VPN. Este método no es del todo seguro porque las claves que genera cada participante de la VPN lo hacen sin tener en cuenta parámetros del otro participante.

Para solucionar este problema se puede utilizar el método 2 de generación de claves, en el que ambos lados de la comunicación contribuyen en la generación aleatoria de cada clave. Por tanto, la otra opción es utilizar el método 2 o key-method 2 (mediante la opción `--key-method 2`). De esta manera los mensajes tienen una nomenclatura similar a `tipoMensaje_V2`. Este método es más robusto que el método 1 de generación de claves. El formato de un mensaje de intercambio de claves utilizando el método 2 se muestra en la figura 34.

El método 2 de generación de claves, o key-method 2 (del comando `--key-method 2`), combina los bits obtenidos mediante la función `RAND_bytes` de OpenSSL de ambos extremos de la conexión y los utiliza en una función pseudo aleatoria (PRF) propia del protocolo TLS. El método 2 es el más recomendado y el que utiliza OpenVPN a partir de la versión 2. Como se puede ver en el formato del mensaje para intercambio de claves utilizando el método 2, éste tiene los siguientes campos:

- **4 bytes a cero.**
- **Key-method** (método de intercambio de claves): indica, mediante 1 byte, el método de intercambio de claves usado. En este caso el valor de este campo es 2, ya que el resto de valores posibles están para posibles métodos futuros.
- **Premaster secret** (clave previa): son 48 bits aleatorios generados únicamente por la maquina que hace de cliente. Este campo no está presente en el mensaje de intercambio de claves del servidor. Este campo tiene la misma funcionalidad tanto en TLS como en SSL.
- **Random 1 y Random 2** (aleatorios 1 y 2): son utilizadas para el proceso de generación de la clave. Ambos campos tienen un tamaño de 32 bits. Puesto que ambos lados de la VPN contribuyen con dos campos con números aleatorios, ninguno de los lados puede determinar ninguna de las claves.
- **Option string length** (longitud del campo opciones): este campo son 2 bytes que contienen la longitud del “string” de opciones, terminando en null. Como ocurre en el método 1 de intercambio de claves, este campo debe coincidir en ambos lados de la VPN para verificar que están configurados idénticamente.

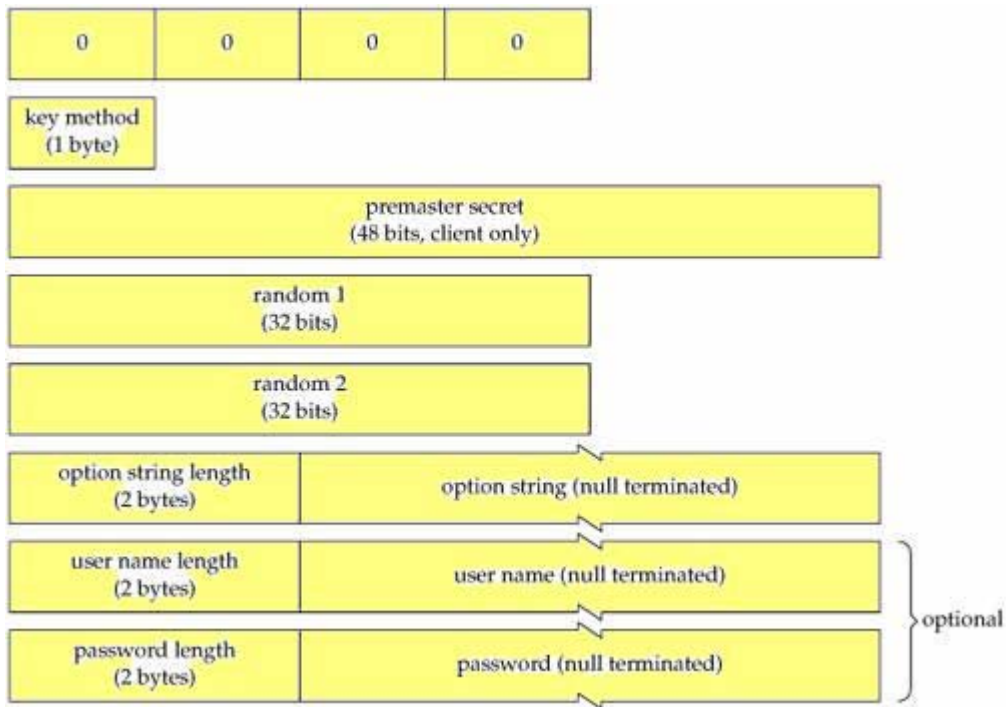


Figura 34. Mensaje de OpenVPN para el Intercambio de Claves (Método 2)

- **Option string** (“string” de opciones): este campo de n bytes contiene opciones que han de coincidir en ambos lados de la VPN con el objetivo de verificar que ambos lados están configurados idénticamente.
- **User name length** (longitud del nombre de usuario): este campo son 2 bytes que contiene la longitud del campo nombre de usuario, terminado en null.
- **User name**: campo de n bytes que contiene el nombre de usuario.
- **Password length** (longitud del password): este campo de 2 bytes contiene la longitud del campo password, terminado en null.
- **Password**: campo de n bytes que contiene el password del usuario.

Hay que destacar, de este mensaje, que los campos user name (nombre de usuario) y password (password de usuario) solo son utilizados cuando OpenVPN corre bajo Proxy HTTP (por ejemplo, un servidor Proxy squid), y éste Proxy necesita autenticación mediante nombre de usuario y password. Por tanto, estos campos son opcionales y no están presentes a no ser que se utilice un servidor Proxy.

El proceso de generación de la clave secreta es idénticamente igual que el método utilizado en el protocolo TLS. En primer lugar, ambos extremos de la comunicación generan la clave maestra (master secret) concatenando el string OpenVPN master secret, el valor del campo Random 1 del cliente y el valor del campo Random 1 del servidor, y este resultado se utiliza, junto a la clave previa (premaster secret), en una operación HMAC. La operación quedaría de esta manera:

```
PRF (premaster secret, OpenVPN master secret || client Random 1 || server Random 1)
```

Figura 35. Operación para el Calculo de la Master Secret en OpenVPN

La función PRF utiliza los algoritmos de hashing MD5 y SHA-1 y realiza unas operaciones OR con los resultados. El resto de operaciones son idénticas a las explicadas en el apartado 1.2.6 para el cálculo de la clave maestra en TLS.

Una vez que los dos participantes tienen la clave maestra, cada uno genera 4 claves: una para cifrar, otra para descifrar, una para la autenticación del tráfico de entrada mediante un HMAC y otra para la autenticación del tráfico de salida mediante un HMAC. La manera de generar estas claves es concatenando el string OpenVPN key expansion, el Random 2 del cliente, el Random 2 del servidor, el session ID del cliente y el session ID del servidor. El resultado de esta operación junto a la clave maestra es utilizado como parámetros de entrada en una función PRF para obtener las claves:

```
PRF (master secret, OpenVPN key expansion || client Random 2 || server Random 2 || client SID || server SID)
```

Figura 36. Operación para el Cálculo de las 4 Claves en OpenVPN

Cuando llega el momento de una renovación de claves, se produce un nuevo proceso de intercambio de claves igual al utilizado anteriormente y con el mismo método que el utilizado anteriormente, método 1 o 2. Para ayudar en la transición entre las claves viejas y las claves renovadas, OpenVPN mantiene un conjunto de tres grupos de claves:

- Las claves activas.
- Las claves que van a ser “jubiladas” o retiradas.
- Las claves que han sido descartadas como retiradas, debido a que la negociación para renovar las claves ha fallado.

Estos tres conjuntos de claves definen el campo `key_id` de la cabecera del paquete OpenVPN, ya sea para TCP o para UDP. El identificador de clave o `key_id`, indica cual de los tres conjuntos de claves es utilizado en el paquete.

Tras definir el paquete de control de OpenVPN y mostrar sus características principales y como se obtienen las claves tras el intercambio de claves, vamos a definir el otro tipo de paquetes que utiliza OpenVPN: el **paquete de datos** de OpenVPN. Hay que destacar que, en un paquete encapsulado por OpenVPN, la parte cifrada y la parte autenticada se extienden a lo largo de la cabecera de la carga de datos o data payload header como bien se puede ver en la figura 29.

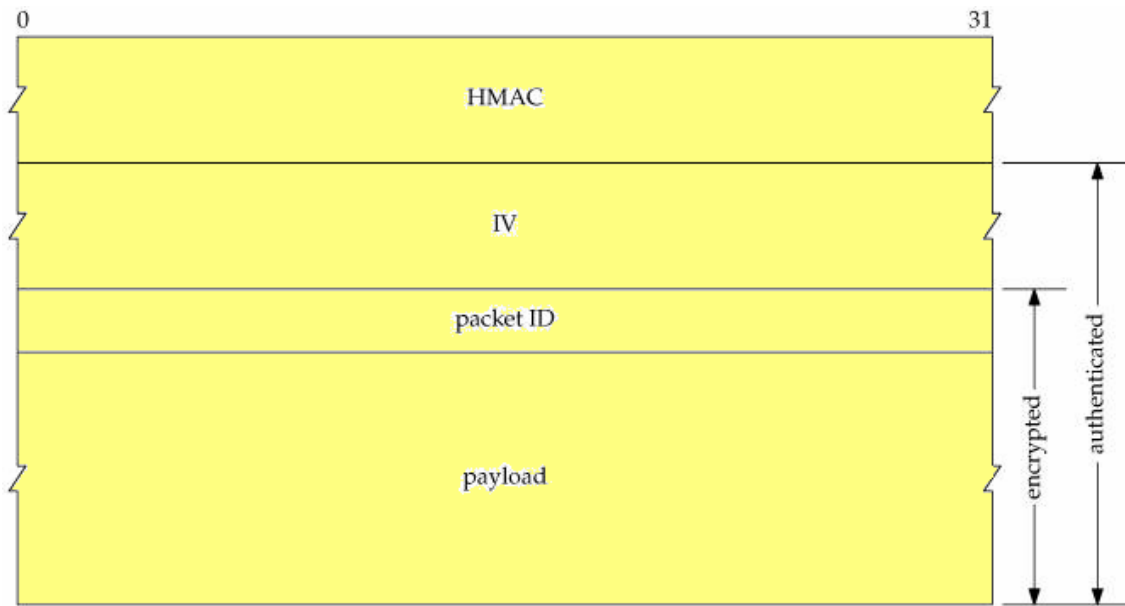


Figura 37. Formato del Paquete de Datos OpenVPN

- El campo **HMAC** consiste en un campo para autenticar mensajes utilizando como técnicas de hashing los algoritmos SHA-1 o MD5. Como se puede ver en la figura 36 este campo autentica los campos payload, packet ID, y el vector de inicialización (IV).
- El **vector de inicialización (IV)** es un vector generado de manera aleatoria para el cifrado en modo CBC (Cipher Block Chaining). Normalmente, suele tener un tamaño de 64 o 128 bits. Además, OpenVPN soporta los modos CFB (Cipher FeedBack) y OFB (Output FeedBack), los cuales requieren el vector IV.
- El campo **packet ID** (o identificador de paquete) es utilizado como un número de secuencia para prevenir ataques de reproducción de paquetes en la VPN.
- **Payload** o campo de datos de usuario, de n bits, que contiene el paquete IP encapsulado o la trama Ethernet encapsulada.

Como se puede ver en la figura la parte autenticada se compone de los campos IV, packet ID y el payload da datos mientras que la parte cifrada del mensaje consta de los campos packet ID y payload de datos.

OpenVPN mantiene una ventana deslizante con dichos números de secuencia o packet ID y rechaza los paquetes si su número de secuencia está a la izquierda de la ventana o si está dentro de la ventana pero ya se recibió un paquete con el mismo número de secuencia previamente. Cuando OpenVPN recibe un paquete con un número de secuencia a la derecha de la ventana, la ventana se desplaza hacia la derecha, de manera que en el extremo derecho de la ventana se encuentra el nuevo número de secuencia o packet ID. Además, OpenVPN aplica un test de tiempos para rechazar los paquetes recibidos fuera de orden que han sido recibidos después de t segundos de recibir un paquete que contiene un número de secuencia mayor. Este parámetro t es configurable y tiene un valor por defecto de 15 segundos en OpenVPN. El tamaño de la ventana también es configurable y tiene un tamaño por defecto de 64 números de secuencia o

packet ID. También hay que destacar, que el tamaño del campo packet ID es de 32 bits cuando el modo de cifrado es CBC. Cuando el modo de cifrado es CFB u OFB, o cuando se utiliza el modo de seguridad por clave pre-compartida (static key mode), el tamaño de este campo es de 64 bits. En el modo CFB/OFB los 64 bits provienen de la concatenación de un instante de tiempo (timestamp) y el valor de un contador ascendente, y dicho resultado está contenido en el vector de inicialización (IV) como un método de optimización para ahorrar espacio.

Los paquetes de datos (P_DATA) y los paquetes de control (P_CONTROL o P_ACK) utilizan diferentes números de secuencia del campo packet ID ya que el paquete de datos se utilizan sobre un canal inseguro mientras que los paquetes de control se utilizan sobre un canal seguro. Además, cada uno de estos paquetes utiliza claves diferentes para su HMAC.

Destacar que cuando se utiliza la opción --tls-auth, todos los tipos de mensajes son protegidos por una autenticación por HMAC proporcionada gracias a esta opción, incluso los primeros paquetes del protocolo Handshake de TLS. Como ya se ha comentado en esta memoria la utilización de la opción --tls-auth permite a OpenVPN rechazar los paquetes entrantes falsos de manera sencilla y rápida sin malgastar recursos durante el establecimiento de una conexión TLS que, en el último momento, falla, por ejemplo.

También hay que destacar que el vector de inicialización (IV) es generado de manera aleatoria por un generador de números pseudoaleatorios que son inicializados con una semilla generada por la función RAND_bytes de OpenSSL. Las funciones HMAC, el algoritmo de cifrado y el algoritmo de descifrado son proporcionadas por la interfaz EVP de OpenSSL y permite al usuario poder escoger un método de cifrado, los tamaños de clave y las funciones Hash utilizadas para el HMAC. Por defecto, BlowFish con un tamaño de clave de 128 bits (este tamaño no es fijo) es el algoritmo de cifrado escogido por OpenVPN y el algoritmo SHA1 con un tamaño de clave de 160 bits es la función Hash para los HMACs.

Como se ha podido comprobar, el canal de datos de OpenVPN reúne precisamente los requerimientos de una VPN: el cifrado y la autenticación del túnel entre dos redes o máquinas. El canal de datos de OpenVPN esta bien diseñado y no tiene, aparentemente, ninguna debilidad en la seguridad que pueda ser explotada. En resumen, el modelo utilizado por OpenVPN es muy parecido al modelo utilizado en el protocolo ESP de IPsec y tiene la ventaja de haber sido desarrollado conociendo las lecciones aprendidas y posibles desventajas del modelo de IPsec.

2.4.3.- Asignación de Direcciones

En este apartado se comenzará por aclarar el **direccionamiento IP** utilizado típicamente para las redes privadas, como puede ser cualquier VPN particular implementada con

OpenVPN. Para ello hay que decir que la IANA ha reservado los siguientes espacios de direcciones IP para las redes privadas (definida en el RFC 1918):

Grupo A	10.0.0.0	10.255.255.255	10/8
Grupo B	172.16.0.0	172.31.255.255	172.16/12
Grupo C	192.168.0.0	192.168.255.255	192.168/16

Figura 38. Espacio de Direcciones IP para las Redes Privadas según la IANA

Estos bloques de direcciones IP son normalmente utilizados en las configuraciones de las VPN, pero es importante seleccionar las direcciones que minimizan la probabilidad de que haya un conflicto con las direcciones IP de red o de subred. Los tipos de conflictos que se han de prevenir son:

- Conflictos provocados por tener diferentes sitios de la VPN con LANs que utilizan la misma numeración de red.
- Conexiones de acceso remoto desde sitios que están utilizando redes privadas que provocan conflictos con las redes o subredes de la propia VPN.

Por ejemplo, suponemos que utilizamos una dirección de subred 192.168.0.0/24 para la VPN. Entonces, intentamos conectar a la VPN desde una conexión a Internet de una cafetería, la cual utiliza el mismo direccionamiento de subred para su LAN por Wifi. En este caso se tendría un conflicto debido a que nuestra maquina no sabría si la IP 192.168.0.1 se refiere a la puerta de enlace de la LAN Wifi de la cafetería o a la dirección utilizada por la VPN.

Otro ejemplo, sería suponer que queremos conectarnos a un sitio a través de la VPN, y dicho sitio utiliza el direccionamiento 192.168.0.0/24 para su LAN. En este caso esta configuración no podría trabajar si no añadimos un sistema de traducción de direcciones NAT, porque la VPN no conocería como encaminar los paquetes entre múltiples sitios si esos sitios no utilizan un direccionamiento que únicamente los identifiquen a ellos mismos.

Por tanto, la mejor solución es utilizar un direccionamiento de red 10.0.0.0/24 o 192.168.0.0/24 para nuestra LAN privada. Otra solución es utilizar alguna dirección que tenga una probabilidad baja de ser utilizada por la LAN de algún aeropuerto, un hotel o una WLAN de alguna cafetería, donde puedas conectarte a cualquier lugar de manera remota. La mejor opción utilizada es la de usar direcciones IP que estén por la mitad del bloque de direcciones 10.0.0.0/8 (por ejemplo la 10.66.77.0/24).

Otro aspecto importante de OpenVPN es el direccionamiento IP que utiliza para la implementación de la VPN para la implementación de túneles, ya que el modo Tunnel (mediante túneles) ha sido el modo empleado en este proyecto fin de carrera para la implementación de las VPN en cada escenario. Para dicha implementación, OpenVPN asigna una subred con una mascara de red /30 a la VPN con el fin de proporcionar

compatibilidad con los clientes que utilizan plataformas tipo Windows (2000/XP/Vista) debido a limitaciones dadas por los controladores TUN/TAP de Windows (más conocidos como TAP-Win32 drivers) para la emulación de túneles.

Si se sabe previamente que todos los clientes que se conectarán al servidor OpenVPN no utilizan un sistema operativo Windows se puede evitar esta característica utilizando la directiva `--ifconfig-pool-linear` (por tanto, esta directiva es incompatible si alguna maquina utiliza Windows).

En la versión 1.6 de OpenVPN, cuando se ejecuta una instancia de OpenVPN por cliente, esto se traduce a lo que nosotros esperábamos: un enlace punto a punto por cada cliente y el servidor OpenVPN, es decir, una interfaz “tun” por cada enlace punto a punto cliente-servidor. Por tanto, el servidor implementará tantos interfaces “tun” como clientes estén conectados.

Sin embargo, en la versión 2.0 de OpenVPN podemos soportar múltiples clientes conectados al servidor OpenVPN mediante un solo interfaz “tun” y un solo puerto (el 1194) en el servidor y su respectivo interfaz “tun” en cada cliente. De esta manera, el enlace punto a punto, desde el punto de vista del servidor, se puede ver como un enlace entre el sistema operativo y OpenVPN. Una vez dentro de OpenVPN (del servidor), se crean los enlaces punto a punto por cada cliente que se conecta a la VPN. Si todos los sistemas operativos utilizados (tanto clientes como servidor) pueden implementar enlaces punto a punto “reales” sobre la interfaz “tun”, entonces la VPN se podría implementar dándole una dirección IP al servidor y una dirección IP diferente por cada cliente, olvidando el problema tener que contar con una subred con mascara de red /30.

Pero si alguno de las máquinas que componen la VPN utiliza un sistema operativo tipo Windows, esta máquina no soporta enlaces punto a punto “reales”, por lo que hay que emularlas de alguna manera utilizando, para ello, una subred con mascara de red /30 dentro del propio direccionamiento IP de la VPN.

Por tanto, si no tenemos el problema de utilizar máquinas con sistema operativo Windows un ejemplo de enlace simple podría ser un enlace punto a punto con las direcciones IP 192.168.1.1 <-> 192.168.1.2, donde el servidor tendría la IP 192.168.1.1 y el cliente tendría la dirección IP 192.168.1.2. En el caso de conectarse otro cliente se le daría la dirección 192.168.1.3, por lo que se tendría un enlace con las direcciones IP 192.168.1.1 <-> 192.168.1.3.

Sin embargo, utilizando alguna máquina con un sistema operativo Windows, aparecería el problema de tener que utilizar un direccionamiento de subred con mascara de red /30 por cada cliente que se conecte al servidor OpenVPN. Por ejemplo, la primera subred con mascara /30 disponible tiene la dirección 192.168.1.0/30 donde:

- 192.168.1.0 es la dirección de subred y dirección de red de la VPN.
- 192.168.1.1 es la dirección IP asignada al servidor OpenVPN.

- 192.168.1.2 es la dirección IP asignada al otro extremo del enlace punto a punto del servidor, pero no es asignada a ningún cliente.
- 192.168.1.3 es la dirección de broadcast de esta subred.

Y la siguiente subred con máscara /30 disponible para el primer cliente que se conecte al servidor OpenVPN tendrá la dirección de subred 192.168.1.4/30 donde:

- 192.168.1.4 es la dirección de subred.
- 192.168.1.5 es la dirección IP virtual del servidor OpenVPN.
- 192.168.1.6 es la dirección IP asignada al primer cliente que se conecta a la VPN.
- 192.168.1.7 es la dirección de broadcast de esta subred.

Por tanto, para que el primer cliente conectado al servidor OpenVPN alcance cualquier punto de la VPN, primero tendrá que pasar por el servidor OpenVPN, por lo que se establecerá una ruta en dicho cliente, que encaminará el tráfico a la dirección IP virtual del servidor 192.168.1.5.

Como 192.168.1.5 es solo una dirección IP virtual dentro del servidor OpenVPN, utilizada como un extremo final para las rutas del cliente con la IP 192.168.1.6, OpenVPN no tiene ningún problema para responder los pings enviados a la dirección 192.168.1.5. Pero, en este ejemplo, hay que destacar que la dirección IP real del servidor no es la 192.168.1.5 sino que es la 192.168.1.1, ya que es la dirección IP que se reconoce en el sistema operativo del servidor, pero no hay ningún problema para responder pings que proceden de la IP 192.168.1.6 (IP del cliente).

Esta característica en el direccionamiento de OpenVPN hace que se desperdicien una pequeña cantidad de direcciones IP de la red de la VPN, pero es la mejor manera de conseguir una configuración consistente y estable que pueda trabajar con todos los sistemas operativos soportados por OpenVPN, en especial, los sistemas operativos tipo Windows (2000/XP/Vista).

2.4.4.- UDP y TCP

OpenVPN permite, desde la versión 1.5, la posibilidad de utilizar los protocolos TCP o UDP como protocolos de transporte para establecer la comunicación con el host remoto. Para poder manejar esta característica OpenVPN proporciona la directiva `--proto p` donde `p` puede tener los valores `udp`, `tcp-client` o `tcp-server`. El protocolo por defecto es `udp`, por tanto, OpenVPN utilizará el protocolo `udp` cuando la directiva `--proto` no sea utilizada. También se puede especificar que se quiere utilizar el protocolo `udp` especificando, para ello, la línea `--proto udp`. Para operar con el protocolo TCP, uno de los extremos (el servidor) debe incluir la línea `--proto tcp-server` y en el otro extremo (los clientes) se debe incluir la línea `--proto tcp-client`. El extremo que incluye la directiva explicada con el valor `tcp-server` esperará indefinidamente la llegada de una

petición de conexión. El extremo que incluye la directiva con el valor `tcp-client` hará el intento de conectarse al otro extremo, esperará, y si dicho intento falla, se quedará “dormido” durante 5 segundos (ajustable mediante la directiva `--connect-retry`, que tiene el valor por defecto de 5 segundos) e intentará conectarse de nuevo.

En Internet, existen parámetros que varían durante el tiempo, dependiendo del tipo de conexión, como pueden ser el ancho de banda, el retardo o la tasa de pérdidas. Otro ejemplo de parámetros variables de Internet son los temporizadores adaptativos utilizados por el protocolo TCP. Cuando el temporizador de un segmento TCP expira, el siguiente temporizador es incrementado de manera exponencial para prevenir problemas de congestión.

La política de fiabilidad del protocolo TCP mediante temporizadores funciona bien en Internet en la mayoría de las conexiones. Como TCP intenta por todos los medios no romper la conexión, el temporizador puede ser incrementado en un rango de incluso minutos. Este modelo de fiabilidad puede no funcionar bien cuando se implementa otra capa fiable además de TCP, como por ejemplo, cuando una conexión TCP está por encima de otra conexión TCP en la pila de protocolos. Este caso no es normal en Internet, pero si puede ser normal en los sistemas VPN, ya que el segmento TCP original puede ser encapsulado dentro de otro segmento TCP por la aplicación utilizada por la VPN, lo que no es una buena técnica. Por ejemplo, este problema se da en el caso de correr el protocolo PPP (Point to Point Protocol) sobre SSH u otro protocolo basado en TCP.

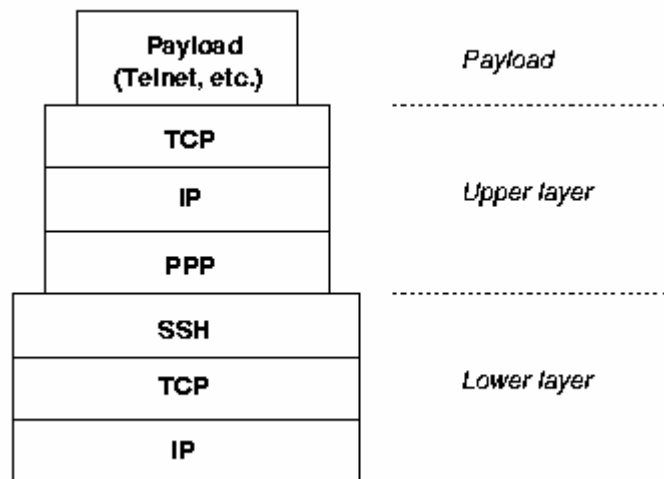


Figura 39. Ejemplo de Uso de TCP sobre TCP

En este ejemplo la capa TCP superior y la capa TCP inferior tienen diferentes temporizadores. Cuando la conexión de la capa superior empieza a funcionar rápidamente, sus temporizadores también lo hacen de la misma manera. Por el contrario, también puede suceder que la conexión funcione lentamente y tenga unos temporizadores lentos.

En esta situación, si la conexión TCP inferior comienza a perder paquetes, pone en cola una retransmisión e incrementa los temporizadores. Como la conexión está bloqueada durante cierto tiempo, la capa TCP superior no recibirá el ACK pertinente, y también pondrá en cola una retransmisión. Como el temporizador de la capa TCP superior es todavía menor que el temporizador de la capa TCP inferior, la capa TCP superior encolará retransmisiones de manera más rápida que la capa inferior TCP puede procesarlos. De esta manera la congestión incrementará cuanto más rápido ponga en cola las retransmisiones la capa TCP superior.

Resumiendo, la fiabilidad que proporcionan las capas TCP es contra productiva en este caso. La capa TCP superior es completamente innecesaria, porque la capa inferior garantiza la entrega de segmentos TCP pero la capa superior TCP no sabe nada de esto, por lo que dicha capa siempre asume que tiene protocolos no fiables por debajo de ella y hace su función de protocolo fiable (aunque no deba hacerlo, en este caso).

Otro problema es que TCP optimiza el tamaño del paquete de manera que no sea necesario fragmentar el paquete durante su tránsito por Internet. El problema surge cuando se encapsula TCP sobre TCP, obteniendo un paquete más largo ya que ahora tenemos un paquete TCP, con un tamaño máximo de paquete, encapsulado dentro de otro paquete TCP, por lo que, intrínsecamente, será de mayor tamaño. Por tanto, esto provocará problemas de fragmentación que pueden ser graves en el sentido en que todos los paquetes chocarán con el primer router que no sea capaz de encaminar paquetes demasiado grandes. Este problema, en Internet, se resume en que el host destino nunca podrá reensamblar el mensaje completo. Por tanto, de repente nos encontramos con el doble de paquetes, el doble de posibilidades de perder y retransmitir un paquete y el doble de paquetes que llegan y deben ser reensamblados.

OpenVPN ha sido diseñado originalmente para operar de manera óptima utilizando como protocolo de transporte UDP, pero TCP puede utilizarse en situaciones donde UDP no puede ser usado. En comparación con UDP, como se ha explicado anteriormente, TCP es menos eficiente y menos robusto cuando es utilizado sobre redes que utilizan alguna capa fiable y con posibles congestiones. Existen algunos casos, sin embargo, donde utilizar TCP puede tener ventajas de seguridad y robustez frente a la utilización de UDP, como en el caso de utilizar túneles no IP o de utilizar aplicaciones sobre protocolos que no tienen ningún nivel de fiabilidad. Otro aspecto a favor de la utilización de UDP frente a TCP es que la utilización de UDP frente a TCP proporciona mejor protección frente a ataques de denegación de servicio (DoS) y frente al escaneo de puertos.

En muchas aplicaciones que se pueden utilizar sobre OpenVPN, como pueden ser los sistemas de VoIP, también conviene utilizar el protocolo UDP frente al protocolo TCP por diversas razones. Para empezar, en la mayoría de los casos, los sistemas VoIP utilizan el protocolo RTP (Real-time Transport Protocol) que es transportado utilizando el protocolo UDP. Como ya se ha comentado TCP proporciona una conexión fiable, pero este aspecto, en los sistemas VoIP no es bueno. Si utilizamos TCP en VoIP, cuando un paquete es rechazado, el emisor vuelve a retransmitir el paquete ante la petición del receptor o receptores. Pero en este caso al reensamblar los paquetes de voz

retransmitidos en un stream de audio podrían resultar demasiado tarde para obtener una reproducción fiable del sonido original. Por el contrario, UDP da por bueno paquetes enviados fuera de orden y no existe verificación (ACK) de si el paquete llega.

Capítulo 3: Comenzando con OpenVPN

3.1.- Instalación del paquete OpenVPN

En este apartado se comentará de manera breve el proceso de **instalación de OpenVPN** en sistemas operativos Windows y Linux, ya que han sido los sistemas operativos utilizados a lo largo de este proyecto fin de carrera. La instalación de OpenVPN es **fácil e independiente de la plataforma utilizada**. En este apartado, además, se probarán diferentes métodos de instalación, se compilará el código fuente proporcionado por el proyecto OpenVPN y se habilitarán algunas características de red en el kernel de Linux para los dispositivos TUN/TAP, de manera que se pueda utilizar OpenVPN utilizando estos drivers. Además se tendrán en cuenta y se comentarán los paquetes necesarios para completar la instalación de OpenVPN con éxito.

Hay que destacar que la versión de OpenVPN que se va a instalar es la **versión 2.0.9**, ya que es la última versión más estable que se puede descargar de la Web. Aunque existe una versión superior de OpenVPN, la versión 2.1, esta versión está todavía en desarrollo, por lo que la pagina Web de OpenVPN solo proporciona versiones beta de esta versión. La descarga se ha realizado desde la Web: <http://openvpn.net>.

3.1.1.- Prerrequisitos en la Instalación de OpenVPN

Algunos prerrequisitos son imprescindibles si se quiere instalar OpenVPN en una máquina. Las versiones de Windows que soportan OpenVPN son las versiones XP, 2000 y Vista, sin embargo, OpenVPN puede ser instalado en casi todas las distribuciones de Linux existentes. Además, OpenVPN necesitan las siguientes características en el sistema donde va a ser instalado:

- El sistema debe proporcionar soporte para los drivers TUN/TAP: en casi todas las distribuciones de Linux, a partir de la versión 2.4 del kernel, proporcionan soporte para los drivers TUN/TAP. Solo en caso de utilizar una distribución de Linux con una versión de kernel menor que la versión 2.4 o si hemos creado nuestro propio kernel, tendremos que instalar algún paquete para que dé soporte a los drivers TUN/TAP. Una solución a este problema puede ser, por ejemplo, instalar la aplicación VTUN comentada en el apartado 2.2.2. Para ello se puede descargar desde la Web: <http://vtun.sourceforge.net/tun>.
- Las librerías OpenSSL han de estar instalados en el sistema (opcional): en casi todas las distribuciones de Linux es complemento necesario la instalación del paquete OpenSSL. En muchas distribuciones dicho paquete viene ya instalado. Sin embargo, si se desea compilar OpenVPN desde el código fuente, se tiene que descargar e instalar OpenSSL, previamente, desde: <http://www.openssl.org>.
- La librería de compresión LZO ha de estar instalada (opcional): de nuevo, la mayoría de las distribuciones de Linux actuales proporciona esta aplicación ya instalada por defecto. Sin embargo si se tiene una versión antigua que no

proporcione la librería de compresión LZO, se puede descargar sin ningún problema desde su Web: <http://www.oberhumer.com/opensource/lzo>.

En las instalaciones de Windows (archivos .exe ejecutable) se instalan a la vez todos los paquetes que hacen falta para poder utilizar OpenVPN, es decir, no es necesario descargar ningún archivo de instalación excepto el que proporciona OpenVPN para los sistemas operativos Windows. También hay que destacar que las distribuciones de Linux incluyen un gestor de paquetes que pueden proporcionar los paquetes anteriormente descritos y facilitar la instalación de estos.

En Windows, podemos encontrar el paquete OpenVPN junto con una interfaz gráfica de usuario (GUI) y algunas aplicaciones y funcionalidades extra en la misma instalación si se descarga desde la Web: <http://openvpn.se>.

En algunas distribuciones de Linux, como SuSE, incorporan herramientas de instalación como YaST que contienen los paquetes descritos hasta ahora, incluyendo OpenVPN. SuSE, por ejemplo, maneja paquetes RPM que la Web de OpenVPN también proporciona para su descarga. Además facilita la descarga de dichos paquetes a través de la descarga desde los repositorios oficiales del proyecto SuSE. Las distribuciones de Linux tipo Debian, como Ubuntu, utilizan un manejador de paquetes muy sofisticado que también permite la instalación de este software mediante los repositorios de los servidores proporcionados. Por otra parte, como sucede en la mayoría de los proyectos de código abierto, también se puede instalar OpenVPN mediante el código fuente que OpenVPN proporciona para poder descargarlo desde la Web.

3.1.2.- Instalación de OpenVPN Windows

En los sistemas Windows soportados por OpenVPN (XP, 2000 y Vista) la instalación es muy sencilla, ya que todas las aplicaciones y paquetes necesarios (OpenVPN, librerías de OpenSSL, librería de compresión LZO y los drivers TAP-win32) se instalan a partir de un mismo archivo ejecutable mediante unos cuantos pasos. El ejecutable instalado en dichas máquinas se ha descargado de la Web <http://openvpn.se>, ya que este ejecutable, además de las aplicaciones ya mencionadas anteriormente, incluye una GUI para OpenVPN y una aplicación para la creación y manejo de certificados llamada “My Certificate Wizard”. La instalación y utilización de OpenVPN en plataformas Windows destaca, sobre todo, por su facilidad, tanto en instalación como en uso.

En Windows, OpenVPN puede ser ejecutado mediante el comando “OpenVPN” junto con sus opciones a través de la consola de comandos, o bien puede ser arrancado cargando un archivo de configuración con la extensión .ovpn (extensión .conf en el resto de sistemas operativos). Además, OpenVPN puede ser ejecutado como servicio en máquinas Windows, lo que significa que OpenVPN puede ser ejecutado automáticamente al arrancar el sistema operativo.

La instalación de la versión 2.0.9 junto a la GUI de OpenVPN y algunas aplicaciones y funcionalidades extra en Windows XP se puede realizar fácilmente los pasos que aparecen en las siguientes imágenes:

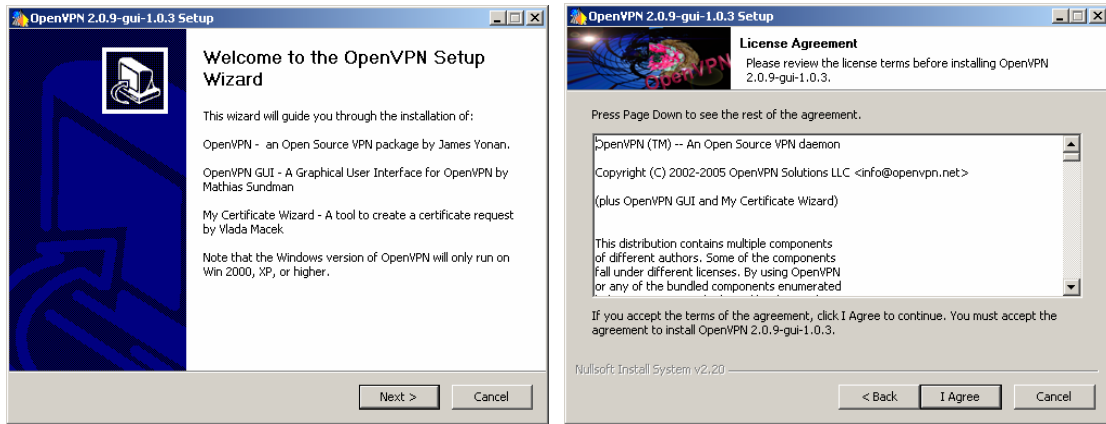


Figura 40. Pasos 1 y 2 en la Instalación en Windows XP de OpenVPN

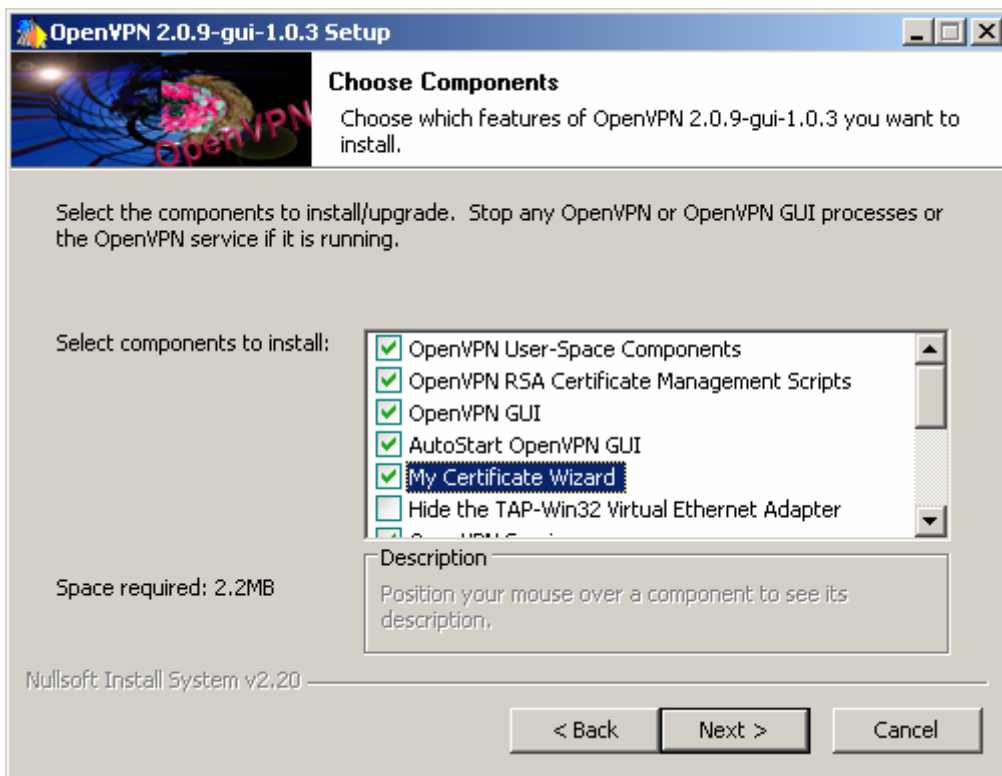


Figura 41. Paso 3: Componentes Instalados en Windows XP

Hay que destacar que OpenVPN debe ser instalado y ejecutado mediante un usuario que tenga privilegios de administrador. Esta restricción se puede eludir en el caso de ejecutar OpenVPN, en segundo plano, como un servicio ya que, en tal caso, también podrían utilizar OpenVPN usuarios que no fueran administradores, una vez instalado.

Los paquetes y aplicaciones que se pueden instalar mediante el ejecutable de Windows se pueden ver en la siguiente figura:

Option	Feature	Client Install	Server Install
OpenVPN User-Space Components	The OpenVPN program	x	x
OpenVPN RSA Certificate Management Scripts	easy-rsa for Windows		x
OpenVPN GUI	The graphical user interface	x	
AutoStart OpenVPN GUI	Link for auto start	x	
My Certificate Wizard	Certificate requests for a certificate authority	x	
Hide the TAP-Win32 VEA	Interface is not shown in network setup		
OpenVPN Service	Configure OpenVPN as a service		x
OpenVPN File Associations	Configuration files (*.ovpn) are associated with OpenVPN	x	x
OpenSSL DLLs	Dynamic link libraries	x	x
TAP-WIN32 VEA	Virtual network interface	x	x
Add OpenVPN to PATH	Openvpn.exe is in the path of every user's command line	x	x
Add Shortcuts to Start Menu	Shortcut to start menu	x	x

Figura 42. Aplicaciones Instaladas en Windows XP

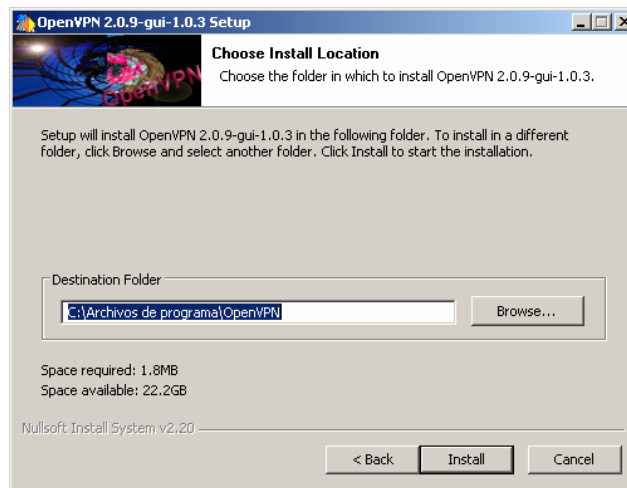


Figura 43. Paso 4: Directorio de Instalación de OpenVPN en Windows XP

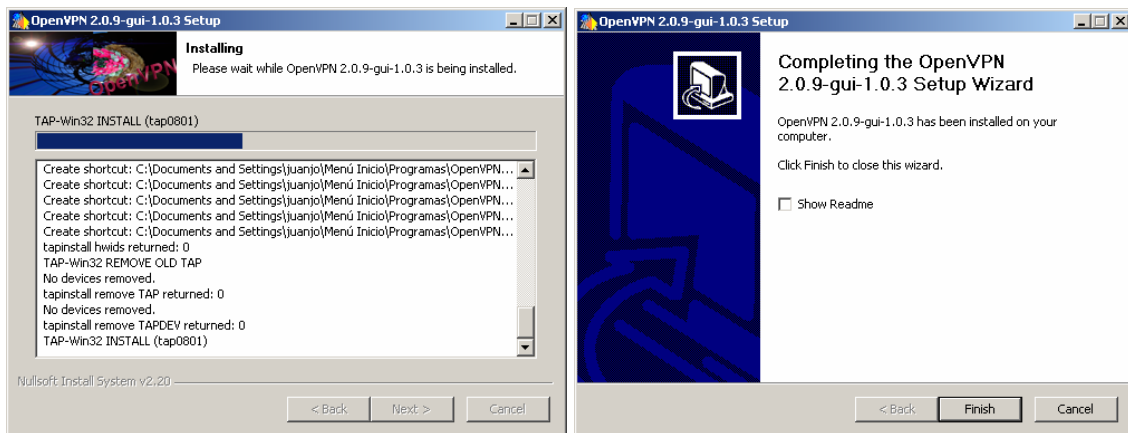


Figura 44. Pasos 5 y 6: Progreso y Finalización de la Instalación de OpenVPN

Una vez terminado el proceso de instalación, podemos comprobar como tenemos un icono en la barra de tareas que representa la interfaz TUN/TAP que habilitada para su funcionamiento y otro icono en la barra de herramientas que pertenece al proceso de la GUI de OpenVPN.

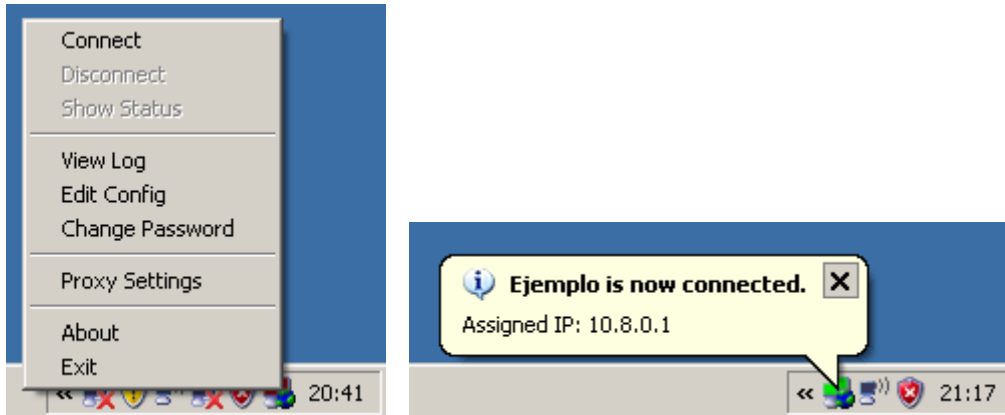


Figura 45. Icono del Proceso de la GUI Instalada en Windows XP

Cuando terminamos la instalación de OpenVPN en Windows, añadimos una interfaz de red más (virtual) para la interfaz TUN/TAP, aunque se pueden añadir más de una si se desea establecer más de un túnel. Si establecemos un túnel entre dos puntos con OpenVPN utilizando Windows XP tendremos esa interfaz TUN/TAP funcionando y con una dirección MAC por defecto.



Figura 46. Interfaces de Red tras Instalar OpenVPN en Windows XP

Como se puede ver en las imágenes siguientes, cuando arrancamos el túnel, se añade una interfaz nueva, en este caso con la dirección IP 10.8.0.1 y con la dirección MAC 00:FF:47:59:52:8F.

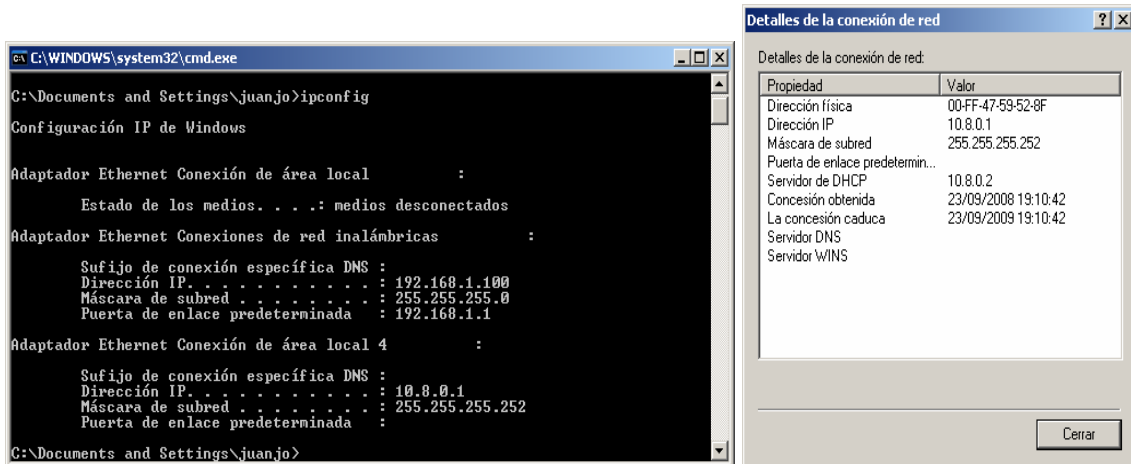


Figura 47. Dirección IP y MAC de la Interfaz Virtual en Windows XP

Por último, comentar que, aunque se haya instalado una GUI de OpenVPN en nuestro Windows, su uso se limita a conectar/desconectar el túnel, ver el estado de éste y poder modificar algunas opciones de Proxy. Es decir, esta GUI no proporciona una interfaz gráfica amigable para diseñar el fichero de configuración, por lo que se tendrá que diseñar a partir de un procesador de texto, de manera rudimentaria.

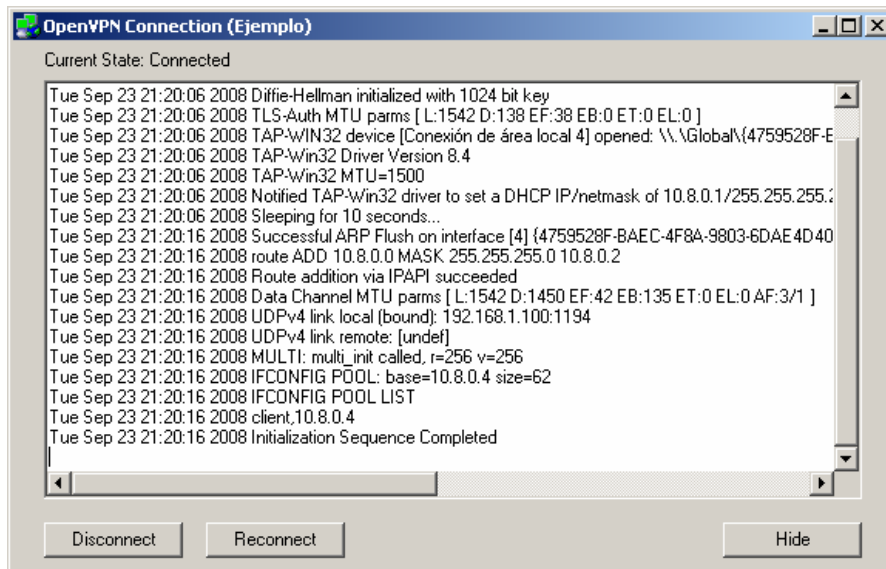


Figura 48. Estado de Conexión OpenVPN utilizando la GUI en Windows XP

3.1.3.- Instalación de OpenVPN en OpenSuSE

Instalar OpenVPN en una distribución SuSE de Linux es tan fácil como la instalación en Windows. En las distribuciones SuSE casi todos los procesos de administración del sistema pueden realizarse con la interfaz de administración YaST. En nuestro caso, OpenVPN puede ser instalado completamente mediante esta interfaz de administración del sistema de SuSE, es decir, YaST. El problema de este tipo de instalación, es que se necesita una versión no muy antigua de SuSE para poder instalar la versión 2.0.9 de

OpenVPN, ya que, en otro caso, puede que la versión que incluya de OpenVPN sea más vieja y no podamos contar con algunas de las características de la versión 2.0.9. Por ello, se ha utilizado OpenSuSE 10.2 como sistema operativo, ya que cuenta con la versión 2.0.9 de OpenVPN, que es la que se está utilizando en este proyecto fin de carrera. Para encontrar YaST en el entorno gráfico KDE de la distribución OpenSuSE 10.2 solo hay que entrar en el menú principal, luego en “System” y por último en “YaST”. Si no hemos entrado como root, el sistema nos pedirá el password de root para poder entrar en YaST.

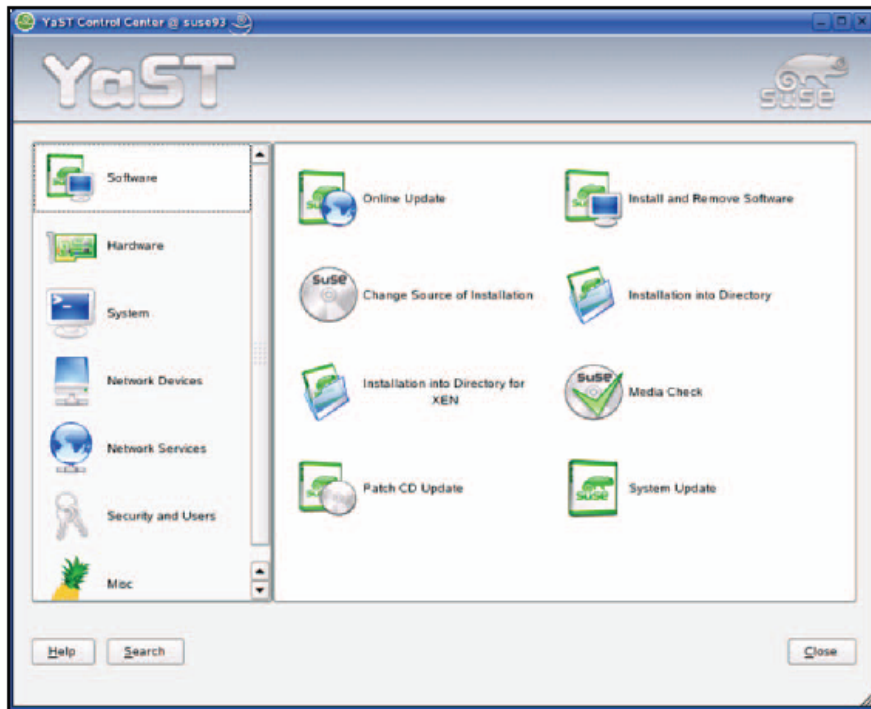


Figura 49. Aspecto Gráfico del Menú Principal de YaST en SuSE

El menú principal de YaST consiste en varios módulos diferentes, que son representados por símbolos en la parte derecha y agrupados por etiquetas en la izquierda. Para poder instalar OpenVPN tenemos que pinchar en la etiqueta de la izquierda “Software” y en el icono de la derecha llamado “Install and Remove Software”. De esta manera podremos entrar a la interfaz de instalación y desinstalación de paquetes de OpenSuSE.

El manejo del instalador y desinstalador de paquetes de YaST es muy sencillo. Para instalar OpenVPN solo se ha de introducir la palabra “openvpn” en el buscador y, una vez encontrado el paquete OpenVPN en la base de datos, poner un tic en la línea que aparece con el nombre del paquete y la descripción y, por último, pinchar en Accept. Además de poder instalarlo, también se puede ver información diversa, como una descripción técnica del paquete, los archivos que se van a instalar y las dependencias del paquete que vamos a instalar, que en nuestro caso es OpenVPN.

SuSE u OpenSuSE nos pedirá el CD o DVD de instalación de nuestra distribución para poder instalar OpenVPN. Esto no será así si hemos configurado YaST para que utilice los servidores Web/FTP del proyecto SuSE para su instalación.

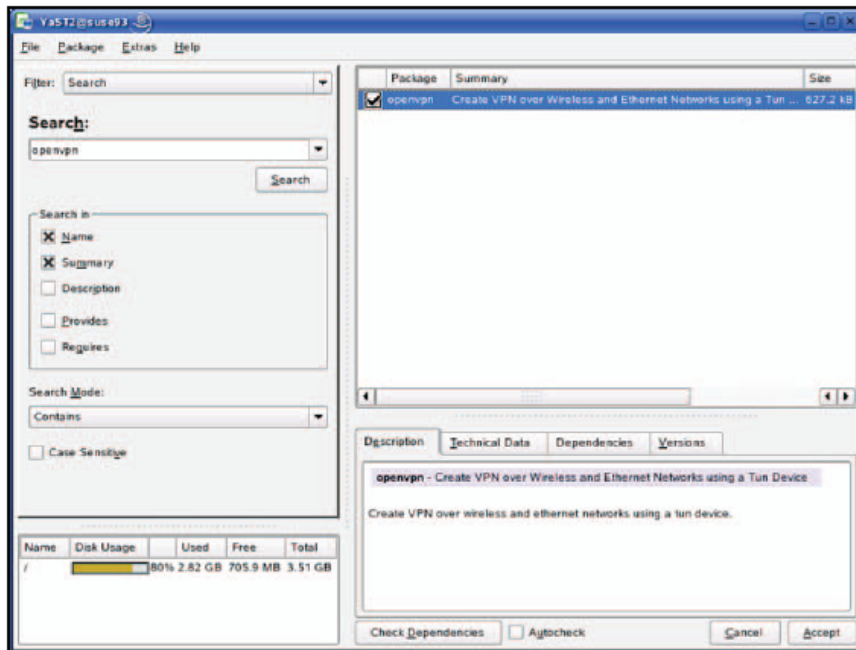


Figura 50. Instalador/Desinstalador de Paquetes de YaST en SuSE

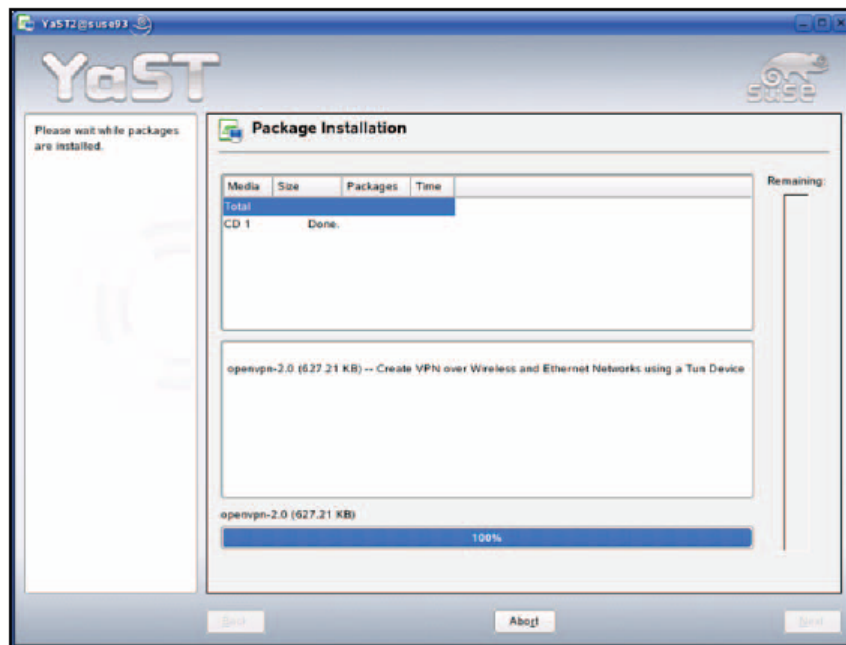


Figura 51. Progreso de la Instalación de OpenVPN en YaST de SuSE (I)

Los pasos son sencillos, copiar los archivos necesarios del CD/DVD de instalación o descargárselos de los servidores FTP/Web de SuSE se procede, en un siguiente paso, a

la instalación de OpenVPN. Tras finalizar la instalación YaST preguntará si se desea instalar otro paquete y tendremos que pinchar en “Finish”.

También hay que destacar que esta instalación de OpenVPN no incorpora ninguna interfaz gráfica ni paquetes extra, y no hay que olvidar que se han de instalar, previamente, los paquetes LZO, OpenSSL y comprobar que el kernel que disponemos tiene una versión superior a la 2.4 para que incorpore los controladores virtuales TUN/TAP. Por tanto, el manejo de OpenVPN se basa exclusivamente en comandos de consola ya que para poder contar con alguna herramienta gráfica habría que instalarla aparte (ver Anexo I: GUIs y otros Programas).

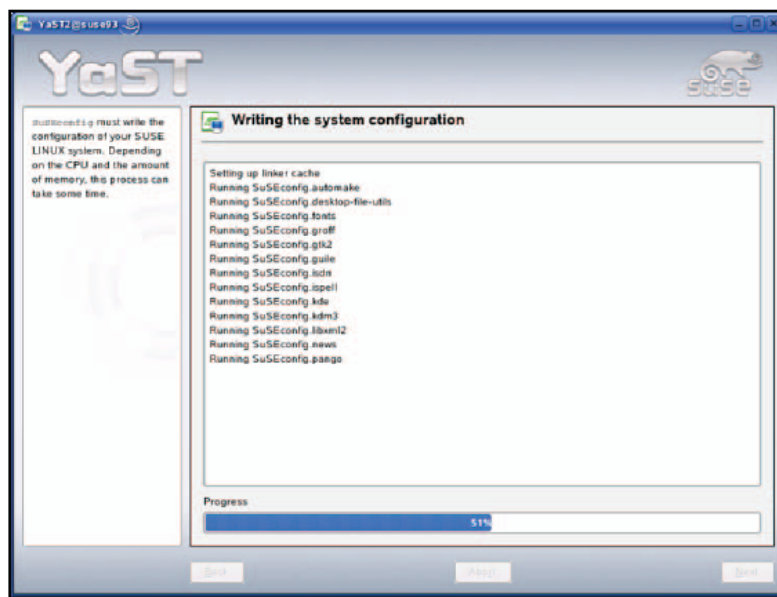


Figura 52. Progreso de la Instalación de OpenVPN en YaST de SuSE (II)

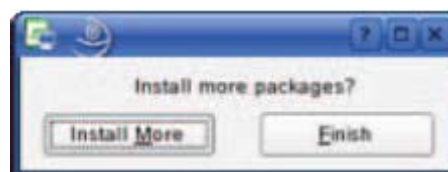


Figura 53. Fin de la Instalación de OpenVPN en YaST de SuSE

3.1.4.- Instalación de OpenVPN en sistemas RPM

Existen muchas distribuciones de Linux que manejan paquetes de tipo RPM gracias al RPM Package Manager, que es una herramienta de administración de paquetes pensada básicamente para Linux. Entre las distribuciones de Linux más conocidas que manejan paquetes RPM se encuentran Red Hat, SuSE (OpenSuSE en la actualidad) y Fedora.

La Web de OpenVPN no proporciona un paquete en formato RPM de OpenVPN, pero se puede hacer un paquete RPM a partir del fichero descargado de la Web de

OpenVPN mediante el comando, sencillo de utilizar, “rpm”. Para ello se seguirán los siguientes pasos, mediante comandos “rpm” en un terminal de consola, para hacer el paquete RPM e instalar OpenVPN a partir de dicho paquete RPM:

- **Paso 1:** construimos nuestro fichero RPM a partir del paquete de OpenVPN en código fuente descargado de la Web de OpenVPN:

```
rpmbuild -tb openvpn-[versión].tar.gz
```

- **Paso 2:** una vez se tiene el paquete RPM, se puede instalar como cualquier paquete RPM.

```
rpm -ivh openvpn-[detalles].rpm
```

- **Paso 3:** o bien, se puede actualizar la versión de una versión ya instalada mediante la siguiente línea.

```
rpm -Uvh openvpn-[detalles].rpm
```

En la siguiente tabla se muestran los archivos más importantes instalados, con sus rutas y su función principal:

Full Path and File Installed by OpenVPN	Function
/etc/openvpn	Directory containing configuration files
/etc/init.d/openvpn	Start/stop script for services
/usr/sbin/rcopenvpn	
/usr/sbin/openvpn	The binary
/usr/share/doc/openvpn	Documentation files
/usr/share/man/man8/openvpn.8.gz	Manual page
/usr/share/doc/openvpn/examples/sample-config-files	Example configuration files
/usr/share/doc/openvpn/examples/sample-keys	Example keys and certificates
/usr/share/doc/openvpn/examples/easy-rsa	easy-rsa—a collection of scripts useful for creating tunnels
/usr/share/doc/openvpn/changelog.Debian.gz	
/usr/share/doc/openvpn/changelog.gz	Version history
/usr/share/openvpn/verify-cn	verify-cn function (revoke command)
/usr/lib/openvpn/openvpn-auth-pam.so	Libraries for PAM-Authentication and chroot mode
/usr/lib/openvpn/openvpn-down-root.so	
/usr/share/doc/packages/openvpn/suse	
/usr/share/doc/packages/openvpn/suse/openvpn.init	SuSE-specific start/stop scripts
/var/run/openvpn	Process ID of the running OpenVPN process

Figura 54. Archivos Instalados de OpenVPN y su Función en Sistemas RPM

3.1.5.- Instalación de OpenVPN en Debian

Probablemente la distribución donde más fácil se instala OpenVPN es Debian o las distribuciones basadas en Debian, tal como la familia de distribuciones Ubuntu (Ubuntu, Kubuntu, Xubuntu,...). La instalación de OpenVPN en sistemas Debian se realiza mediante el comando “apt”. El sistema de gestión de paquetes APT simplifica la instalación y eliminación de programas en distribuciones tipo Debian. Para instalar OpenVPN mediante el comando “apt” se ha de introducir, simplemente, la siguiente línea de comando en un terminal de consola:

`apt-get install openvpn`

```

debian01:~# apt-get install openvpn
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  openvpn
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 293kB of archives.
After unpacking 762kB of additional disk space will be used.
Get:1 http://ftp.uni-erlangen.de testing/main openvpn 2.0-4 [293kB]
Fetched 293kB in 1s (247kB/s)
Preconfiguring packages ...
Selecting previously deselected package openvpn.
(Reading database ... 9727 files and directories currently installed.)
Unpacking openvpn (from ../openvpn_2.0-4_i386.deb) ...
Setting up openvpn (2.0-4) ...
Restarting virtual private network daemon:.

debian01:~#

```

Figura 55. Instalación de OpenVPN mediante APT en un Sistema Debian

Full Path and File Installed by OpenVPN	Function
/etc/openvpn	Directory containing configuration files
/etc/network/if-up.d/openvpn	Start/stop openvpn when the network goes up/down
/etc/network/if-down.d	
/etc/network/if-down.d/openvpn	
/etc/init.d/openvpn	Start/stop script for services
/sbin/openvpn	The binary
/usr/share/doc/openvpn	Documentation files
/usr/share/man/man8/openvpn.8.gz	Manual page
/usr/share/doc/openvpn/examples/sample-config-files	Example configuration files
/usr/share/doc/openvpn/examples/sample-keys	Example keys
/usr/share/doc/openvpn/examples/easy-rsa	easy-rsa—a collection of scripts useful for creating tunnels
/usr/share/doc/openvpn/changelog.Debian.gz	Version history
/usr/share/doc/openvpn/changelog.gz	
/usr/share/openvpn/verify-cn	verify-cn function (revoke command)
/usr/lib/openvpn/openvpn-auth-pam.so	Libraries for PAM-Authentication and chroot mode
/usr/lib/openvpn/openvpn-down-root.so	

Figura 56. Archivos Instalados de OpenVPN y su Función en Sistemas Debian

3.1.6.- Instalación de Open VPN mediante Código Fuente

El último método de instalación que se explicará de OpenVPN, no depende del sistema de administración de paquetes que se utilice. Simplemente el sistema operativo ha de contar herramientas de desarrollo tales como un compilador de C y la herramienta “make”. Para la instalación mediante este método se utilizará un sistema operativo Debian y habrá que descargarse el código fuente de OpenVPN desde su Web. Los comandos que se han de utilizar son simples “make” y “configure”, para ello solo hay que seguir los siguientes pasos:

- **Paso 1:** descomprimir el paquete comprimido para obtener el código fuente.
`tar xzf openvpn-[versión].tar.gz`
- **Paso 2:** desde el nivel más alto del directorio donde se encuentran los ficheros fuente de OpenVPN ejecutar los siguientes comandos:

```
./configure
debian01:~/openvpn-2.0.2# ./configure
checking for ifconfig... /sbin/ifconfig
checking for ip... ip
checking for route... /sbin/route
checking build system type... i686-pc-linux
checking host system type... i686-pc-linux
checking target system type... i686-pc-linux
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
(...)
checking for SSL_CTX_new in -lssl... yes
configure: creating ./config.status
config.status: creating Makefile
config.status: creating openvpn.spec
config.status: creating config-win32.h
config.status: creating install-win32/openvpn.nsi
config.status: creating config.h
config.status: executing depfiles commands
debian01:~/openvpn-2.0.2#
```

Figura 57. Instalación de OpenVPN mediante el Código Fuente (I)

```
make
debian01:~/openvpn-2.0.2# make
make all-am
make[1]: Entering directory `/root/openvpn-2.0.2'
if gcc -DHAVE_CONFIG_H -I. -I. -I. -I. -g -O2 -MT base64.o -MD -MP -MF
.deps/base64.Tpo" -c -o base64.o base64.c; \
then mv -f ".deps/base64.Tpo" ".deps/base64.Po"; else rm -f
.deps/base64.Tpo"; exit 1; fi
(...)
```

Figura 58. Instalación de OpenVPN mediante el Código Fuente (II)

```
make install
debian01:~/openvpn-2.0.2# make install
make[1]: Entering directory `/root/openvpn-2.0.2'
test -z "/usr/local/sbin" || mkdir -p -- . "/usr/local/sbin"
/usr/bin/install -c 'openvpn' '/usr/local/sbin/openvpn'
test -z "/usr/local/man/man8" || mkdir -p -- . "/usr/local/man/man8"
/usr/bin/install -c -m 644 './openvpn.8' '/usr/local/man/man8/openvpn.8'
make[1]: Leaving directory `/root/openvpn-2.0.2'
debian01:~/openvpn-2.0.2#
```

Figura 59. Instalación de OpenVPN mediante el Código Fuente (III)

Aunque en este ejemplo se ha descrito la instalación de manera directa, existen más opciones para el comando `configure` que pueden listarse utilizando la opción `--help` de dicho comando.

3.1.7.- Otros Aspectos a Tener en Cuenta

Cuando tengamos OpenVPN instalado en nuestro sistema operativo Linux se debe de crear, cargar y habilitar el routing del dispositivo TUN/TAP que tenemos integrado en el kernel de Linux, en el caso de utilizar una versión 2.4 o mayor de kernel. Para ello debemos de introducir los siguientes comandos en un terminal de consola:

- **Paso 1:** creamos un nodo dispositivo.
`mknod /dev/net/tun c 10 200`
- **Paso 2:** se añade al fichero `/etc/modules.conf` la siguiente línea.
`alias char-major-10-200 tun`
- **Paso 3:** se carga el driver TUN/TAP.
`modprobe tun`
- **Paso 4:** se habilita el routing en el driver TUN/TAP.
`echo 1 > /proc/sys/net/ipv4/ip_forward`

Los pasos 1 y 2 solo se han de realizar una vez en el sistema operativo, sin embargo, los pasos 3 y 4, en algunas distribuciones, sobre todo en las antiguas, se han de realizar cada vez que se reinicie el sistema operativo.

Por el contrario, en máquinas equipadas con algún sistema operativo Windows estos pasos no son necesarios, aunque el paquete OpenVPN también proporciona scripts para crear, actualizar y eliminar drivers TUN/TAP.

3.1.8.- Ejemplo de un Túnel Sencillo

A continuación se va a crear un túnel sencillo entre dos máquinas Linux sin utilizar ninguno de los dos modos de seguridad (el modo mediante claves pre-compartidas o mediante certificados utilizando para ello TLS). Para ello simplemente, se creará un fichero de configuración por cada extremo del túnel. A estos ficheros de configuración de OpenVPN los llamaremos `Extremo_A.conf` y `Extremo_B.conf`:

Extremo_A.conf	Extremo_B.conf
<code>remote IPextremoB</code>	<code>remote IPextremoA</code>
<code>dev tun</code>	<code>dev tun</code>
<code>ifconfig 10.8.0.1 10.8.0.2</code>	<code>ifconfig 10.8.0.2 10.8.0.1</code>
<code>verb 6</code>	<code>verb 6</code>

Figura 60. Archivos de Configuración de un Túnel Sencillo con OpenVPN

Este ejemplo solo cuenta con 4 directivas: con la directiva “remote” le indicamos al túnel cual es la IP real del otro extremo, con la directiva “dev tun” indicamos que lo que

se quiere implementar es un túnel y no un puente Ethernet, con la directiva “ifconfig” asignamos las IP virtuales que van a tener los dos extremos del túnel y con la directiva “verb” indicamos el grado de información que nos va a devolver la consola durante la conexión o por si hubiera algún error. También hay que destacar que, aunque no se ha utilizado la opción “proto”, OpenVPN utiliza por defecto el protocolo UDP como protocolo de transporte del túnel.

Una vez que se han creado los ficheros de configuración hay que ejecutarlos utilizando para ello la directiva --config, comprobando que el túnel se establece con éxito. En la imagen que se puede ver a continuación se ha establecido el túnel entre dos máquinas Windows.

openvpn --config Extremo_A.conf (en una de las máquinas)

openvpn --config Extremo_B.conf (en la otra máquina)

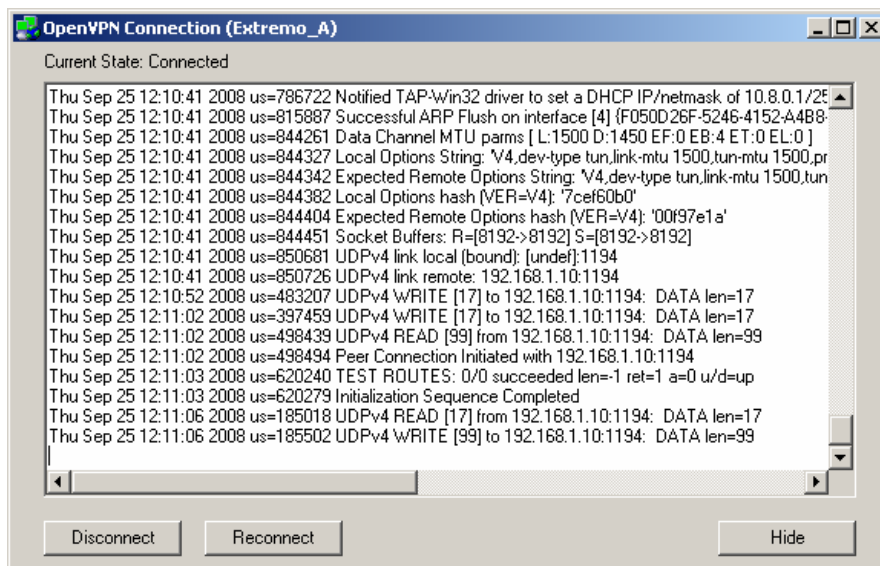


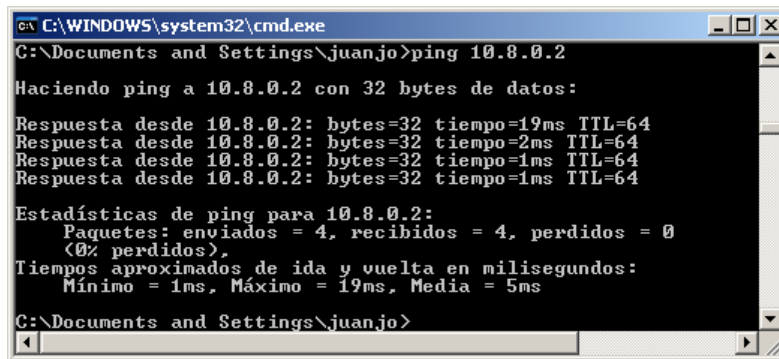
Figura 61. Archivos de Configuración de un Túnel Sencillo con OpenVPN

Para comprobar que el túnel se ha establecido con éxito se pueden comprobar las interfaces con las que se cuenta en el sistema Windows mediante el comando “ipconfig” y también utilizar el comando “ping” al otro extremo del túnel para comprobar que hay conectividad.

3.2.- Implementación de la Seguridad

Para implementar la seguridad en los túneles OpenVPN y en las soluciones en las que implementemos un servidor al que se conectan los clientes OpenVPN se tendrá que hacer uso de algunos scripts para añadir algún método de seguridad a nuestra VPN. Dichos scripts se basan en líneas en las que se utiliza el comando “openssl” junto a sus posibles opciones, proporcionado por el paquete criptográfico OpenSSL. Con estos scripts podremos crear claves, certificados, Autoridades Certificadoras,... necesarias cuando vamos a utilizar algún modo de seguridad, ya sea el modo de claves pre-

compartidas (static-keys) o el modo TLS utilizando certificados. Además, estos scripts pueden ser ejecutados tanto en sistemas operativos Windows (2000/XP/Vista) como en cualquier distribución de Linux.



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\juanjo>ping 10.8.0.2

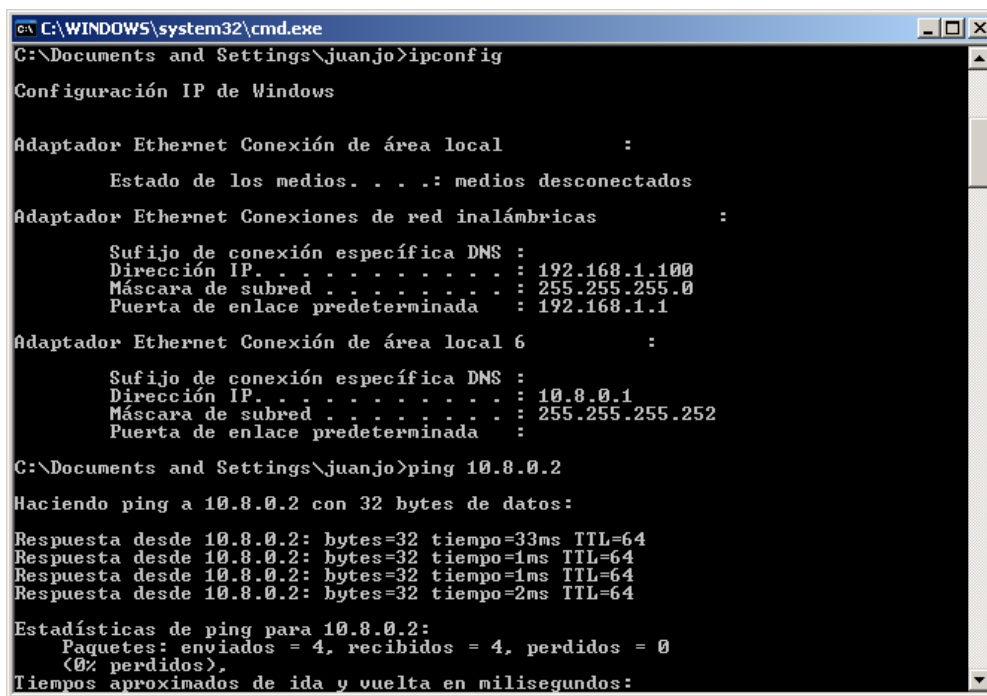
Haciendo ping a 10.8.0.2 con 32 bytes de datos:

Respuesta desde 10.8.0.2: bytes=32 tiempo=19ms TTL=64
Respuesta desde 10.8.0.2: bytes=32 tiempo=2ms TTL=64
Respuesta desde 10.8.0.2: bytes=32 tiempo=1ms TTL=64
Respuesta desde 10.8.0.2: bytes=32 tiempo=1ms TTL=64

Estadísticas de ping para 10.8.0.2:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 1ms, Máximo = 19ms, Media = 5ms

C:\Documents and Settings\juanjo>
```

Figura 62. Conectividad en un Túnel Sencillo con OpenVPN



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\juanjo>ipconfig

Configuración IP de Windows

Adaptador Ethernet Conexión de área local :

    Estado de los medios. . . : medios desconectados
Adaptador Ethernet Conexiones de red inalámbricas :

    Sufijo de conexión específica DNS :
    Dirección IP. . . . . : 192.168.1.100
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada : 192.168.1.1

Adaptador Ethernet Conexión de área local 6 :

    Sufijo de conexión específica DNS :
    Dirección IP. . . . . : 10.8.0.1
    Máscara de subred . . . . . : 255.255.255.252
    Puerta de enlace predeterminada :

C:\Documents and Settings\juanjo>ping 10.8.0.2

Haciendo ping a 10.8.0.2 con 32 bytes de datos:

Respuesta desde 10.8.0.2: bytes=32 tiempo=33ms TTL=64
Respuesta desde 10.8.0.2: bytes=32 tiempo=1ms TTL=64
Respuesta desde 10.8.0.2: bytes=32 tiempo=1ms TTL=64
Respuesta desde 10.8.0.2: bytes=32 tiempo=2ms TTL=64

Estadísticas de ping para 10.8.0.2:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
```

Figura 63. Conectividad en un Túnel Sencillo en Windows

Para poder utilizar el modo de seguridad mediante TLS tendremos que establecer una infraestructura de clave pública o PKI, que consiste en:

- Un certificado público (conocido también como clave pública) y una clave privada para el servidor y un certificado público y una clave privada para cada cliente.
- Un certificado público y clave privada de la Autoridad Certificadora (CA) que tendrá la función de firmar cada uno de los certificados del cliente y del servidor.

Tanto el servidor como el cliente se autenticarán verificando, primero, que el certificado que le ha presentado el otro participante fue firmado por la Autoridad Certificadora (CA) y, tras esto, comprobarán algunos campos del certificado, como el “common name” del certificado o el tipo de certificado (tipo cliente o tipo servidor). Este modelo de seguridad tiene algunas características que se desean en una VPN:

- El servidor solo necesita su propio certificado y clave privada. No es necesario que el servidor conozca los certificados de cada uno de los clientes que puedan conectarse.
- El servidor solo aceptará clientes cuyos certificados fueron firmados por el certificado de la Autoridad Certificadora (CA), ya que dicho certificado lo tienen tanto el servidor como los clientes. Además, como el servidor puede verificar si la firma es o no de la CA sin necesidad de acceder a la clave privada de esta CA (esta clave es la más importante de toda la infraestructura de clave pública, PKI), dicha clave privada puede residir en una máquina diferente a la del servidor e incluso en una máquina no conectada a la red.
- Si una clave privada (del servidor, de algún cliente o de la CA) está comprometida, esta puede ser deshabilitada añadiendo el certificado a una lista de revocación de certificados o CRL. La CRL permite rechazar certificados comprometidos sin necesitar hacer una reconstrucción de la infraestructura de clave pública o PKI.
- El servidor puede forzar a los clientes reglas de acceso específicas basadas en los campos del certificado, como el “common name”.

3.2.1.- Creación de una Clave Estática

Este apartado solo es aplicable en el caso de utilizar un modelo de seguridad no basado en TLS, es decir, basado en una clave pre-compartida (o modelo static-key). Este modelo de clave pre-compartida, como se ha comentado en apartados anteriores de esta memoria, consiste en que ambos extremos del túnel comparten la misma clave. Dicha clave ha tenido que ser creada previamente en una de las máquinas (u otra externa que no participa en la VPN) y debe de transferirse utilizando un canal seguro, para que no pueda ser interceptada por ningún intruso. Para ello se pueden usar aplicaciones tales como SCP o similares.

Para crear la clave estática e indicar a OpenVPN que se va a utilizar dicho modelo de seguridad no es necesario utilizar ningún script, simplemente se usará directivas de OpenVPN. En concreto se utilizan las directivas `--genkey` y `--secret`. Como se dijo en el apartado 2.4.1 se puede utilizar una clave compuesto por 2 subclaves o por 4 subclaves. Para ello, en el fichero de configuración de ambos extremos, se tiene que incluir la directiva “secret clave [direction]”, donde “direction” puede tomar los valores 0 (para 2 claves, por defecto) o 1 (para 4 claves).

El paso previo a la realización de los ficheros de configuración es la creación de la clave. Para ello utilizaremos las siguientes líneas de comando, que son iguales tanto para Windows como para Linux:

`openvpn --genkey --secret nombreClave`

Al instalar OpenVPN en Windows, este trae consigo un script para crear una clave estática en un solo paso. Simplemente hay que pinchar en Inicio->Programas->OpenVPN->Generate a Static OpenVPN key. Este script generará una clave estática que, como en el caso de utilizar Linux, tendrá un tamaño de 2048 bits.

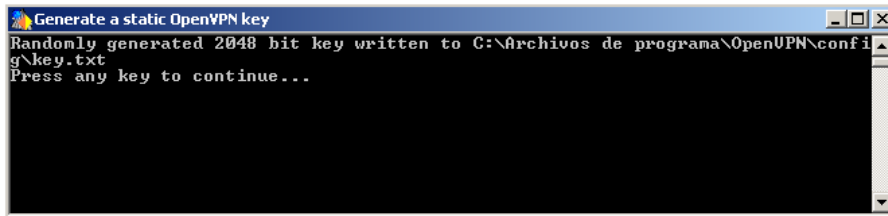


Figura 64. Programa para Generar una Clave Estática en Windows

3.2.2.- Creación de Clave/Certificado de la CA

En este apartado vamos a generar el certificado público y la clave privada de la CA. Para ello se va a hacer uso de los scripts proporcionados por OpenVPN que se encuentran en la carpeta “easy-rsa” en la ruta donde se haya instalado OpenVPN. El procedimiento para ejecutar dichos scripts en Windows y Linux son muy similares y se hacen mediante consola. La única diferencia es el formato, ya que en Windows se trata de archivos con extensión .bat, mientras que en Linux se trata de archivos de shell.

Si se está utilizando Windows, el primer paso será ejecutar el archivo “init-config.bat” desde la consola para copiar algunos archivos necesarios al mismo directorio (esto, además, sobrescribirá los archivos “vars.bat” y “openssl.cnf” ya existentes).

Una vez tenemos el fichero llamado vars (vars.bat en Windows), lo editamos con un editor de texto. Entre los parámetros que podemos modificar encontramos la ruta del fichero donde se crearán las claves y certificados, el tamaño de las claves privadas (del servidor, cliente y CA) para su uso en el algoritmo RSA, y los valores por defecto de algunos campos de los certificados tales como KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG y KEY_EMAIL. Tras haberlo editado ejecutamos el script.

Por último solo queda ejecutar el script build-ca (build-ca.bat en Windows) para generar la clave privada y el certificado de la Autoridad Certificadora (CA).

En la figura 66 se muestra la salida que ofrece por consola una máquina con Linux al crear el par certificado/clave de la CA. Nótese que algunos de los campos que se muestran en la imagen, durante la creación del certificado y clave privada de la CA, han sido fijados en el script vars (o vars.bat en Windows). Como se puede ver en la imagen el “common-name” del certificado de la CA es “PFC_CA”.

Windows	Linux
<code>init-config.bat</code>	<code>./vars</code>
<code>vars.bat</code>	<code>./clean-all</code>
<code>clean-all.bat</code>	<code>./build-ca</code>
<code>build-ca.bat</code>	

Figura 65. Comandos para Crear el Par Certificado/Clave de la CA

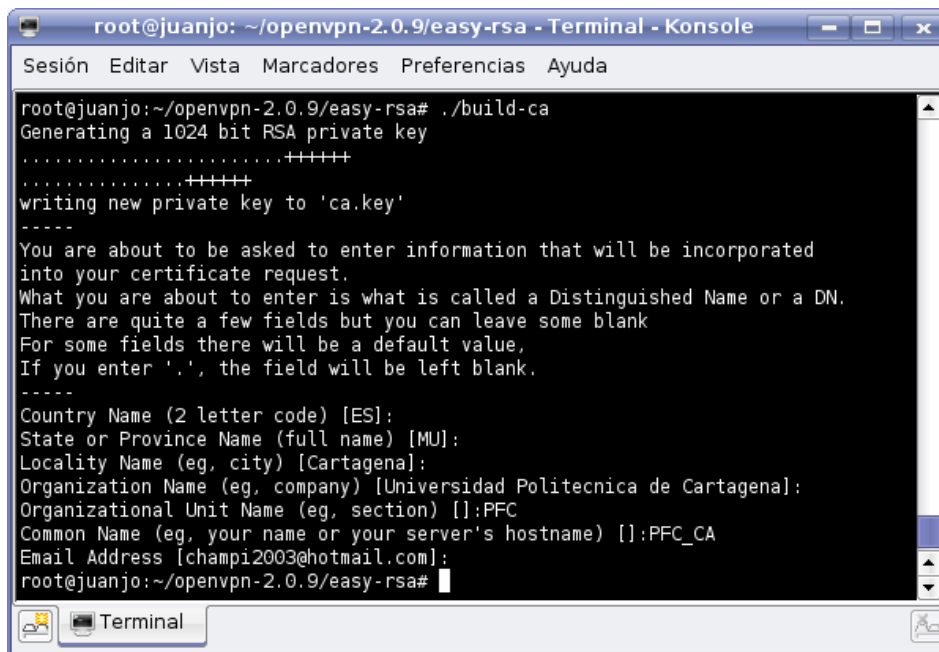


Figura 66. Creación del Par Certificado/Clave de la CA en Linux

Mediante este procedimiento hemos generado 2 ficheros nuevos

- **ca.crt**: fichero correspondiente al certificado público de la CA.
- **ca.key**: fichero correspondiente a la clave privada de la CA, la cual debe mantenerse protegida ya que es la clave más importante de toda la PKI.

3.2.3.- Creación de Clave/Certificado del Servidor

Los pasos para crear la clave privada y el certificado público del servidor son similares a los pasos seguidos para crear el mismo par para la CA. Primero se ha de asegurar que se han creado y asignado las variables que se utilizarán en el resto de scripts. Para ello volvemos a ejecutar el script vars (vars.bat en Windows). Para crear el par clave/certificado del servidor ejecutamos el script build-key-server (build-key-server.bat en Windows).

Como podemos comprobar en la imagen 68 se ha creado, en una máquina con Linux, el certificado/clave del servidor y se ha introducido, como campo “common name” del certificado, el nombre “server”. Tras completar todos los campos del certificado del

servidor nos preguntan si queremos firmar el certificado, utilizando para ello el certificado de la CA, y tendremos que responderle que si. Por último, nos preguntará si nos queremos comprometernos con el certificado que vamos a crear, y también deberemos de contestar si.

Windows	Linux
<code>vars.bat</code>	<code>./vars</code>
<code>build-key-server.bat nombreCert</code>	<code>./build-key-server nombreCert</code>

Figura 67. Comandos para Crear el Par Certificado/Clave del Servidor

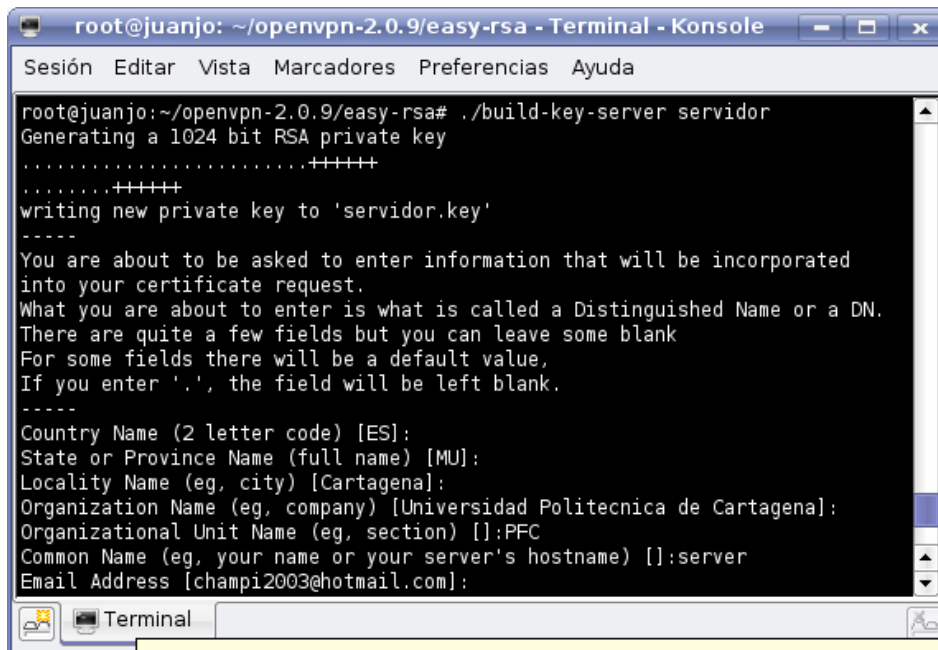


Figura 68. Creación del Par Certificado/Clave del Servidor en Linux (I)

Tras realizar estos pasos hemos generado 4 ficheros nuevos:

- **servidor.crt**: fichero correspondiente al certificado público del servidor.
- **servidor.key**: fichero correspondiente a la clave privada del servidor, la cual debe permanecer protegida.
- **01.pem**: fichero correspondiente al certificado público del servidor en formato PEM. El nombre del fichero proviene de su número de serie del certificado, el cual es 01.
- **servidor.csr**: este fichero sirve para poder crear el certificado del servidor en otra máquina que pueda crearlo y firmarlo, ya que este fichero tiene toda la información que le hace falta.

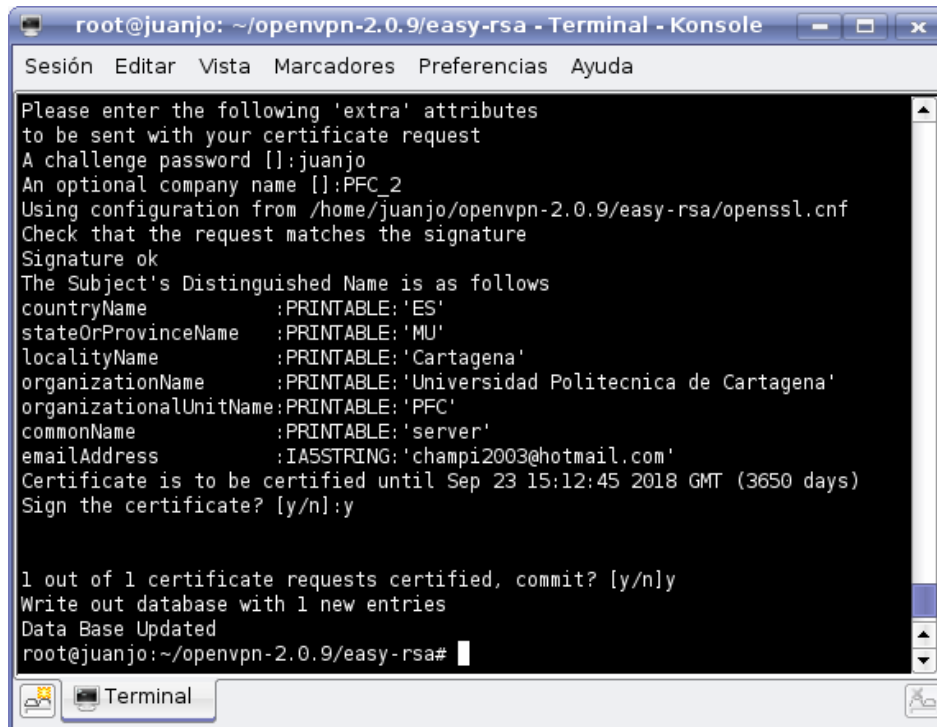


Figura 69. Creación del Par Certificado/Clave del Servidor en Linux (II)

3.2.4.- Creación de Clave/Certificado del Cliente

Para crear la clave privada y el certificado público de los clientes se hacen casi los mismos pasos que se han realizado para el servidor y para la Autoridad Certificadora (CA). De nuevo, se ha de ejecutar el script vars (vars.bat en Windows) para crear y asignar las variables que van a hacer falta para poder ejecutar el resto de scripts. En este caso, el script para crear la clave y el certificado del cliente se utiliza el script build-key (o build-key.bat en Windows).

Windows	Linux
vars.bat	. ./vars
build-key.bat nombreCertClient1	./build-key nombreCertClient1
build-key.bat nombreCertClient2	./build-key nombreCertClient2
build-key.bat nombreCertClient3	./build-key nombreCertClient3
...	...

Figura 70. Comandos para Crear el Par Certificado/Clave del Cliente

Al introducir el campo “common name” en cada uno de los certificados de cliente, conviene utilizar algún número para diferenciar uno de otro. Por ejemplo, podemos utilizar los valores de “common name”, client1, client2, client3, para los tres primeros clientes. Al igual que en el caso del servidor, tras completar todos los campos del certificado de cada cliente nos preguntan si queremos firmar el certificado, utilizando para ello el certificado de la CA, y tendremos que responderle que si. Por último, nos

preguntará si nos queremos comprometernos con el certificado que vamos a crear, y también deberemos de contestar si.

```

root@juanjo: ~/openvpn-2.0.9/easy-rsa - Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda
root@juanjo:~/openvpn-2.0.9/easy-rsa# ./build-key cliente
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'cliente.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [MU]:
Locality Name (eg, city) [Cartagena]:
Organization Name (eg, company) [Universidad Politecnica de Cartagena]:
Organizational Unit Name (eg, section) []:PFC
Common Name (eg, your name or your server's hostname) []:client
Email Address [champi2003@hotmail.com]:
    
```

Figura 71. Creación del Par Certificado/Clave del Cliente en Linux (I)

Tras realizar estos pasos hemos generado 4 ficheros nuevos:

- **client.crt**: fichero correspondiente al certificado público del cliente.
- **client.key**: fichero correspondiente a la clave privada del cliente, la cual debe permanecer protegida.
- **02.pem**: fichero correspondiente al certificado público del cliente en formato PEM. El nombre del fichero proviene de su número de serie del certificado, el cual es 02.
- **client.csr**: este fichero sirve para poder crear el certificado del cliente en otra máquina que pueda crearlo y firmarlo, ya que este fichero tiene toda la información que le hace falta.

3.2.5.- Creación de los Parámetros Diffie Hellman

Los parámetros Diffie Hellman deben ser generados para el servidor OpenVPN. Para ello se ha de volver a ejecutar el script vars (vars.bat en Windows) como se ha hecho en los casos anteriores y después ejecutar el script build-dh. El tamaño de estos parámetros dependerá del valor que le hayamos asignado a la variable KEY_SIZE en el script vars (lo normal es utilizar un tamaño de 1024 bits o 2048 bits). En este proyecto se ha utilizado un tamaño para los parámetros Diffie Hellman de 1024 bits.

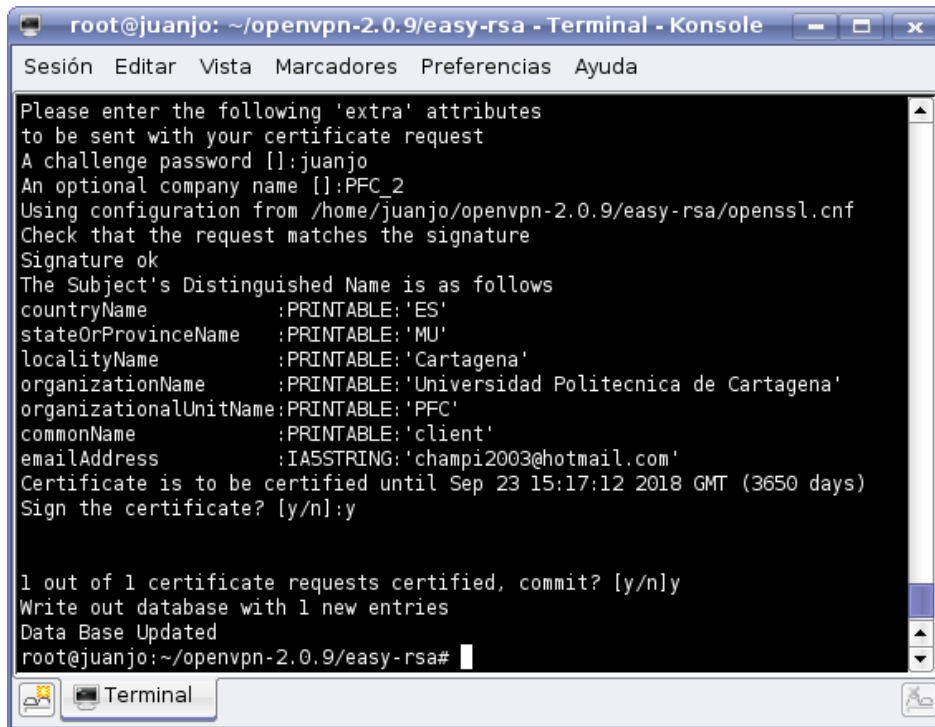


Figura 72. Creación del Par Certificado/Clave del Cliente en Linux (II)

Windows	Linux
<code>vars.bat</code>	<code>./vars</code>
<code>build-dh.bat</code>	<code>./build-dh</code>

Figura 73. Comandos para Crear los Parámetros Diffie Hellman

Tras ejecutar este script se genera un fichero en formato PEM llamado **dh1024.pem**. Los parámetros Diffie Hellman se utiliza para poder realizar el intercambio de una clave entre dos participantes de manera segura. Para ello se realizan una serie de funciones matemáticas que utilizan estos parámetros y que garantizan que solo los dos participantes conocerán la clave una vez se realice todo el proceso y funciones matemáticas.

3.2.6.- Creación de una CRL

En este apartado vamos a ver como crear una Lista de Revocación de Certificados o CRL mediante la utilización del script `revoke-full` (o `revoke-full.bat` en Windows) proporcionado para ello. La revocación de un certificado significa invalidar un certificado, que ha sido previamente firmado, para que no vuelva a ser utilizado con intenciones de autenticación. Algunas razones por las que revocar un certificado son:

- La clave privada asociada al certificado ha sido robada o está en peligro.
- El usuario de una clave privada olvida la contraseña de la clave.
- Se desea terminar con el acceso de un usuario a la VPN.

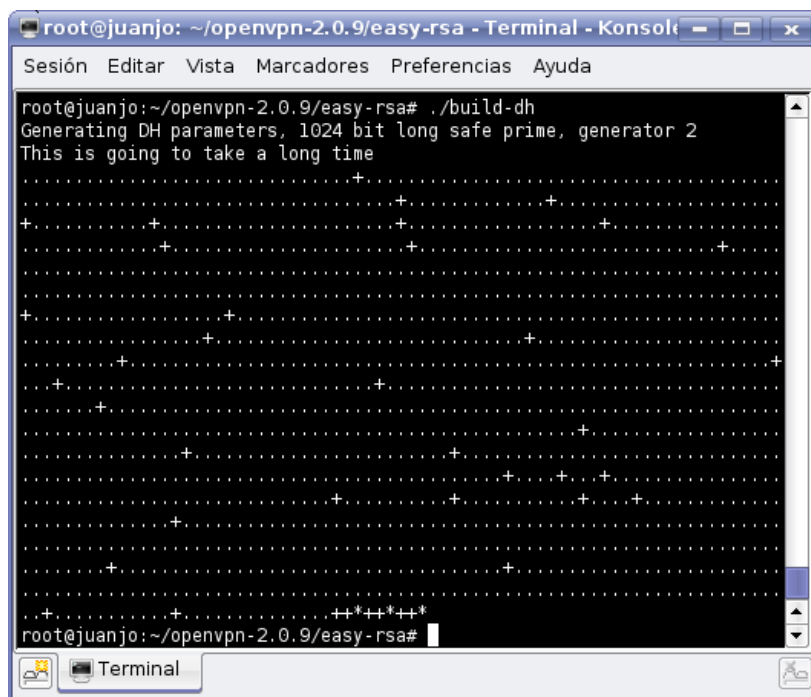


Figura 74. Creación de los Parámetros Diffie Hellman en Linux

Como ejemplo se han creado dos certificados llamados cliente1.crt y cliente2.crt para revocar, posteriormente, el certificado del cliente2. Para revocar el certificado del cliente2, llamado cliente2.crt solo se ha de introducir las siguientes dos líneas en consola:

Windows	Linux
<code>vars.bat</code>	<code>./vars</code>
<code>revoke-full.bat cliente2</code>	<code>./revoke-full cliente2</code>

Figura 75. Comandos para Crear los Parámetros Diffie Hellman

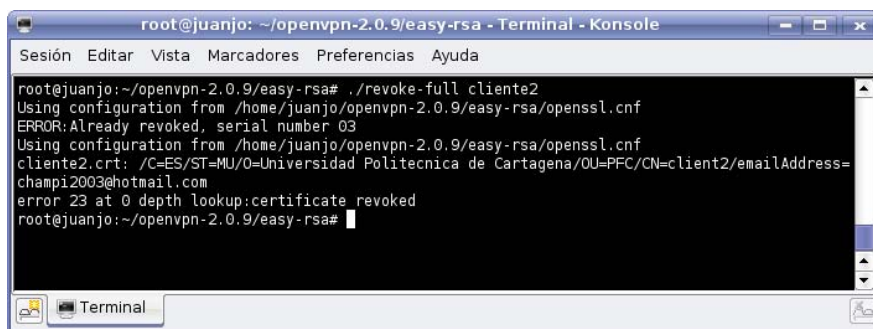


Figura 76. Creación de una CRL en Linux

Tras ejecutar estas dos líneas se genera un fichero llamado **crl.pem** que contiene la CRL que hemos creado. Este fichero se deberá de copiar en el lado del servidor OpenVPN para poder habilitar la CRL e impedir que los clientes cuyo certificado este revocado no puedan autenticarse y, por tanto, no puedan acceder a la VPN. Para poder habilitar la Lista de Revocación de Certificados o CRL en el lado del servidor tendremos que

añadir una línea con la directiva “crl-verify” de OpenVPN en el fichero de configuración del servidor, indicando el fichero que contiene la CRL:

```
crl-verify crl.pem
```

Una vez configurado el servidor con la CRL creada, se puede comprobar que ésta funciona de manera correcta. Para ello, una vez el servidor está arrancado, intentamos acceder a la VPN con un cliente que utiliza el certificado con nombre cliente2.crt. Como podemos ver en la figura 77 y en la figura 78, este cliente no se puede conectar al servidor, ya que está utilizando el certificado cliente2.crt que ha sido revocado por la CRL. Si intentáramos acceder como cliente pero utilizando otro certificado, como el cliente1.crt, si podríamos acceder a la VPN con éxito.

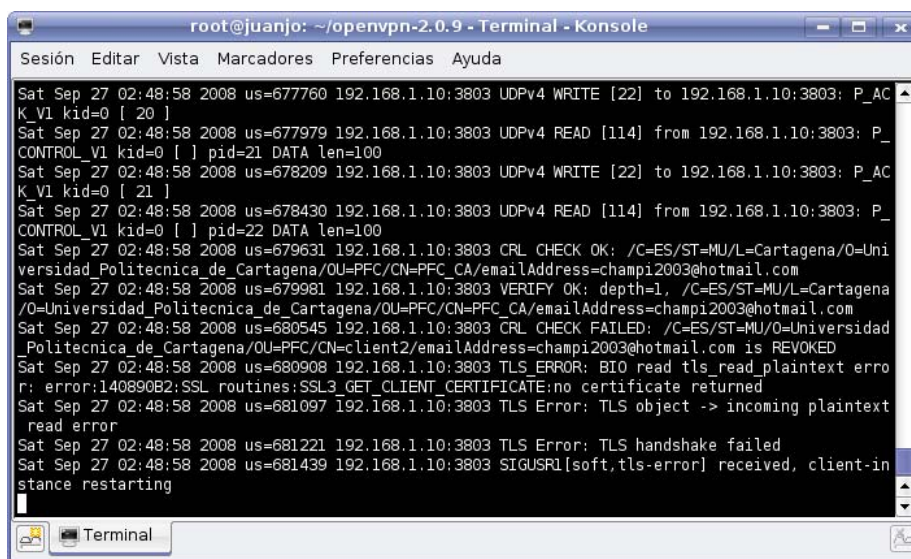


Figura 77. Revocación de un Certificado en el lado del Servidor por la CRL

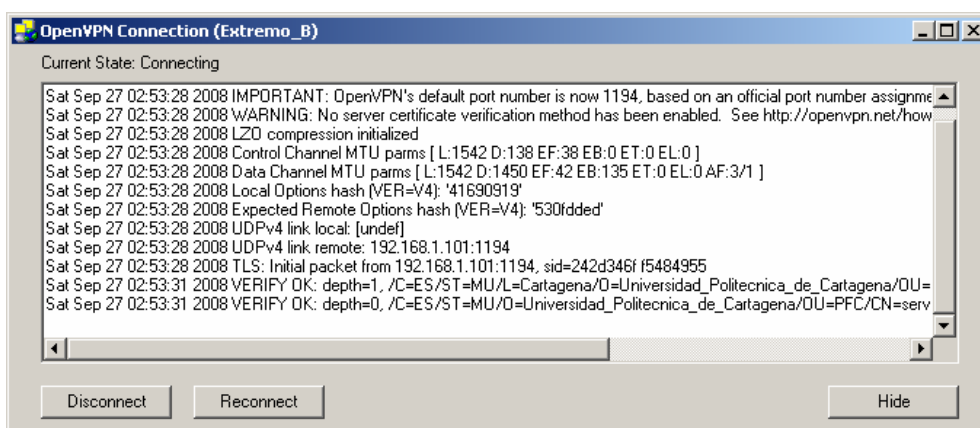


Figura 78. Revocación de un Certificado en el lado del Cliente por la CRL

Cuando utilizamos Listas de Revocación de Certificados o CRL tendremos que tener algunos aspectos en cuenta:

- Cuando utilizamos la opción de OpenVPN “crl-verify”, el fichero que contiene la CRL (crl.pem) será releído cada vez que se conecta un nuevo cliente o cada vez que un cliente renegocie la conexión SSL/TLS (cada hora, por defecto). Esto significa que se puede actualizar el fichero CRL mientras el demonio del servidor OpenVPN está ejecutándose, y tener inmediatamente los efectos de la CRL actualizada. Si el cliente al que se le ha revocado el certificado esta conectado en el momento en el que se ha actualizado la CRL, se puede reiniciar el servidor mediante una señal como SIGUSR1 o SIGHUP.
- Como la directiva “crl-verify” puede ser utilizada en ambos extremos de la VPN, clientes y servidor, no es necesario, generalmente, distribuir el fichero con la CRL a los clientes excepto si el certificado del servidor también ha sido revocado.
- El fichero de la CRL (crl.pem en nuestro ejemplo) no es privado y puede ser leído por cualquiera.
- Si se utiliza la directiva “--chroot” (será explicada brevemente en el siguiente apartado) se ha de copiar el fichero de la CRL en el directorio al que llamamos “jaula” (o subdirectorios).
- Una razón típica por la que se necesita revocar un certificado es porque el usuario de dicho certificado cifra su clave privada con un password y, posteriormente, olvida ese password. Revocando el certificado original es posible generar un nuevo par certificado/clave privada con el campo “common name” que se utilizó para el certificado revocado original.

3.2.7.- Más Técnicas para Proteger la VPN

Una vez se ha establecido el modelo de seguridad en nuestra VPN tenemos que pensar si este modelo es suficiente y podemos confiar en utilizar solo este nivel de seguridad o si, por el contrario, tenemos que implementar alguna medida de seguridad o capa de seguridad extra que proteja nuestro sistema en caso de que alguna de las capas de seguridad sea vulnerable a algún tipo de ataque.

La primera técnica para añadir a nuestro modelo de seguridad lo proporciona la directiva “--tls-auth”. Añadiendo esta opción en ambos extremos de la VPN añadimos un HMAC extra a todos los paquetes del protocolo Handshake de SSL/TLS para añadir un nivel más de autenticación. De esta manera se podría decir que estamos añadiendo un firewall HMAC, ya que cualquier paquete UDP que no tenga la misma HMAC será tirado sin realizar ningún tipo de procesamiento ni malgastar recursos extra. De esta manera podemos proteger la VPN de:

- Ataques DoS o ataques por inundaciones en el puerto OpenVPN.
- Escaneos de puerto para hallar que puertos UDP del servidor están abiertos.
- Vulnerabilidades por desbordamientos de buffer en la implementación de SSL/TLS.
- Intentos de iniciaciones del protocolo Handshake de SSL/TLS de máquinas no autorizadas.

El uso de la directiva “--tls-auth” requiere de una clave secreta compartida entre ambos extremos del túnel que será utilizada junto con los pares certificado/clave privada de ambos extremos (para utilizar en RSA). Para generar esta clave se puede utilizar el mismo paso realizado para generar la clave estática en el modelo de seguridad mediante claves estáticas (o pre-compartidas). Además, como sucedía en dicho modelo mediante claves pre-compartidas, se tendrá que transferir la clave de un extremo a otro mediante un canal seguro, por ejemplo, utilizando WinSCP (o aplicación similar). Como ejemplo, llamaremos al fichero que contiene la clave `ta.key`:

```
openvpn --genkey --secret ta.key
```

Una vez tengamos la clave en cada uno de las máquinas que componen la VPN añadimos la directiva “tls-auth” a los ficheros de configuración:

```
tls-auth ta.key 0 (en el lado del servidor)
```

```
tls-auth ta.key 1 (en el lado de los clientes)
```

La segunda recomendación es la de utilizar UDP como protocolo de transporte de OpenVPN. Esto no ofrece ninguna capa extra de seguridad al modelo ya implementado, pero permite que OpenVPN trabaje de manera más eficiente y que no tenga problemas en el caso de utilizar TCP, por ejemplo, cuando tengamos una conexión muy congestionada. Como ya se ha comentado en capítulos anteriores para indicar UDP como protocolo de transporte se utiliza la opción “--proto” mediante la siguiente línea en todos los ficheros de configuración de las máquinas de la VPN:

```
proto udp
```

En los sistemas operativos UNIX podemos restringir los privilegios del demonio de OpenVPN. Si queremos que el demonio de OpenVPN no tenga casi privilegios se ha de hacer que dicho demonio funcione con el usuario “nobody” y el grupo “nobody”. “Nobody” es un usuario o grupo con los privilegios justos para hacer funcionar OpenVPN, pero no para hacer mucho más. Si un atacante encuentra una vulnerabilidad con la que pueda romper OpenVPN, podrá realizar cualquier operación que lo permisos del usuario (o grupo) que OpenVPN tiene. Si este usuario de OpenVPN es “nobody”, el atacante tendrá las capacidades para realizar operaciones muy limitadas, por lo que no podrá hacer mucho daño a la máquina afectada. Para aplicar esta técnica de reducción de privilegios tanto para usuario como para grupo tendremos que utilizar las directivas “--user” y “--group”, por lo que habrá que introducir en cada uno de los archivos de configuración las siguientes dos líneas:

```
user nobody
```

```
group nobody
```

La siguiente recomendación sobre seguridad para nuestra VPN se complementa con la anterior. La directiva “--chroot” permite bloquear el demonio OpenVPN en lo que se conoce como una jaula, donde el demonio no será capaz de acceder a ningún directorio de la máquina excepto al indicado por la opción “--chroot”. Por ejemplo:

```
chroot jail
```

Si introducimos esta línea en el fichero de configuración hará que OpenVPN quede encerrado en el directorio dado por el parámetro “jail” por lo que dicho demonio no podrá acceder a ningún fichero que se encuentre fuera de dicha jaula. Este método es importante desde el punto de vista de la seguridad ya que si un ataque compromete al servidor, por ejemplo, ejecutando algún tipo de código malicioso, éste será bloqueado ya que no podrá acceder a ningún fichero ni directorio que no se encuentre dentro de lo que hemos llamado directorio jaula.

Como se dijo en el apartado 3.2.2, dentro del conjunto de scripts para generar los certificados y claves privadas de cada uno de los componentes de la VPN encontramos un script en el que se puede editar la variable `KEY_SIZE`. Esta variable indica el tamaño de las claves privadas que se van a generar, así como el tamaño de la clave Diffie Hellman. Por defecto, la variable `KEY_SIZE` tiene tamaño 1024 bits, por lo que es recomendable utilizar un tamaño mayor como por ejemplo 2048 o 4096 bits. Este incremento no tendrá un impacto negativo en las prestaciones de la VPN, excepto una negociación a través del protocolo Handshake de SSL/TLS un poco más lenta y un proceso de generación de parámetros Diffie Hellman más lento (mediante el script `build-dh`).

Por defecto OpenVPN utiliza, como algoritmo de cifrado simétrico, el algoritmo Blowfish con un tamaño de clave, por defecto, de 128 bits. Para muchos usuarios este tamaño de clave le parecerá demasiado corto. OpenVPN soporta cualquier algoritmo de cifrado que proporcione la librería OpenSSL. Por ejemplo, podemos utilizar el algoritmo AES, muy utilizado en la actualidad, con un tamaño de clave de 256 bits, añadiendo a los archivos de configuración la siguiente línea:

```
cipher AES-256-CBC
```

Como se ha visto en el apartado 3.2.2, al crear el par certificado/clave privada de la CA, se genera un archivo llamado `ca.key`, que contiene la clave privada de la CA. Esta clave privada es la más importante y la única que no se utiliza en todo el procedimiento de intercambio de claves. Por tanto, una buena practica relacionada con la seguridad de nuestra VPN es la de tener la clave privada de la CA en otra máquina que no esté conectada a ninguna red o en cualquier dispositivo de almacenamiento con el fin de que no pueda ser interceptada por ningún intruso.

3.2.8.- Técnicas Contra Ataques Man-in-the-middle

En criptografía, un ataque man-in-the-middle es un ataque en el que el enemigo adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido violado. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas. El ataque man-in-the-middle es particularmente significativo en el protocolo original de intercambio de claves de Diffie-Hellman, cuando éste se emplea sin autenticación. La posibilidad de un ataque man-in-the-middle sigue siendo un problema potencial de seguridad serio, incluso para muchos sistemas criptográficos basados en clave pública.

En OpenVPN existen técnicas de defensa contra estos tipos de ataques como los que se explican, brevemente, a continuación:

- Crear nuestros certificados de servidor con los scripts comentados en apartados anteriores. Ejecutando el script “build-key-server” se creará un certificado exclusivo de servidor asignándolo como “nsCertType=server” (de esta manera se le dice al protocolo SSL/TLS que este certificado es de tipo servidor). Tras hacer el certificado del servidor por este procedimiento podemos hacer que los clientes no acepten ningún certificado que no sea del tipo servidor utilizando, para ello, la directiva “--ns-cert-type”. Por tanto, una buena recomendación para prevenir ataques man-in-the-middle es la de incluir en el fichero de configuración de los clientes la siguiente línea:

```
ns-cert-type server
```

- Utilizar la directiva “--tls-remote name” para que el cliente o el servidor solo acepte certificados que tengan el campo “common name” con el valor del parámetro “name”. De esta manera, podemos hacer que los clientes solo acepten/rechacen conexiones basándose en el “common name” del certificado del servidor. Por ejemplo, en nuestro caso, para que los clientes solo acepten certificados del servidor que tenga el “common name” como “server” (ya que se ha establecido así en el apartado 3.2.3) habría que poner la siguiente línea en el fichero de configuración de los clientes:

```
tls-remote server
```

- Utilizar algún plugin o script junto a la directiva “--tls-verify” de OpenVPN para que el servidor acepte o rechace conexiones de clientes basándose en la comprobación de algunos de los detalles de los certificados X509. Esta función es muy útil si el cliente del que confiamos tiene un certificado el cual fue firmado por una CA que también firmó otros certificados a otros clientes, los cuales no tenemos la confianza de todos ellos, pero se quiere hacer una selección de algunos certificados que si serán aceptados. Esta directiva nos permitirá pasarle como argumento cualquier plugin o script que testeará alguno de los campos del certificado X509, como por ejemplo, el campo “common-name”. OpenVPN proporciona un script, llamado “verify-cn”, para utilizarlo junto con la directiva “--tls-verify” que, incluyéndolo en el lado del servidor, comprobará el campo “common-name” del certificado que le pase cualquier cliente:

```
tls-verify \./verify-cn valorDelCommonName"
```

- Firmar los certificados del servidor con una CA y los certificados del cliente con una CA diferente. La directiva “--ca” del fichero de configuración del cliente deberá hacer referencia al fichero de la CA del servidor, mientras que la directiva “--ca” del fichero de configuración del servidor deberá de hacer referencia al fichero de la CA del cliente.

3.3.- Otros Métodos de Autenticación

OpenVPN 2.0 incluye algunas características que permiten al servidor OpenVPN mantener un cierto nivel de seguridad mediante el uso de login y password de un cliente que quiera conectarse y utilizar esta información del cliente para que se autentique al servidor.

Para poder implementar este método de autenticación primero se tendrá que añadir al fichero de configuración del cliente la directiva “--auth-user-pass”. Dicha línea le dirá al cliente de OpenVPN que va a requerir su login y su password para poder conectarse al servidor. Hay que destacar que el login y password del cliente se pasa al servidor sobre el canal seguro proporcionado por TLS, por lo que no podrá ser interceptado.

El siguiente paso será configurar el servidor para usar autenticación mediante login y password basado en algún plugin, el cual puede ser un script, un fichero objeto compartido o un plugin DLL. El servidor OpenVPN realizará una llamada al plugin que se utilice cada vez que un cliente de la VPN intente conectarse, pasándole el login y password proporcionado por dicho cliente. El plugin para la autenticación mediante login y password puede controlar si el servidor OpenVPN permite o no conectar al cliente que ha proporcionado dicho login y password. Para el plugin devolverá el valor 1 si falla la autenticación o el valor 0 si la autenticación se realiza con éxito.

Hay que recordar que estos plugins utilizan el paquete PAM (Pluggable Authentication Module) y algunos ficheros y cabeceras del paquete PAM-devel. Como se dijo en el apartado 2.2.6, PAM no está disponible para sistemas operativos Windows, por lo que se tendrá que utilizar algún sistema operativo UNIX para implementar el servidor OpenVPN de manera que éste autentificará a los clientes que quieran conectarse a la VPN.

3.3.1.- Login y Password Mediante Scripts Para Plugins

En OpenVPN 2.0 se pueden utilizar scripts para plugins mediante la utilización de la directiva de OpenVPN “--auth-user-pass-verify” en el archivo de configuración del servidor. OpenVPN proporciona un script escrito en lenguaje Perl con el nombre de “auth-pam.pl” para utilizar junto con la directiva “--auth-user-pass-verify”. El script “auth-pam.pl” autentificará a los usuarios de la VPN en un servidor de Linux utilizando PAM, el cual podría combinarse junto con autenticación Shadow Password, RADIUS o LDAP. La siguiente línea, en el fichero de configuración del servidor, basta para poder implementar este método:

```
auth-user-pass-verify auth-pam.pl via-file
```

Conviene consultar el manual de OpenVPN 2.0.x, en concreto la directiva que hemos utilizado, “--auth-user-pass-verify”, ya que esta opción tiene más de un método para pasar el login y password que conviene comprender.

3.3.2.- Login y Password Mediante Plugins de Objetos Compartidos

Los objetos compartidos son normalmente módulos compilados escritos en código C que tienen que ser cargados en el servidor OpenVPN en tiempo de ejecución. Para ello, OpenVPN 2.0 proporciona un fichero escrito en lenguaje C llamado “auth-pam.c”, el cual trabaja con librerías y funciones del paquete PAM. Este fichero tiene que ser compilado para generar el fichero objeto que será cargado en OpenVPN utilizando, para

ello, la directiva de OpenVPN “--plugin”. Cuando se compila este fichero se genera el fichero objeto compartido llamado “openvpn-auth-pam.so”. Una vez generado dicho fichero se ha de añadir esta línea en el fichero de configuración del servidor OpenVPN:

```
plugin openvpn-auth-pam.so tipoServicio
```

Donde tipoServicio es un parámetro que corresponde a un servicio proporcionado por PAM ubicado, normalmente en distribuciones Linux, en el fichero /etc/pam.d. En la mayoría de las pruebas realizadas en este proyecto final de carrera se utilizará el servicio Login.

3.3.3.- Login y Password Sin Certificado de Cliente

Cuando utilizamos la autenticación por login y password en el servidor, por defecto, se utiliza una doble autenticación, ya que el cliente, además de pasar el login y password para autenticarse, pasará su certificado público también. Existe la posibilidad de implementar únicamente la autenticación del cliente mediante login y password utilizando la directiva “--client-cert-not-required”. Para ello se ha de introducir la siguiente línea en el fichero de configuración del servidor:

```
client-cert-not-required
```

Capítulo 4: Configuración de los Escenarios

4.1.- Breve Descripción del Laboratorio

Las pruebas para implementar los siguientes escenarios se han realizado en el laboratorio docente IT-2 de la UPCT con el material disponible en este laboratorio. Para la realización de las pruebas que se incluyen en este capítulo se ha contado con el siguiente material en el laboratorio:

- Tres PCs con dos particiones, una con sistema operativo SuSE 8.2 con todos los paquetes necesarios instalados (vistos en esta memoria) y otra con Windows XP. Estos PCs disponen de dos tarjetas de red 10/100 ya que podrán hacer a su vez la función de routing.
- Un portátil con dos particiones, una para Windows XP con SP2 (Service Pack 2) y una distribución de Linux, Kubuntu 7.04 Feisty. En ambos sistemas operativos se ha instalado todo lo necesario para poder realizar pruebas tanto con Windows como con Linux.
- 3 Switch Micronet EtherFast 10/100M.
- Latiguillos RJ-45 Cat.5e para el conexionado.

Los Switch utilizados aprenden solas las direcciones IP (su configuración se podría decir que es automática), por tanto se pasará directamente a explicar los escenarios y los ficheros de configuración utilizados.



Figura 79. Switch Utilizado para las Pruebas en el Laboratorio

4.2.- Túnel Sencillo Seguro LAN to LAN

Una configuración muy típica dentro de las tecnologías VPN consiste, muy resumidamente, en conectar dos o más sucursales, delegaciones u oficinas, que están separadas por la red de redes (Internet), a través de un túnel seguro, permitiendo a las redes internas de cada oficina transmitir datos seguros, al resto de oficinas, a través de dicho túnel. De esta manera, una máquina situada dentro de la red de una oficina

aparenta que está en la red de otra oficina que puede estar separada cientos o miles de kilómetros utilizando para ello Internet y OpenVPN.

Para simular este escenario hemos disminuido la complejidad de lo que podrían ser las redes de las oficinas, ya que esto supondría incluir una instalación de cableado estructurado y no es representativo para este proyecto fin de carrera.

Por otro lado, no se ha utilizado ninguna conexión a Internet, por lo que las redes de las oficinas que hemos implementado (oficina A y oficina B) estarán separadas por una red aparte implementada mediante un switch que simulará Internet. En una situación real habría que utilizar dos conexiones a Internet y habría que tener en cuenta si las direcciones que nos proporciona el ISP son fijas o dinámicas, ya que habría que actuar de una forma u otra en los ficheros de configuración. En este proyecto se han supuesto IPs fijas. Además en una situación real habría que tener en cuenta los modems o routers de entrada a la red, aunque no serían problema ya que la mayoría de estos routers domésticos o avanzados hacen de NAT y, como ya se ha dicho, OpenVPN soporta NAT sin ningún problema. En esta configuración se supone que los servidores hacen la función de routing, ya que disponen de dos tarjetas de red para ello. Además, para no confundir, utilizaremos direcciones IP privadas para las redes de las oficinas y direcciones IP no privadas para la red denominada como Internet.

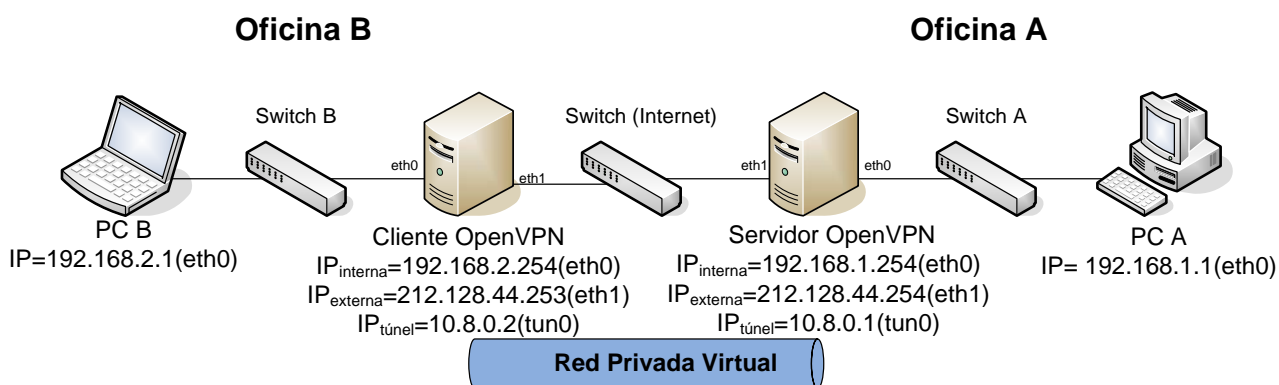


Figura 80. Esquema del Escenario 1

En este escenario las máquinas que hacen de extremos del túnel (cliente y servidor OpenVPN) tendrán sistemas operativos SuSE 8.2 ya que, así, OpenVPN podrá repartir las direcciones IP de los extremos del túnel de uno en uno (como sucede en los protocolos punto a punto). Como se explicó en el apartado 2.4.3, Windows no puede implementar túneles punto a punto reales (una IP por cada punto), por lo que los implementa mediante subredes /30 (4 direcciones IP por cada túnel en lugar de 2). Además existen otras opciones de OpenVPN que solo están disponibles para máquinas no-Windows. Por otro lado, también hay que mencionar que los PCs de las oficinas (PC A y PC B) pueden implementar cualquier sistema operativo, ya que en estos PCs no hace falta utilizar OpenVPN.

A continuación vamos a describir la configuración del sistema operativo donde se encuentra ubicado el servidor OpenVPN, como por ejemplo, la asignación de las IPs, incluir alguna ruta, etc.

Para ello, primero paramos el proceso `rcnetwork` de todas las máquinas y asignamos las direcciones IP a cada interfaz de red de las máquinas.

Todas las máquinas:

```
rcnetwork stop
```

Servidor OpenVPN:

```
ifconfig eth0 192.168.1.254 up  
ifconfig eth1 212.128.44.254 up
```

Cliente OpenVPN:

```
ifconfig eth0 192.168.2.254 up  
ifconfig eth1 212.128.44.253 up
```

PC A:

```
ifconfig eth0 192.168.1.1 up
```

PC B:

```
ifconfig eth0 192.168.2.1 up
```

También se tiene que habilitar el IP-forwarding para que las máquinas que incorporan OpenVPN (las que hacen de extremos del túnel y routing) puedan retransmitir los paquetes que le llegan de una interfaz hacia la otra interfaz de red. Además tenemos que cargar los módulos TUN/TAP en el sistema operativo.

En el Cliente OpenVPN y Servidor OpenVPN

```
echo "1" > /proc/sys/net/ipv4/ip_forward  
modprobe tun
```

En este punto, solo tenemos comunicación entre las dos máquinas que incluyen OpenVPN (cliente y servidor OpenVPN) y entre máquinas de la misma oficina. Para que los PCs que hay dentro de la red de las oficinas (PC A y PC B) y las máquinas OpenVPN puedan comunicarse con la red de la otra oficina se ha de incluir una nueva entrada en la tabla de rutas del servidor OpenVPN y del cliente OpenVPN.

Servidor OpenVPN:

```
route add -net 192.168.2.0 netmask 255.255.255.0 gw 10.8.0.2
```

Cliente OpenVPN:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 10.8.0.1
```

4.2.1.- Seguridad Implementada

En este escenario se ha implementado el modelo de seguridad mediante certificados, es decir, utilizando una PKI y el protocolo TLS para establecer un canal seguro y poder generar las claves necesarias. Para ello, utilizaremos los scripts descritos en el apartado 3.2 de esta memoria. Además es muy importante añadir la directiva “--tls-client” en el fichero de configuración de la máquina Cliente OpenVPN y la directiva “--tls-server” en el fichero de configuración del Servidor OpenVPN.

Primero se tendrá que crear la Autoridad Certificadora (CA), que generará dos ficheros: ca.crt y ca.key. El fichero ca.key podemos guardarlo en algún dispositivo seguro que no tenga conexión (por seguridad) pero el fichero ca.crt tenemos que tenerlo tanto en el Cliente OpenVPN como en el Servidor OpenVPN. Como se comentó en el apartado 3.2 tendremos que utilizar algún canal seguro que proporcione algún protocolo o aplicación para transferir el fichero ca.crt a ambas máquinas (Cliente y Servidor OpenVPN). Como ejemplo, se propone utilizar WinSCP o alguna aplicación similar.

Servidor o Cliente OpenVPN (luego se ha de transferir a la otra máquina):

```
. ./vars  
./clean-all  
./build-ca
```

Tras crear la CA se tienen que crear, en la misma máquina donde se ha creado la CA, el certificado público y la clave privada del Servidor OpenVPN y transferirla en caso de no haberla creado en la máquina Servidor OpenVPN mediante un canal seguro. En este escenario se han creado una clave privada y certificado llamados servidor.key y servidor.cert.

```
. ./vars  
./build-key-server servidor
```

Después hacemos el mismo procedimiento para crear el par certificado/clave privada del Cliente OpenVPN. En este caso llamaremos cliente.key y cliente.crt a los ficheros. Como se ha realizado en el apartado anterior estos ficheros tienen que ser transferidos al Cliente OpenVPN mediante un canal seguro en el caso de no haber sido creados en dicha máquina.

```
. ./vars
./build-key cliente
```

Para acabar de implementar la seguridad de este escenario es necesario generar los parámetros Diffie Hellman. En este caso, el fichero generado, dh1024.pem (o dh2048.pem, según el tamaño de la clave para RSA) solo debe estar ubicado en la máquina Servidor OpenVPN, por lo que habría que transferirlo sobre un canal seguro únicamente en el caso de generar este fichero en otra máquina que no sea el Servidor OpenVPN.

```
. ./vars
./build-dh
```

4.2.2.- Archivos de Configuración

En este apartado estudiaremos cada línea incluida en los archivos de configuración del Servidor OpenVPN, llamado “servidor.conf”, y del Cliente OpenVPN, llamado “cliente.conf”. Como ya se ha comentado en este capítulo este escenario se ha implementado utilizando máquinas Linux, por lo que la extensión de los ficheros de configuración es “.conf”.

El fichero de configuración del Servidor OpenVPN (“servidor.conf”) es el siguiente:

```
local 212.128.44.254
port 1194
dev tun
proto udp
ifconfig 10.8.0.1 10.8.0.2
tls-server
dh dh1024.pem
ca ca.crt
cert servidor.crt
key servidor.key
comp-lzo
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Servidor OpenVPN (servidor.conf):

- **local 212.128.44.254:** vinculamos OpenVPN a la interfaz con la dirección IP 212.128.44.254.
- **port 1194:** utilizamos el puerto 1194.

- **dev tun:** indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **ifconfig 10.8.0.1 10.8.0.2:** indicamos que el Servidor OpenVPN va a tener la dirección IP en su interfaz “tun” 10.8.0.1 y que, en el otro extremo del túnel, el Cliente OpenVPN tendrá la dirección IP en su interfaz “tun” 10.8.0.2.
- **tls-server:** indica que esta máquina actuará como servidor durante el establecimiento del protocolo TLS.
- **dh dh1024.pem:** cargamos los parámetros Diffie Hellman.
- **ca ca.crt:** cargamos el certificado público de la CA.
- **cert servidor.crt:** cargamos el certificado del Servidor OpenVPN.
- **key servidor.key:** cargamos la clave privada del Servidor OpenVPN.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **keepalive 10 120:** se puede traducir en que el Servidor OpenVPN enviará un ping cada 10 segundos y esperará 120 segundos máximo para recibir contestación, sino deducirá que el otro extremo (el Cliente OpenVPN) ha caído.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Servidor OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Servidor OpenVPN.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.
- **verb 6:** indica el grado de detalle de información de estado del túnel.

El fichero de configuración del Cliente OpenVPN (“cliente.conf”) es el siguiente:

```
dev tun
proto udp
remote 212.128.44.254 1194
ifconfig 10.8.0.2 10.8.0.1
tls-client
nobind
ca ca.crt
cert cliente.crt
key cliente.key
comp-lzo
resolv-retry infinite
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
```

verb 6

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Cliente OpenVPN (cliente.conf).

- **dev tun:** indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **remote 212.128.44.254 1194:** indica a que máquina se tiene que conectar para establecer el túnel y utilizando el puerto 1194 asignado por la IANA.
- **ifconfig 10.8.0.2 10.8.0.1:** indicamos que el Cliente OpenVPN va a tener la dirección IP en su interfaz “tun” 10.8.0.2 y que, en el otro extremo del túnel, el Servidor OpenVPN tendrá la dirección IP en su interfaz “tun” 10.8.0.1.
- **tls-client:** indica que esta máquina actuará como cliente durante el establecimiento del protocolo TLS.
- **nobind:** hacemos que OpenVPN no se vincule a la IP local y puerto de esta máquina. Esta directiva se utiliza cuando utilizamos la directiva “remote”.
- **ca ca.crt:** carga el certificado de la CA.
- **cert cliente.crt:** carga el certificado del Cliente OpenVPN.
- **key cliente.key:** carga la clave privada del Cliente OpenVPN.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **resolv-retry infinite:** el cliente intentará de manera indefinida resolver la dirección o nombre de host dado por la directiva “remote”.
- **keepalive 10 120:** se puede traducir en que el Cliente OpenVPN enviará un ping cada 10 segundos y esperará 120 segundos máximo para recibir contestación, sino deducirá que el otro extremo (el Servidor OpenVPN) ha caído.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Cliente OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Cliente OpenVPN.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.
- **verb 6:** indica el grado de detalle de información de estado del túnel.

4.2.3.- Resultados y Conclusiones

Tras tener el montaje de este primer escenario listo se ha comprobado, para empezar, que existe comunicación entre ambos extremos de la VPN (Cliente OpenVPN y Servidor OpenVPN) y que también existe comunicación entre los PCs de ambas oficinas (PC A y PC B), por lo que se debe haber introducido las rutas descritas en el apartado 4.2 tanto en el Servidor OpenVPN como en el Cliente OpenVPN. Una vez introducidas dichas rutas la tabla de rutas de cada máquina quedaría de la siguiente manera (recordar que se utiliza el comando “route -n” en máquinas Linux y el comando “route print” en máquinas Windows para poder ver la tabla de rutas):

Servidor OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.2	0.0.0.0	255.255.255.255	tun0
192.168.1.0	0.0.0.0	255.255.255.0	eth0
212.128.44.0	0.0.0.0	255.255.255.0	eth1
192.168.2.0	10.8.0.2	255.255.255.0	tun0
0.0.0.0	212.128.44.253	0.0.0.0	eth1

Cliente OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.1	0.0.0.0	255.255.255.255	tun0
192.168.2.0	0.0.0.0	255.255.255.0	eth0
212.128.44.0	0.0.0.0	255.255.255.0	eth1
192.168.1.0	10.8.0.1	255.255.255.0	tun0
0.0.0.0	212.128.44.254	0.0.0.0	eth1

PC A (Oficina A)			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
192.168.1.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.1.254	0.0.0.0	eth0

PC B (Oficina B)			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
192.168.2.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.2.254	0.0.0.0	eth0

Como se puede ver en las tablas de encaminamiento y se ha podido comprobar en el laboratorio mediante el comando “ping”, existe comunicación entre ambos extremos del túnel (Cliente OpenVPN y Servidor OpenVPN), de manera que la información que sale desde una de estas máquinas y llega hasta la otra máquina va cifrada.

Por otra parte, también existe comunicación entre las máquinas de ambas oficinas (PC A y PC B) de manera que la información que va desde la salida de la oficina A hasta la entrada de la red de la oficina B va por el túnel, por lo que va a ir cifrada de manera segura. Hay que tener en cuenta que la información que va desde el PC A (o PC B en el otro caso) hasta el Servidor OpenVPN (o Cliente OpenVPN en el otro caso) no va cifrada, por lo que se debería recurrir a otra solución para poder tener esa información cifrada y segura.

El funcionamiento es sencillo. Por ejemplo, si el PC A envía un “ping” al PC B, el PC A genera un paquete con la dirección IP destino 192.168.2.1 e IP origen 192.168.1.1. Como esta IP destino (192.168.2.1) no se encuentra en la tabla de encaminamiento del PC A, entonces utiliza la entrada por defecto (destino 0.0.0.0), por tanto, envía dicho paquete a la dirección IP 192.168.1.254, perteneciente al Servidor OpenVPN. Cuando el Servidor OpenVPN mira la IP destino de este paquete y ve que es la 192.168.2.1 mira en la tabla de encaminamiento y ve que tiene que retransmitirla por la interfaz del túnel OpenVPN y comprueba que la puerta de enlace es la 10.8.0.2 (dirección IP del Cliente OpenVPN). Pero antes de ser retransmitido, el paquete original con IP origen 192.168.1.1 e IP destino 192.168.2.1 es empaquetado dentro de un paquete OpenVPN, que utiliza el protocolo de transporte UDP y que lleva como IP origen 212.128.44.254 e IP destino 212.128.44.253. Una vez el paquete es empaquetado como se ha dicho anteriormente, se envía desde el Servidor OpenVPN al Cliente OpenVPN y una vez que le llega a este último, éste lo desempaqueta y obtiene el paquete original, cuya IP origen era la 192.168.1.1 y cuya IP destino era la 192.168.2.1. El Cliente OpenVPN, solo debe mirar la IP destino (192.168.2.1) y consultando en su tabla de encaminamiento comprobará que dicho paquete va para la red de su oficina, por lo que lo transmitiría al PC B, el cual es el destino. Para responder a dicho ping habría que seguir los mismos pasos pero con la IP origen 192.168.2.1 e IP destino 192.168.1.1.

De esta manera, si suponemos que la red donde se encuentra el switch intermedio es Internet, tendríamos la información bien asegurada y cifrada. Así, una oficina que tuviera un servidor exclusivo en su red corporativa podría ser utilizado por una oficina de la misma empresa que estuviera separada por Internet (y por varios kilómetros de distancia) y hacerlo de manera que su información no pueda ser vista.

4.3.- Servidor de Túneles Seguro para Configuración Roadwarrior

La configuración Roadwarrior (o teletrabajador) es también muy utilizada ya que permite a clientes de OpenVPN conectarse al servidor OpenVPN utilizando para ello cualquier conexión a Internet que tengan disponible. Por lo tanto, un usuario podría establecer un túnel seguro con el servidor de su oficina desde la conexión a Internet de un ciber café, un aeropuerto, etc.

En este escenario, al estar limitados en material, también hemos hecho algunas suposiciones como en el escenario anterior. Para empezar, se ha reducido la complejidad de la red de la oficina, como en el caso anterior, implementando su red mediante un switch, el PC A y el Servidor OpenVPN (que haría de puerta de enlace y, por tanto, haría la función de routing). Igual que en el caso anterior se utilizará un switch para simular que tenemos una red por medio, es decir, Internet.

Los comandos utilizados para fijar las IPs de cada uno de las máquinas siguen la misma estructura que en el escenario anterior (ver el principio del apartado 4.2). Los módulos TUN/TAP deben ser cargados tanto en el Servidor OpenVPN como en cada uno de los Clientes OpenVPN tal como se hizo en el escenario anterior. Además hay que habilitar el ip-forwarding.

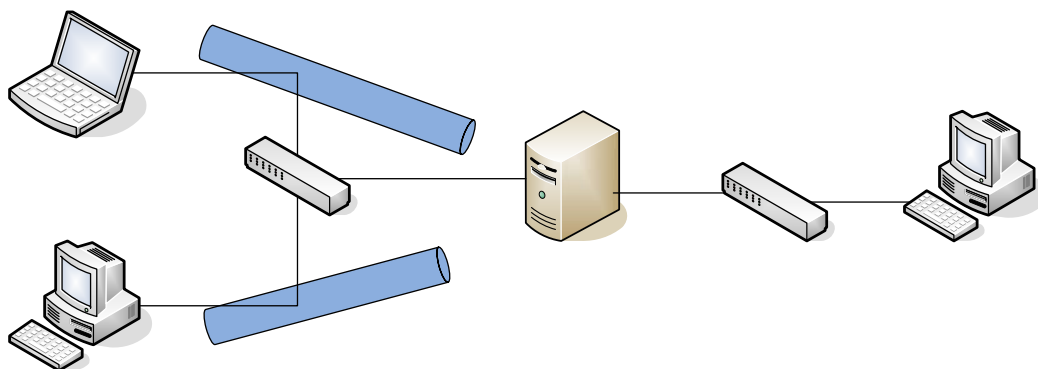


Figura 81. Esquema del Escenario 2

Los PCs que actúan como Cliente OpenVPN (Cliente OpenVPN 1 y Cliente OpenVPN 2) y el PC A, situado en la oficina A, tendrán instalados sistemas operativos Windows XP, ya que en un caso real, suele ser el sistema operativo más utilizado a nivel de usuario. El Servidor OpenVPN tendrá instalado un sistema operativo Linux SuSE 8.2, ya que es la mejor opción al tener que implementar un servidor.

La arquitectura de este escenario va a ser utilizada en los escenarios del capítulo siguiente, ya que en los ficheros de configuración utilizados para el servidor se utiliza la directiva “--server network netmask” que, ha diferencia de la configuración del escenario anterior, simplifica la configuración del Servidor OpenVPN y permite implementar una gran cantidad de túneles utilizando para ello una sola interfaz TUN/TAP en el Servidor OpenVPN y un solo puerto. Por ejemplo, la línea “server 10.8.0.0 255.255.255.0” equivale a las siguientes líneas:

```
mode server
tls-server
```



```
if dev tun:
    ifconfig 10.8.0.1 10.8.0.2
    ifconfig-pool 10.8.0.4 10.8.0.251
    route 10.8.0.0 255.255.255.0
    if client-to-client:
        push "route 10.8.0.0 255.255.255.0"
    else:
        push "route 10.8.0.1"

if dev tap:
    ifconfig 10.8.0.1 255.255.255.0
    ifconfig-pool 10.8.0.2 10.8.0.254 255.255.255.0
    push "route-gateway 10.8.0.1"
```

A continuación se describirá, brevemente, lo que indica cada línea:

- **mode server:** indica que ésta máquina va a actuar como servidor. La directiva “mode” tiene como parámetro por defecto “p2p” (“mode p2p”), que es el modo utilizado en el escenario anterior. Gracias a la opción “mode server” podemos, mediante una única interfaz “tun” en el servidor, implementar más de un túnel para poder conectar más de un cliente OpenVPN.
- **tls-server:** le indicamos a OpenVPN que en el protocolo TLS esta máquina actuará como servidor.

Las siguientes líneas que vamos a dedicar se refieren al caso de utilizar el modo Tunnel de OpenVPN, es decir, que se utilizarán dispositivos “tun” en lugar de dispositivos “tap”, por lo que se comentará solo el bloque de líneas de la primera sentencia “if”:

- **ifconfig 10.8.0.1 10.8.0.2:** indicamos al servidor que su interfaz “tun” tendrá la IP 10.8.0.1 y el extremo opuesto del túnel tendrá la IP 10.8.0.2. El extremo opuesto es ficticio, pero se le ha de dar una IP.
- **ifconfig-pool 10.8.0.4 10.8.0.251:** indica que el Servidor OpenVPN va a dar a los clientes que se vayan conectando direcciones desde la 10.8.0.4 hasta la 10.8.0.251.
- **route 10.8.0.0 255.255.255.0:** añade una ruta a la interfaz “tun” que se ha implementado en el Servidor OpenVPN.

Al tener máquinas Windows tenemos el problema de tener que utilizar subredes /30 para implementar los túneles, por lo que el servidor utiliza las cuatro primeras IPs para implementar el túnel:

- 10.8.0.0: Dirección de Red y de Subred
- 10.8.0.1: Dirección IP del túnel en el extremo del Servidor OpenVPN. Es la dirección IP que se le da a la interfaz “tun” del Servidor OpenVPN.
- 10.8.0.2: es la dirección IP asignada al otro extremo del túnel que ha implementado el Servidor OpenVPN, pero no es asignada a ningún cliente.
- 10.8.0.3: es la dirección de broadcast de subred.

La línea “ifconfig-pool 10.8.0.4 10.8.0.251”, indica que el Servidor OpenVPN va a dar a los clientes que se vayan conectando direcciones desde la 10.8.0.4 hasta la 10.8.0.251. Pero puede ser engañoso ya que hay que tener en cuenta que las subredes tienen direcciones de subred y de broadcast que no podrán ser asignadas a los clientes, por tanto, las direcciones que van a ser asignadas se muestran a continuación:

```
[ 1, 2] [ 5, 6] [ 9, 10] [ 13, 14] [ 17, 18]
[ 21, 22] [ 25, 26] [ 29, 30] [ 33, 34] [ 37, 38]
[ 41, 42] [ 45, 46] [ 49, 50] [ 53, 54] [ 57, 58]
[ 61, 62] [ 65, 66] [ 69, 70] [ 73, 74] [ 77, 78]
[ 81, 82] [ 85, 86] [ 89, 90] [ 93, 94] [ 97, 98]
[101,102] [105,106] [109,110] [113,114] [117,118]
[121,122] [125,126] [129,130] [133,134] [137,138]
[141,142] [145,146] [149,150] [153,154] [157,158]
[161,162] [165,166] [169,170] [173,174] [177,178]
[181,182] [185,186] [189,190] [193,194] [197,198]
[201,202] [205,206] [209,210] [213,214] [217,218]
[221,222] [225,226] [229,230] [233,234] [237,238]
[241,242] [245,246] [249,250] [253,254]
```

El primer cliente tendrá asignada a su interfaz “tun” la IP 10.8.0.6 y como extremo contrario del túnel la IP 10.8.0.5, que es una IP que no se utiliza, pero sirve para representar el otro extremo del túnel. De la misma manera, el segundo cliente en conectarse tendrá la IP 10.8.0.10 en su interfaz “tun”, el tercero la IP 10.8.0.14 y el cuarto cliente tendrá la IP 10.8.0.18.

Tras la línea con la directiva “route” hay una sentencia “if”, en pseudocódigo, que comprueba si la directiva “cliente-to-client” está incluida en el fichero de configuración o no. La directiva “client-to-client” está pensada para que los clientes OpenVPN que se conecten al servidor OpenVPN puedan comunicarse entre ellos añadiendo alguna ruta en sus tablas, como se explicará a continuación.

Si la directiva “client-to-client” está habilitada en el fichero de configuración del Servidor OpenVPN, entonces se ejecuta:

```
push route 10.8.0.0 255.255.255.0
```

Esta línea utiliza la directiva “push”, que es muy útil en OpenVPN para que los clientes tengan en cuenta algunas directivas de manera remota, sin necesidad de tocar el fichero de configuración de ningún cliente. Para ello, el cliente tiene que tener habilitada la directiva “pull”. Por ejemplo, en la línea que estamos viendo, el Servidor OpenVPN le dice a los Clientes OpenVPN que añadan a la tabla de rutas de la interfaz “tun” la red 10.8.0.0 con máscara de red 255.255.255.0. De esta manera, los clientes podrán comunicarse dentro de la red 10.8.0.0, es decir, que podrán comunicarse entre ellos.

Si la directiva “client-to-client” no está incluida en el fichero de configuración del Servidor OpenVPN, entonces se ejecutaría la sentencia “else” del código anteriormente visto. Por tanto, habría que tener en cuenta la línea:

```
push "route 10.8.0.1"
```

Esta línea les indica a los clientes que introduzcan en la tabla de rutas la manera de llegar al Servidor OpenVPN (siempre que los clientes incluyan la directiva “pull”). Por tanto, los clientes que se conecten a la VPN no podrán comunicarse con el resto de clientes, únicamente con el Servidor OpenVPN.

4.3.1.- Seguridad Implementada

La seguridad implementada en este escenario es muy similar a la implementada en el escenario anterior, un modelo de seguridad basado en certificados, utilizando una infraestructura de clave pública (PKI) y el protocolo TLS para establecer un canal seguro para el intercambio y generación de claves. Pero esta vez hay que tener en cuenta que ahora la máquina Servidor OpenVPN tiene el modo server habilitado (directiva “--server” de OpenVPN), por lo que los clientes se irán conectando a esta máquina conforme quieran acceder a la VPN. Es decir, habrá que crear un certificado público y una clave privada para cada cliente que desee conectarse o bien utilizar el mismo certificado público y clave privada para todos los clientes (no recomendable).

Como se hizo en la parte de seguridad del escenario anterior, utilizaremos los scripts descritos en el apartado 3.2 de esta memoria para implementar las claves y certificados. En este caso no es necesario incluir las directivas “--tls-client” y “--tls-server” en los ficheros de configuración de los Clientes y del Servidor OpenVPN respectivamente, ya que se utilizan las directivas “--client” y “--server”, respectivamente, que incluyen estas directivas, como ya se ha visto anteriormente con la directiva “--server”.

Primero se tendrá que crear la Autoridad Certificadora (CA), que generará dos ficheros: ca.crt y ca.key. El fichero ca.key podemos guardarlo en algún dispositivo seguro que no tenga conexión (por seguridad) pero el fichero ca.crt tenemos que tenerlo tanto en el Cliente OpenVPN como en el Servidor OpenVPN. Igual que en el escenario anterior, tendremos que utilizar WinSCP, por ejemplo, para transferir los certificados, claves privadas de los clientes y el certificado y clave privada de la CA mediante un canal seguro desde el Servidor OpenVPN (donde suponemos que se generarán) a cada uno de los Clientes OpenVPN que se conecte.

En el Servidor OpenVPN (posteriormente el fichero ca.crt se ha de transferir desde el Servidor OpenVPN a todos los Clientes OpenVPN):

```
. ./vars  
. /clean-all  
. /build-ca
```

Tras crear la CA se tienen que crear el certificado público y la clave privada del Servidor OpenVPN. En este escenario se han creado una clave privada y certificado llamados `servidor.key` y `servidor.cert`.

```
. ./vars
./build-key-server servidor
```

Después realizamos el mismo procedimiento para crear los pares certificado/clave privada de los Clientes OpenVPN que vamos a implementar. En este proyecto se conectaron dos Clientes OpenVPN: Cliente OpenVPN 1 y Cliente OpenVPN 2. Por tanto, se generarán dos pares certificado/clave y se deberán transferir mediante un canal seguro o de manera segura para que no puedan ser interceptados.

```
. ./vars
./build-key cliente1
./build-key cliente2
```

Para acabar de implementar la seguridad de este escenario es necesario generar los parámetros Diffie Hellman. En este caso, el fichero generado, `dh1024.pem` (o `dh2048.pem`, según el tamaño de la clave para RSA) solo debe estar ubicado en la máquina Servidor OpenVPN, por lo que habría que transferirlo sobre un canal seguro únicamente en el caso de generar este fichero en otra máquina que no sea el Servidor OpenVPN.

```
. ./vars
./build-dh
```

4.3.2.- Archivos de Configuración

En este apartado estudiaremos cada línea incluida en los archivos de configuración del Servidor OpenVPN, llamado “`servidor.conf`”, y de los Clientes OpenVPN, llamados “`cliente.ovpn`” (no es necesario que tengan nombres diferentes ya que se ejecutan en máquinas diferentes). Como ya se ha comentado en este capítulo el Servidor OpenVPN tiene instalado un SuSE 8.2, por lo que la extensión del fichero de configuración será “.conf”. Sin embargo, los Clientes OpenVPN tienen instalados un sistema operativo Windows XP, por lo que la extensión de los ficheros de configuración será “.ovpn”.

El fichero de configuración del Servidor OpenVPN (“`servidor.conf`”) es el siguiente:

```
local 212.128.44.254
proto udp
dev tun
ca ca.crt
cert servidor.crt
key servidor.key
dh dh1024.pem
server 10.8.0.0 255.255.255.0
```

```
push "route 192.168.1.0 255.255.255.0"  
ifconfig-pool-persist ipp.txt  
client-to-client (opcional)  
duplicate-cn (opcional)  
keepalive 10 120  
comp-lzo  
persist-key  
persist-tun  
status openvpn-status.log  
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Servidor OpenVPN (“servidor.conf”):

- **local 212.128.44.254:** vinculamos OpenVPN a la interfaz con la dirección IP 212.128.44.254.
- **port 1194:** utilizamos el puerto 1194.
- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **dev tun:** indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **ca ca.crt:** cargamos el certificado público de la CA.
- **cert servidor.crt:** cargamos el certificado del Servidor OpenVPN.
- **key servidor.key:** cargamos la clave privada del Servidor OpenVPN.
- **dh dh1024.pem:** cargamos los parámetros Diffie Hellman.
- **server 10.8.0.0 255.255.255.0:** indica que esta máquina va a actuar como un servidor y asignará las direcciones IP de la VPN a los clientes que se vayan conectando. Esta directiva está explicada más detalladamente al principio del apartado 4.3.
- **push "route 192.168.1.0 255.255.255.0":** añadimos una ruta en la tabla de rutas de los Clientes OpenVPN. Mediante esta línea indicamos al Cliente OpenVPN que envíe los paquetes que tengan como destino la red de la oficina (192.168.1.0) por la interfaz del túnel, por lo que los Clientes OpenVPN podrán alcanzar la red de la oficina utilizando un túnel seguro por Internet.
- **ifconfig-pool-persist ipp.txt:** mantiene un fichero llamado “ipp.txt” en el que se encuentran los clientes que se conectan al Servidor OpenVPN incluyendo la dirección IP que se le ha asignado y el Common Name del certificado que han entregado al Servidor OpenVPN.
- **client-to-client:** esta línea es opcional y sirve, como se ha dicho al principio del apartado 4.3, para que los clientes puedan comunicarse entre ellos.
- **duplicate-cn:** esta línea es opcional y sirve para poder tener certificados y claves privadas duplicados en varios clientes a la vez. De esta manera solo

habría que crear un par certificado/clave privada para todos los clientes. Es una opción muy poco recomendable.

- **keepalive 10 120:** se puede traducir en que el Servidor OpenVPN enviará un ping cada 10 segundos y esperará 120 segundos máximo para recibir contestación, sino deducirá que el otro extremo ha caído.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Servidor OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Servidor OpenVPN.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.
- **verb 6:** indica el grado de detalle de información de estado del túnel.

El fichero de configuración de cada Cliente OpenVPN (“cliente.ovpn”) tiene la siguiente forma:

```
client
dev tun
proto udp
remote 212.128.44.254 1194
nobind
resolv-retry infinite
persist-key
persist-tun
ca ca.crt
cert clienteX.crt (X = 1 ó 2)
key clienteX.key (X = 1 ó 2)
comp-lzo
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Cliente OpenVPN (“cliente.ovpn”).

- **client:** indicamos a OpenVPN que esta máquina actuará como cliente. Más adelante, en este mismo apartado, se explica a que equivale este comando.
- **dev tun:** indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **remote 212.128.44.254 1194:** indica a que máquina se tiene que conectar para establecer el túnel y utilizando el puerto 1194 asignado por la IANA.

- **nobind:** hacemos que OpenVPN no se vincule a la IP local y puerto de esta máquina. Esta directiva se utiliza cuando utilizamos la directiva “remote”.
- **resolv-retry infinite:** el cliente intentará de manera indefinida resolver la dirección o nombre de host dado por la directiva “remote”.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Cliente OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Cliente OpenVPN.
- **ca ca.crt:** carga el certificado de la CA.
- **cert clienteX.crt:** carga el certificado del Cliente OpenVPN.
- **key clienteX.key:** carga la clave privada del Cliente OpenVPN.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.
- **verb 6:** indica el grado de detalle de información de estado del túnel.

Como se puede apreciar en el fichero de configuración de los Clientes OpenVPN, se utiliza la directiva “client”. Esta directiva, igual que la directiva “server”, utilizada en el fichero de configuración del Servidor OpenVPN, tiene una equivalencia en cuanto a comandos OpenVPN se refiere. La directiva “client” es igual a poner las siguientes dos líneas de directivas OpenVPN:

```
pull
tls-client
```

Cada una de estas directivas tiene una funcionalidad, que se explica a continuación:

- **pull:** indica al cliente de OpenVPN que tenga en cuenta las directivas que el servidor de OpenVPN ha enviado mediante la directiva “push”.
- **tls-client:** indica que esta máquina actuará como cliente durante el establecimiento del protocolo TLS.

4.3.3.- Resultados y Conclusiones

Tras tener el montaje de este segundo escenario preparado se ha comprobado, para comenzar, que existe comunicación entre Clientes OpenVPN y el Servidor OpenVPN. Si se quiere que haya comunicación entre los Clientes OpenVPN hay que habilitar la directiva “--client-to-client”. Además hay que probar si existe comunicación entre los Clientes OpenVPN implementados y el PC situado en la oficina (PC A). Habrá que comprobar, por tanto, la tabla de rutas de cada máquina implementada. La tabla de rutas de cada máquina quedaría de la siguiente manera (recordar que se utiliza el comando “route -n” en máquinas Linux y el comando “route print” en máquinas Windows para poder ver la tabla de rutas):

Servidor OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.2	0.0.0.0	255.255.255.255	tun0
10.8.0.0	10.8.0.2	255.255.255.0	tun0
192.168.1.0	0.0.0.0	255.255.255.0	eth0
212.128.44.0	0.0.0.0	255.255.255.0	eth1

Cliente OpenVPN 1			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.5	0.0.0.0	255.255.255.255	tun0
10.8.0.1	10.8.0.5	255.255.255.255	tun0
192.168.1.0	10.8.0.5	255.255.255.0	tun0
212.128.44.0	0.0.0.0	255.255.255.0	eth0

Cliente OpenVPN 2			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.9	0.0.0.0	255.255.255.255	tun0
10.8.0.1	10.8.0.9	255.255.255.255	tun0
192.168.1.0	10.8.0.9	255.255.255.0	tun0
212.128.44.0	0.0.0.0	255.255.255.0	eth0

Como se puede apreciar en las tablas de encaminamiento existen algunas IP que, aunque no son asignadas a ninguna interfaz virtual se utilizan para tareas de encaminamiento. Como sabemos, del apartado 2.4.3, cuando utilizamos máquinas Windows, no podemos establecer túneles punto a punto reales (con 2 direcciones IP por túnel, la de un extremo y la del otro extremo). Por tanto, OpenVPN implementa, solo en este caso, los túneles mediante subredes con mascara de red /30 (4 direcciones IP por cada extremo del túnel, de las cuales solo una IP es asignada a la interfaz de red virtual).

Hay que destacar que la entrada de la tabla de rutas de los Clientes OpenVPN en la que se encuentra la dirección 192.168.1.0 ha sido añadida gracias a la directiva "--push" incluida en el fichero de configuración del Servidor OpenVPN. De esta manera le decimos a los Clientes OpenVPN que para enviar paquetes a la red de la oficina, cuya IP es la 192.168.1.0, utilicen la interfaz virtual del túnel, por lo que dicha información pasaría por Internet (en un caso real) de manera segura y cifrada a través del túnel.

Si además, se desea que cada Cliente OpenVPN pueda comunicarse con cualquier otro Cliente OpenVPN habrá que incluir la directiva "--client-to-client", mediante la cual obtendríamos una tabla de encaminamiento con la segunda entrada diferente. En este caso los Clientes OpenVPN no solo saben llegar al Servidor OpenVPN, cuya IP virtual

es la 10.8.0.1, sino que también saben llegar a cualquier Cliente OpenVPN, ya que se ha introducido la red 10.8.0.0 en la tabla de encaminamiento.

Cliente OpenVPN 1 (utilizando la directiva "--client-to-client")			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.5	0.0.0.0	255.255.255.255	tun0
10.8.0.0	10.8.0.5	255.255.255.0	tun0
192.168.1.0	10.8.0.5	255.255.255.0	tun0
212.128.44.0	0.0.0.0	255.255.255.0	eth0

Cliente OpenVPN 2 (utilizando la directiva "--client-to-client")			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.9	0.0.0.0	255.255.255.255	tun0
10.8.0.0	10.8.0.9	255.255.255.0	tun0
192.168.1.0	10.8.0.9	255.255.255.0	tun0
212.128.44.0	0.0.0.0	255.255.255.0	eth0

Si, por ejemplo, el Cliente OpenVPN 1, quiere comunicarse mediante el comando "ping" con el PC A que está situado dentro de la red de la oficina (red 192.168.1.0), este tendrá que enviar un paquete con la IP origen 10.8.0.6 e IP destino 192.168.1.1, el cual es empaquetado dentro de un paquete OpenVPN mediante el protocolo de transporte UDP y con direcciones IP origen 212.128.44.1 e IP destino 212.128.44.254. De esta manera el Cliente OpenVPN 1 envía los paquetes al Servidor OpenVPN. Una vez la información llega al Servidor OpenVPN, este desempaqueta el paquete originalmente enviado y comprueba que la dirección destino es la 192.168.1.1, por lo que inmediatamente se lo transmite al PC A. La respuesta del PC A a este ping es muy similar. El PC A envía paquetes con la IP origen 192.168.1.1 e IP destino 10.8.0.6 al Servidor OpenVPN. Cuando este recibe dichos paquetes los empaqueta en paquetes UDP cuya IP origen es la 212.128.44.254 e IP destino es la 212.128.44.1 y envía dichos paquetes al Cliente OpenVPN 1 para Internet (en un caso real) es la 212.128.44.1. Una vez llegan los paquetes a esta máquina, los desempaqueta y obtiene el paquete original cuya IP origen era la 192.168.1.1 e IP destino era la 10.8.0.6.

De esta manera, en un caso real, si la red de la oficina tuviera un servidor FTP, por ejemplo, y un Cliente OpenVPN se conecta desde la red WiFi de un aeropuerto, éste va a poder utilizar dicho servidor FTP como si se encontrara físicamente en dicha oficina y, además, sus datos serán enviados de manera segura y cifrada.

Capítulo 5: Pruebas en Escenarios Reales

5.1.- Breve Descripción de los Escenarios

Las pruebas para implementar los siguientes escenarios son muy similares a las realizadas en los escenarios del capítulo anterior. En general, contamos casi con el mismo material para realizarlas, pero esta vez contamos con conexiones a Internet para tener en cuenta que tenemos la red de redes (Internet) en medio. Otro punto a tener en cuenta es que el Servidor OpenVPN, en ambos escenarios, ahora solo utiliza una tarjeta de red, ya que es lo más normal a nivel empresarial. Además hay que tener en cuenta la traducción de direcciones NAT, ya que los servidores, en una red empresarial, pueden utilizar direcciones IP privadas, sin embargo el cliente que quiera conectarse a dichos servidores necesitará hacerlo mediante una IP pública, ya que en otro caso, no podrá atravesar Internet y llegar a su destino. Por tanto, en los dos escenarios habrá un servidor que actúa como puerta de enlace y, además, realiza las funciones de NAT, para poder traducir las direcciones IP públicas en direcciones IP privadas.

Para configurar las direcciones IP de las diferentes máquinas, activar los módulos de las interfaces TUN/TAP, activar el ip-forwarding, etc, se utilizará el mismo procedimiento que el que se ha seguido para el apartado 4.2, aunque hay que tener en cuenta que en este caso el Cliente OpenVPN lleva instalado un sistema operativo Windows XP.

5.2.- Servidor de Túneles Seguro con Acceso Mediante Login/Password

En el primer escenario se han utilizado las IP públicas proporcionadas por los laboratorios docentes de la UPCT: laboratorio IT-2 (212.128.44.45) y laboratorio IT-5 (212.128.44.34). El servidor del laboratorio IT-2 actuará como puerta de enlace (normalmente lo hace un router comercial o un modem proporcionado por la compañía que nos proporciona el servicio a Internet) y también hace las funciones de NAT. Por lo tanto, cuando el cliente OpenVPN quiera conectarse al servidor OpenVPN, cuya IP es 192.168.2.98 (IP privada), lo hará mediante la IP del servidor del laboratorio IT-2, cuya IP externa es la IP pública 212.128.44.45, y lo hará por el puerto 1194 de OpenVPN. Cuando el servidor IT-2, compruebe que la IP destino de los paquetes enviados por el cliente OpenVPN es la 212.128.44.45 y el puerto destino es el 1194, traducirá esta IP a la IP destino 192.168.2.98, es decir, la del servidor OpenVPN. De esta manera, el cliente OpenVPN podrá alcanzar al servidor OpenVPN cruzando Internet.

A su vez, la máquina que hace de Cliente OpenVPN tiene la IP pública 212.128.44.34 (aunque lo normal es que la IP pública, hacia Internet, la tenga un modem o router proporcionado por la compañía que ofrece el servicio a Internet).

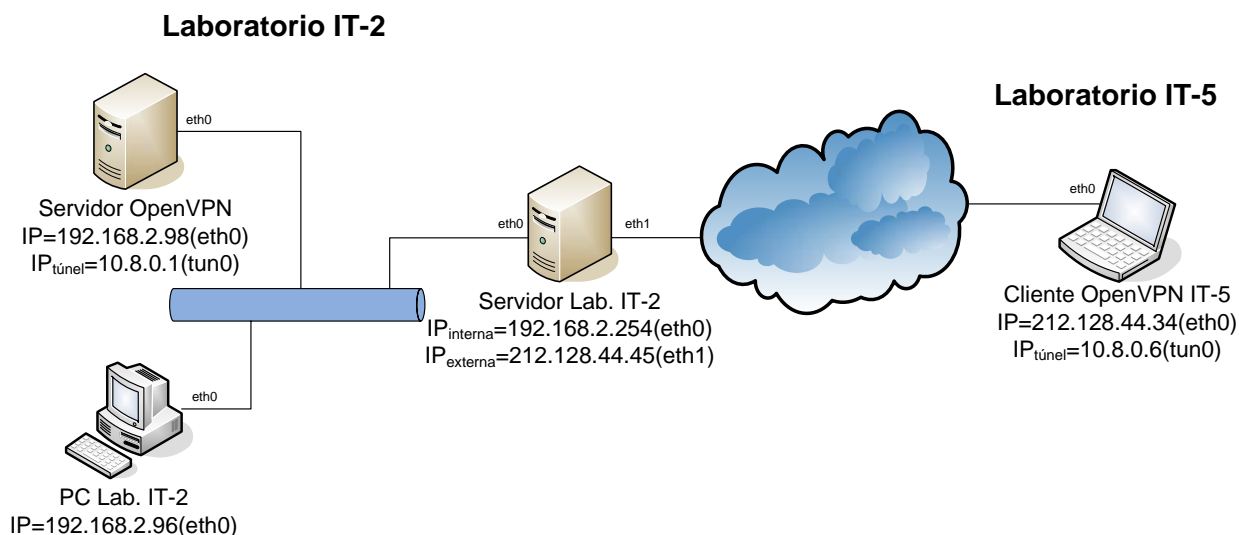


Figura 82. Esquema del Escenario Real 1

En este escenario, al estar limitados en material, también hemos hecho algunas suposiciones como en el escenario anterior. Para empezar, se ha reducido la complejidad de la red de la oficina, como en el caso anterior, implementando su red mediante un switch, un PC del laboratorio IT-2, el Servidor OpenVPN y el Servidor del laboratorio IT-2, que hará las funciones de puerta de enlace y de NAT.

Los comandos utilizados para fijar las IPs de cada uno de las máquinas siguen la misma estructura que en el escenario anterior (ver el principio del apartado 4.2). Los módulos TUN/TAP deben ser cargados tanto en el Servidor OpenVPN como en cada uno de los Clientes OpenVPN tal como se hizo en el escenario anterior. Además hay que habilitar el ip-forwarding.

No hay que olvidar que para que el PC del laboratorio IT-2, cuya IP es 192.168.2.96 pueda alcanzar al Cliente OpenVPN, cuya IP virtual es 10.8.0.6, tendrá que añadir una entrada más en su tabla de encaminamiento. De esta manera cuando el Cliente OpenVPN haga un “ping”, por ejemplo, a la dirección 192.168.2.96, cuando este “ping” le llegue, y vea que la dirección IP origen del paquete original es la 10.8.0.6, en lugar de entregarlo a la puerta de enlace por defecto (el Servidor del Laboratorio IT-2 con IP 192.168.2.254), la entregará al Servidor OpenVPN para que vuelva a empaquetarlo:

```
route add -net 10.8.0.0 netmask 255.255.255.0 gw 192.168.2.98
```

5.2.1.- Seguridad Implementada

La seguridad implementada en estos escenarios es muy similar a la implementada en los escenarios del capítulo 4: se ha seguido el modelo de seguridad mediante certificados, utilizando el protocolo TLS pero además se ha añadido un factor más de autenticación mediante login y password. Para dicha autenticación mediante login y password se ha

utilizado el plugin mediante objetos compartidos proporcionado por OpenVPN y se ha utilizado el servicio Login de PAM.

Cuando el Cliente OpenVPN desee conectarse al Servidor OpenVPN tendrá primero que pasarle el certificado del cliente para establecer un canal seguro mediante todo el proceso que TLS conlleva y, cuando esté preparado el canal seguro, pedirá un login y password que el Cliente OpenVPN tendrá que introducir con éxito si desea acceder a la VPN. Por tanto, la información que el Cliente OpenVPN le pasa al Servidor OpenVPN (el login y el password) no la puede interceptar nadie, ya que será transmitida sobre un canal seguro proporcionado por TLS.

Como se hizo en la parte de seguridad de los escenarios del capítulo, utilizaremos los scripts descritos en el apartado 3.2 de esta memoria para implementar las claves y certificados. En este caso no es necesario incluir las directivas “--tls-client” y “--tls-server” en los ficheros de configuración de los Clientes y del Servidor OpenVPN respectivamente, ya que se utilizan las directivas “--client” y “--server”.

Primero se tendrá que crear la Autoridad Certificadora (CA), que generará dos ficheros: ca.crt y ca.key. El fichero ca.key podemos guardarlo en algún dispositivo seguro que no tenga conexión (por seguridad) pero el fichero ca.crt tenemos que tenerlo tanto en el Cliente OpenVPN como en el Servidor OpenVPN. Igual que en el escenario anterior, tendremos que utilizar WinSCP, por ejemplo, para transferir los certificados, claves privadas de los clientes y el certificado y clave privada de la CA mediante un canal seguro desde el Servidor OpenVPN (donde suponemos que se generarán) a cada uno de los Clientes OpenVPN que se conecte.

Servidor OpenVPN (posteriormente el fichero ca.crt se ha de transferir desde el Servidor OpenVPN a todos los Clientes OpenVPN):

```
. ./vars
./clean-all
./build-ca
```

Tras crear la CA se tienen que crear el certificado público y la clave privada del Servidor OpenVPN. En este escenario se han creado una clave privada y certificado llamados servidor.key y servidor.cert.

```
. ./vars
./build-key-server servidor
```

Después realizamos el mismo procedimiento para crear el par certificado/clave privada del Cliente OpenVPN que vamos implementar. En este escenario solo se implementa un Cliente OpenVPN cuya dirección IP es la IP pública 212.128.44.34.

```
. ./vars
```

```
./build-key client1
```

Para acabar de implementar la seguridad de este escenario es necesario generar los parámetros Diffie Hellman. En este caso, el fichero generado, dh1024.pem (o dh2048.pem, según el tamaño de la clave para RSA) solo debe estar ubicado en la máquina Servidor OpenVPN, por lo que habría que transferirlo sobre un canal seguro únicamente en el caso de generar este fichero en otra máquina que no sea el Servidor OpenVPN.

```
./vars
./build-dh
```

En los escenarios de este capítulo, en particular, se ha implementado la autenticación mediante login y password, por lo que habrá que generar el fichero de tipo objeto compartido “openvpn-auth-pam.so”. Para ello, OpenVPN proporciona una carpeta llamada “/plugin/auth-pam” en el que proporciona los ficheros necesarios y un fichero “make” para ejecutar y generar el fichero “openvpn-auth-pam.so”. Para ello tendremos que tener las librerías del paquete PAM y PAM-devel previamente instaladas. Además, este paso, hay que destacar, que solo se realiza en el Servidor OpenVPN” y también hay que decir que el paquete PAM no es compatible con Windows, por lo que no podremos implementar esta característica en servidores con Windows. Por tanto, para generar el archivo “openvpn-auth-pam.so”, una vez situados en el directorio donde se ha ubicado OpenVPN, vamos al directorio plugin/auth-pam y simplemente introducimos el siguiente comando:

```
make
```

Para implementar la autenticación por login y password hay que incluir alguna directiva en el fichero de configuración del Servidor y del Cliente OpenVPN. En el fichero de configuración del Servidor OpenVPN habrá que añadir la directiva “--plugin”, para indicar a OpenVPN que se va a utilizar el plugin para autenticarse por login y password por PAM y para indicarle que módulo tiene que cargar y el servicio PAM que va a utilizar. Por otro lado, en el fichero del Cliente OpenVPN habrá que añadir la directiva “--auth-user-pass” para indicar a OpenVPN que va a tener que pedir el login y password del cliente.

En esta implementación de login y password el servidor mantendrá una serie de usuarios en el sistema operativo, que tendrá un login y un password. Para poder acceder a la VPN deberán de introducir su login y password siempre que se haya creado en el Servidor OpenVPN un usuario. Existe un plugin para equipar OpenVPN con autenticación mediante login y password utilizando LDAP, pero no ha sido implementado en este proyecto fin de carrera, en el que se plantearon como objetivos la utilización e implementación básicos de OpenVPN.

5.2.2.- Archivos de Configuración

En este apartado estudiaremos cada línea incluida en los archivos de configuración del Servidor OpenVPN, llamado “servidor.conf”, y del Cliente OpenVPN, llamado “cliente.ovpn”. Como ya se ha comentado en este capítulo el Servidor OpenVPN tiene instalado un SuSE 8.2, por lo que la extensión del fichero de configuración será “.conf”. Sin embargo, los Clientes OpenVPN tienen instalados un sistema operativo Windows XP, por lo que la extensión de los ficheros de configuración será “.ovpn”.

El fichero de configuración del servidor (“servidor.conf”) es el que se muestra a continuación:

```
local 192.168.2.98
port 1194
proto udp
dev tun
plugin openvpn-auth-pam.so login
ca ca.crt
cert servidor.crt
key servidor.key
dh dh1024.pem
server 10.8.0.0 255.255.255.0
push "route 192.168.2.0 255.255.255.0"
ifconfig-pool-persist ipp.txt
client-to-client
duplicate-cn
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Servidor OpenVPN (“servidor.conf”):

- **local 192.168.2.98**: vinculamos OpenVPN a la interfaz con la dirección IP 192.168.2.98.
- **port 1194**: utilizamos el puerto 1194.
- **proto udp**: se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **dev tun**: indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **plugin openvpn-auth-pam.so login**
- **ca ca.crt**: cargamos el certificado público de la CA.
- **cert servidor.crt**: cargamos el certificado del Servidor OpenVPN.

- **key servidor.key:** cargamos la clave privada del Servidor OpenVPN.
- **dh dh1024.pem:** cargamos los parámetros Diffie Hellman.
- **server 10.8.0.0 255.255.255.0:** indica que esta máquina va a actuar como un servidor y asignará las direcciones IP de la VPN a los clientes que se vayan conectando. Esta directiva esta explicada más detalladamente al principio del apartado 4.3.
- **push "route 192.168.2.0 255.255.255.0":** añadimos una ruta en la tabla de rutas del Cliente OpenVPN. Mediante esta línea indicamos al Cliente OpenVPN que envíe los paquetes que tengan como destino la red del laboratorio IT-2 (192.168.2.0) por la interfaz del túnel, por lo que el Cliente OpenVPN podrán comunicarse con cualquier PC del laboratorio IT-2, además de poder comunicarse con el Servidor OpenVPN.
- **ifconfig-pool-persist ipp.txt:** mantiene un fichero llamado “ipp.txt” en el que se encuentran los clientes que se conectan al Servidor OpenVPN incluyendo la dirección IP que se le ha asignado y el Common Name del certificado que han entregado al Servidor OpenVPN.
- **client-to-client:** esta línea es opcional y sirve, como se ha dicho al principio del apartado 4.3, para que los clientes puedan comunicarse entre ellos (en el caso de haber más de un Cliente OpenVPN).
- **duplicate-cn:** esta línea es opcional y sirve para poder tener certificados y claves privadas duplicados en varios clientes a la vez. De esta manera solo habría que crear un par certificado/clave privada para todos los clientes. Es una opción muy poco recomendable.
- **keepalive 10 120:** se puede traducir en que el Servidor OpenVPN enviará un ping cada 10 segundos y esperará 120 segundos máximo para recibir contestación, sino deducirá que el otro extremo ha caído.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Servidor OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Servidor OpenVPN.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.
- **verb 6:** indica el grado de detalle de información de estado del túnel.

El fichero de configuración del Cliente OpenVPN (“cliente.ovpn”) es el que se muestra a continuación:

```
client
dev tun
proto udp
remote 212.128.44.45 1194
resolv-retry infinite
```

```
nobind
persist-key
persist-tun
auth-user-pass
ca ca.crt
cert cliente.crt
key cliente.key
comp-lzo
status openvpn-status.log
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Cliente OpenVPN (“cliente.ovpn”):

- **client:** indicamos a OpenVPN que esta máquina actuará como cliente. Esta directiva se ha explicado con más detalle al final del apartado 4.3.2.
- **dev tun:** indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **remote 212.128.44.45 1194:** indica a que máquina se tiene que conectar para establecer el túnel y utilizando el puerto 1194 asignado por la IANA.
- **resolv-retry infinite:** el cliente intentará de manera indefinida resolver la dirección o nombre de host dado por la directiva “remote”.
- **nobind:** hacemos que OpenVPN no se vincule a la IP local y puerto de esta máquina. Esta directiva se utiliza cuando utilizamos la directiva “remote”.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Cliente OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Cliente OpenVPN.
- **auth-user-pass:** indica a OpenVPN que esta máquina tendrá que introducir un login y password para poder acceder a la VPN.
- **ca ca.crt:** carga el certificado de la CA.
- **cert cliente.crt:** carga el certificado del Cliente OpenVPN.
- **key cliente.key:** carga la clave privada del Cliente OpenVPN.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.
- **verb 6:** indica el grado de detalle de información de estado del túnel.

5.2.3.- Resultados y Conclusiones

Tras haber realizado el montaje de este escenario se ha comprobado, para empezar, si se puede establecer la VPN de manera correcta, es decir, si la máquina que va a hacer de Cliente OpenVPN y la máquina que va a hacer de Servidor OpenVPN pueden comunicarse antes de arrancar los procesos de OpenVPN correspondientes a cada uno. Para ello, hay que tener en cuenta que se han introducido los parámetros necesarios en el Servidor del Laboratorio IT-2 para que pueda hacer de NAT. De esta manera, cuando el PC que va a hacer de Cliente OpenVPN quiera conectarse a la VPN, lo hará conectándose a la dirección IP 212.128.44.45 (perteneciente a la IP externa del Servidor de Laboratorio IT-2) por el puerto 1194. Cuando al Servidor del Laboratorio IT-2 le llegue tráfico por ese puerto traducirá la IP destino 212.128.44.45 a la IP destino 192.168.2.98, que es la IP de la máquina que hace de Servidor OpenVPN. Una vez haya comunicación entre estas dos máquinas, con IPs 212.128.44.34 y 192.168.2.98, se podrá establecer el túnel y por tanto también habrá comunicación entre Cliente OpenVPN y Servidor OpenVPN.

Una vez arrancados el Servidor OpenVPN y el Cliente OpenVPN se ha de tener en cuenta que se ha introducido la ruta en la tabla de encaminamiento del PC del Laboratorio IT-2, cuya IP es 192.168.2.98, para que este PC pueda contestar a peticiones del Cliente OpenVPN. Como se dijo anteriormente, al comienzo del apartado 5.2, para añadir esta ruta a la tabla de encaminamiento del PC del Laboratorio IT-2 hay que poner dicha línea en la consola de comandos de dicho PC:

```
route add -net 10.8.0.0 netmask 255.255.255.0 gw 192.168.2.98
```

Por tanto, la tabla de rutas de cada máquina quedaría de la siguiente manera (recordar que se utiliza el comando “route -n” en máquinas Linux y el comando “route print” en máquinas Windows para poder ver la tabla de rutas):

Servidor OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.2	0.0.0.0	255.255.255.255	tun0
10.8.0.0	10.8.0.2	255.255.255.0	tun0
192.168.2.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.2.254	255.255.255.0	eth0

Servidor Laboratorio IT-2			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
192.168.2.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	212.128.44.X (normalmente un nodo perteneciente a la WAN)	0.0.0.0	eth0

Cliente OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.5	0.0.0.0	255.255.255.255	tun0
10.8.0.1	10.8.0.5	255.255.255.255	tun0
192.168.2.0	10.8.0.5	255.255.255.0	tun0
0.0.0.0	212.128.44.X (normalmente un nodo perteneciente a la WAN)	255.255.255.0	eth0

PC Laboratorio IT-2			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.0	192.168.2.98	255.255.255.0	eth0
192.168.2.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.2.254	0.0.0.0	eth0

Cuando el Cliente OpenVPN quiere comunicarse con el PC del Laboratorio IT-2 con un “ping”, por ejemplo, el Cliente OpenVPN genera paquetes con IP origen 10.8.0.6 e IP destino 192.168.2.96. Tras generar paquetes con dichas IPs los empaqueta en datagramas UDP con direcciones IP origen 212.128.44.34 y destino 212.128.44.45 para poder atravesar Internet, y las envía por el puerto 1194. Una vez llegan al Servidor del Laboratorio IT-2, cuya IP pública es la 212.128.44.45, éste traduce dicha dirección pública en la dirección IP privada 192.168.2.98, que es la dirección IP del Servidor OpenVPN. Por tanto, el Servidor del Laboratorio IT-2 solo tendrá que reenviar dichos datagramas al Servidor OpenVPN. Una vez llegan estos datagramas al Servidor OpenVPN, dicho Servidor los desempaqueta y comprueba que la dirección IP destino es la 192.168.2.96, por lo que lo reenvía al PC del Laboratorio IT-2 sin ningún problema. El PC del laboratorio, para responder al “ping”, genera paquetes con la IP origen 192.168.2.96 y con la IP destino 10.8.0.6. Al añadir la ruta 10.8.0.0 a dicho PC, éste le enviará al Servidor OpenVPN. El Servidor OpenVPN una vez obtiene estos paquetes los empaqueta en datagramas UDP con dirección IP origen 192.168.2.98 y dirección IP destino 212.128.44.34, por lo que se lo envía al Servidor del Laboratorio IT-2 por el puerto 1194. El Servidor del Laboratorio IT-2 hace la traducción inversa a la que hizo con anterioridad, por lo que la IP origen del datagrama UDP es la 212.128.44.45 y la IP destino es la 212.128.44.34. Estos datagramas llegarán a través de Internet al Cliente OpenVPN, el cual los desempaquetará y verá que la IP destino es la suya, la 10.8.0.6.

Otro aspecto a tener en cuenta es el de la autenticación mediante login y password, la cual ha funcionado correctamente. Para ello, previamente se ha tenido que crear un usuario en el Servidor OpenVPN. Por ejemplo, si se crea un usuario cuyo nombre de usuario es “Cliente1” y cuya contraseña es “Cliente1pass”, cuando el Cliente OpenVPN intente conectarse a la VPN tendrá que mandar “Cliente1” como login y “Cliente1pass” como password cuando se lo pidan. También se ha podido comprobar que dicho login y password no van en texto plano, ya que, como se ha dicho en apartados anteriores,

cuando el Cliente OpenVPN introduce su login y su password, estos viajan sobre un canal seguro proporcionado por el protocolo TLS.

5.3.- Acceso a Portal Seguro con Acceso Login/Password

En este apartado se va a implementar una solución muy práctica para el ámbito empresarial. Consiste en implementar una VPN con OpenVPN con una arquitectura similar a la del escenario anterior (ver apartado 5.2), en la que los clientes que quieran acceder a la VPN tendrá que entregar sus certificados, como primer método para autenticarse, y escribir su login y su password como segundo método para autenticarse. Además, se instalará un servidor Web Apache en la red corporativa del Servidor OpenVPN (dicha red seguiremos implementandola de manera muy simple, con poco más que un switch). Este servidor Apache podrá ser accedido mediante la IP pública 212.128.44.15 y puerto 80 (el puerto reservado para el protocolo HTTP) gracias a la función de NAT que hace el servidor de entrada a la red corporativa, cuya IP externa es la IP pública 212.128.44.15 y cuya IP interna es la IP privada 192.168.100.254. Este servidor Apache también tendrá una sección que solo podrá ser navegada en el caso de estar conectado a la VPN, es decir, solo aceptará direcciones IP que pertenezcan a la red 10.8.0.0. Por tanto, de esta manera solo se permite el acceso a dicha Web a los usuarios que tengamos registrados en el Servidor OpenVPN, y además, estos usuarios navegarán de manera segura, ya que sus datos irán cifrados por el túnel que va desde los Clientes OpenVPN hasta el Servidor OpenVPN, cruzando Internet (que es una red no segura del todo).

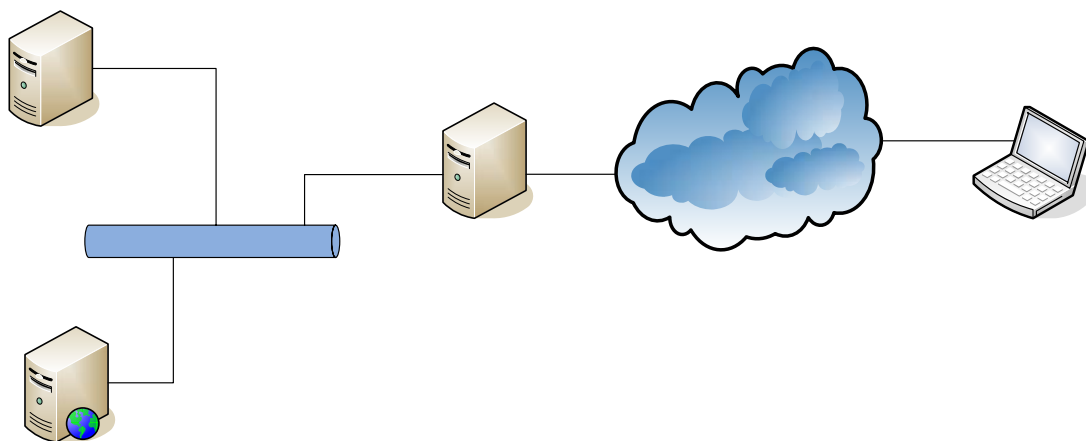


Figura 83. Esquema del Escenario Real 2

En este escenario también se han hecho algunas suposiciones como en el escenario anterior. Para empezar, se ha reducido la complejidad de la red de la oficina, como en el caso anterior, implementando su red mediante un switch, el servidor Apache, el Servidor OpenVPN y el Servidor que haría de puerta de enlace y las funciones de NAT.

Los comandos utilizados para fijar las IPs de cada uno de las máquinas siguen la misma estructura que en el escenario anterior (ver el principio del apartado 4.2). Los módulos TUN/TAP deben ser cargados tanto en el Servidor OpenVPN como en cada uno de los Clientes OpenVPN tal como se hizo en el escenario anterior. Además hay que habilitar el ip-forwarding.

5.3.1.- Seguridad Incorporada

La seguridad implementada en este escenario es la misma que la implementada en el escenario anterior (ver apartado 5.2.1). El modelo de seguridad escogido es mediante el uso de certificados y TLS. Además de utilizar certificados para la autenticación, se utiliza la autenticación mediante login y password (ver apartado 5.2.1 para más detalles).

5.3.2.- Configuración del Servidor Apache

El Servidor Apache ha sido configurado de manera que tendrá una Web pública, para cualquiera que realice una petición a la dirección IP pública 212.128.44.15 por el puerto 80, que es el puerto reservado para el protocolo HTTP, utilizando el protocolo TCP. Por otro lado, este mismo Servidor Apache tendrá otra Web que solo podrán acceder máquinas cuya dirección IP sea la de la VPN, por ejemplo una IP perteneciente a la red 10.8.0.0. De esta manera habría una Web en la empresa que solo podrá ser accedida por los Clientes OpenVPN que, previamente, han tenido que autenticarse presentando su certificado e introduciendo su login y su password.

5.3.3.- Archivos de Configuración

En este apartado estudiaremos cada línea incluida en los archivos de configuración del Servidor OpenVPN, llamado “servidor.conf”, y del Cliente OpenVPN, llamado “cliente.ovpn”. Como ya se ha comentado en este capítulo el Servidor OpenVPN tiene instalado un SuSE 8.2, por lo que la extensión del fichero de configuración será “.conf”. Sin embargo, los Clientes OpenVPN tienen instalados un sistema operativo Windows XP, por lo que la extensión de los ficheros de configuración será “.ovpn”.

El fichero de configuración del servidor (“servidor.conf”) es el que se muestra a continuación:

```
local 192.168.100.200
port 1194
proto udp
dev tun
plugin openvpn-auth-pam.so login
ca ca.crt
cert servidor.crt
key servidor.key
dh dh1024.pem
```

```
server 10.8.0.0 255.255.255.0
push "route 192.168.100.0 255.255.255.0"
ifconfig-pool-persist ipp.txt
client-to-client
duplicate-cn
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Servidor OpenVPN (“servidor.conf”):

- **local 192.168.100.200:** vinculamos OpenVPN a la interfaz con la dirección IP privada 192.168.100.200.
- **port 1194:** utilizamos el puerto 1194.
- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **dev tun:** indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.
- **plugin openvpn-auth-pam.so login**
- **ca ca.crt:** cargamos el certificado público de la CA.
- **cert servidor.crt:** cargamos el certificado del Servidor OpenVPN.
- **key servidor.key:** cargamos la clave privada del Servidor OpenVPN.
- **dh dh1024.pem:** cargamos los parámetros Diffie Hellman.
- **server 10.8.0.0 255.255.255.0:** indica que esta máquina va a actuar como un servidor y asignará las direcciones IP de la VPN a los clientes que se vayan conectando. Esta directiva está explicada más detalladamente al principio del apartado 4.3.
- **push "route 192.168.100.0 255.255.255.0":** añadimos una ruta en la tabla de rutas del Cliente OpenVPN. Mediante esta línea indicamos al Cliente OpenVPN que envíe los paquetes que tengan como destino la red en la que se encuentra el Servidor OpenVPN (192.168.100.0) por la interfaz del túnel, por lo que el Cliente OpenVPN podrán comunicarse con cualquier PC de la red corporativa donde se ubica el Servidor OpenVPN, además de poder comunicarse con el Servidor OpenVPN.
- **ifconfig-pool-persist ipp.txt:** mantiene un fichero llamado “ipp.txt” en el que se encuentran los clientes que se conectan al Servidor OpenVPN incluyendo la dirección IP que se le ha asignado y el Common Name del certificado que han entregado al Servidor OpenVPN.

- **client-to-client**: esta línea es opcional y sirve, como se ha dicho al principio del apartado 4.3, para que los clientes puedan comunicarse entre ellos (en el caso de haber más de un Cliente OpenVPN).
- **duplicate-cn**: esta línea es opcional y sirve para poder tener certificados y claves privadas duplicados en varios clientes a la vez. De esta manera solo habría que crear un par certificado/clave privada para todos los clientes. Es una opción muy poco recomendable.
- **keepalive 10 120**: se puede traducir en que el Servidor OpenVPN enviará un ping cada 10 segundos y esperará 120 segundos máximo para recibir contestación, sino deducirá que el otro extremo ha caído.
- **comp-lzo**: indica que se va a utilizar la librería de compresión LZO.
- **persist-key**: permite que las claves no tengan que ser re-leídas cuando el Servidor OpenVPN es reiniciado.
- **persist-tun**: permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Servidor OpenVPN.
- **status openvpn-status.log**: indica el fichero donde se volcará la información de estado del túnel.
- **verb 6**: indica el grado de detalle de información de estado del túnel.

El fichero de configuración del Cliente OpenVPN (“cliente.ovpn”) es el que se muestra a continuación:

```
client
dev tun
proto udp
remote 212.128.44.17 1194
resolv-retry infinite
nobind
persist-key
persist-tun
auth-user-pass
ca ca.crt
cert cliente.crt
key cliente.key
comp-lzo
status openvpn-status.log
verb 6
```

A continuación vamos a describir brevemente lo que hace cada línea del fichero de configuración del Cliente OpenVPN (“cliente.ovpn”):

- **client**: indicamos a OpenVPN que esta máquina actuará como cliente. Esta directiva se ha explicado con más detalle al final del apartado 4.3.2.
- **dev tun**: indica que vamos a implementar un túnel mediante un dispositivo “tun” de los drivers TUN/TAP.

- **proto udp:** se utilizará el protocolo UDP como protocolo de transporte del túnel.
- **remote 212.128.44.17 1194:** indica a que máquina se tiene que conectar para establecer el túnel y utilizando el puerto 1194 asignado por la IANA.
- **resolv-retry infinite:** el cliente intentará de manera indefinida resolver la dirección o nombre de host dado por la directiva “remote”.
- **nobind:** hacemos que OpenVPN no se vincule a la IP local y puerto de esta máquina. Esta directiva se utiliza cuando utilizamos la directiva “remote”.
- **persist-key:** permite que las claves no tengan que ser re-leídas cuando el Cliente OpenVPN es reiniciado.
- **persist-tun:** permite que el túnel no tenga que ser cerrado y re-abierto de nuevo tras reiniciar el Cliente OpenVPN.
- **auth-user-pass:** indica a OpenVPN que esta máquina tendrá que introducir un login y password para poder acceder a la VPN.
- **ca ca.crt:** carga el certificado de la CA.
- **cert cliente.crt:** carga el certificado del Cliente OpenVPN.
- **key cliente.key:** carga la clave privada del Cliente OpenVPN.
- **comp-lzo:** indica que se va a utilizar la librería de compresión LZO.
- **status openvpn-status.log:** indica el fichero donde se volcará la información de estado del túnel.

5.3.4.- Resultados y Conclusiones

Servidor OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.2	0.0.0.0	255.255.255.255	tun0
10.8.0.0	10.8.0.2	255.255.255.0	tun0
192.168.100.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.100.254	255.255.255.0	eth0

Servidor Laboratorio IT-2			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
192.168.100.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	212.128.44.X (normalmente un nodo perteneciente a la WAN)	0.0.0.0	eth0

Cliente OpenVPN			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.5	0.0.0.0	255.255.255.255	tun0
10.8.0.1	10.8.0.5	255.255.255.255	tun0
192.168.100.0	10.8.0.5	255.255.255.0	tun0
0.0.0.0	212.128.44.X (normalmente un nodo perteneciente a la WAN)	255.255.255.0	eth0

PC Laboratorio IT-2			
Destino	Puerta de Enlace	Mascara de Red	Interfaz
10.8.0.0	192.168.100.200	255.255.255.0	eth0
192.168.100.0	0.0.0.0	255.255.255.0	eth0
0.0.0.0	192.168.100.254	0.0.0.0	eth0

Cuando la máquina que va a hacer de Cliente OpenVPN desee acceder a la Web pública de nuestro Servidor Apache, solo tendrá que acceder a la dirección IP 212.128.44.15 por el puerto 80 utilizando TCP, sin arrancar el Cliente de OpenVPN, utilizando para ello un navegador. De esta manera, los datagramas TCP se enviarán por Internet con la IP origen 212.128.44.17 y con la IP destino 212.128.44.15. Cuando estos datagramas lleguen al Servidor del Laboratorio IT-2 este hará la traducción de la IP destino, de manera que ahora la IP destino pasa a ser la 192.168.100.247, que es la del Servidor Apache. De manera inversa, el Servidor Apache responde a dichas peticiones generadas por la máquina que hace de Cliente OpenVPN (con dicho cliente aun sin arrancar).

Para poder acceder a la Web privada el Cliente OpenVPN tiene que estar corriendo en la máquina. Una vez arrancado el Cliente OpenVPN y mediante un navegador Web, se introduce la dirección IP 192.168.100.254 (esta es la dirección IP privada del Servidor Web). Esta acción genera datagramas TCP por el puerto 80 cuya dirección IP origen es la 10.8.0.6 e IP destino es la 192.168.100.254, pero dichos datagramas son empaquetados en datagramas UDP, por el puerto 1194 (el de OpenVPN) y con dirección IP origen 212.128.44.17 e IP destino 212.128.44.15. De esta manera los paquetes UDP pueden llegar sin ningun problema al Servidor del Laboratorio IT-2 a través de Internet. Una vez en el Servidor del Laboratorio IT-2, este traduce la dirección IP destino 212.128.44.15 por la dirección 192.168.100.200. Una vez traducida la dirección IP destino, el Servidor del Laboratorio IT-2 envía los datagramas UDP al Servidor OpenVPN, y éste desempaqueta los datagramas UDP y obtiene los datagramas TCP originales, cuya IP origen es 10.8.0.6 e IP destino es 192.168.100.254. Por tanto, el Servidor OpenVPN, tras desempaquetar, envía la información al Servidor Apache, que responde enviando información de la Web privada. El proceso para responder al Cliente OpenVPN es el inverso al explicado para enviar las peticiones desde el Cliente OpenVPN.

Capítulo 6: Conclusiones y Líneas Futuras

6.1.- Conclusiones

Este trabajo surge ante el problema que plantea la falta de seguridad que presentan las comunicaciones a través de la infraestructura de redes públicas. En respuesta a ese problema, y debido a la gran importancia que ha adquirido el tráfico de información para las grandes organizaciones y empresas, aparecen las redes privadas virtuales. Elegimos la tecnología para implementar VPNs basada en el protocolo SSL/TLS, para estudiar una estructura concreta capaz de construir redes privadas virtuales. Entre las distintas implementaciones de esta tecnología, optamos por OpenVPN, por ser una herramienta completa y flexible. OpenVPN vuelve innecesaria la adquisición de hardware dedicado, resultando en una solución más económica, simple, flexible y escalable. Ante la necesidad de implementar una VPN siempre debería considerarse seriamente su utilización.

A lo largo de este documento, hemos detallado los conceptos teóricos y algunos casos prácticos de las VPN SSL/TLS, basadas en OpenVPN. En la primera parte, comenzamos por aclarar el concepto de VPN, describir las tecnologías utilizadas actualmente y hacer incapié en la tecnología SSL/TLS para la implementación de estas VPN. En el segundo capítulo se introdujo en OpenVPN en un entorno teórico y se mencionaron muchas de las características más importantes y paquetes complementarios de los que depende OpenVPN. En el tercer capítulo se pasó a la práctica realizando la instalación de OpenVPN y haciendo los preparativos para poder implementar las VPN. En el capítulo 4 se implementaron dos escenarios muy utilizados en los laboratorios docentes de la UPCT: el modelo para unir delegaciones u oficinas y el modelo para conectar clientes Roadwarrior a un servidor de túneles situado en una oficina o delegación. En el capítulo 5 se cogió el modelo de escenario con clientes Roadwarrior para implementarlo en un escenario real, con Internet de por medio, conexiones a Internet en ambos extremos de los túneles, teniendo en cuenta el NAT, utilizando IPs públicas y utilizando un protocolo no seguro como HTTP para utilizarlo de manera segura gracias a OpenVPN.

La tecnología SSL/TLS constituye sin duda alguna una opción flexible y robusta de asegurar las comunicaciones a través de la infraestructura de redes públicas. Esperamos que este documento permita comprender cómo funciona, y cómo se configura, sin demasiada dificultad, una conexión VPN SSL/TLS mediante OpenVPN.

6.2.- Líneas Futuras

En este proyecto se han abordado los aspectos más básicos de OpenVPN relacionados con su configuración, implementación de la seguridad y su puesta en marcha. Para ello se han probado algunos escenarios muy utilizados en el ámbito empresarial.

Por otro lado, un aspecto que se podría tener en cuenta para posibles estudios futuros es incrementar la funcionalidad de la autenticación por login y password. Para ello, existe un plugin para OpenVPN llamado “auth-ldap” para poder implementar la autenticación de los clientes mediante login y password via LDAP, utilizando para ello el paquete OpenLDAP.

Otra línea por la que se podría indagar es en la utilización de la versión beta 2.1 de OpenVPN, que ya puede ser descargada de su Web. Esta versión incorpora algunas características nuevas como la posibilidad de no tener que utilizar subredes /30 para implementar conexiones punto a punto cuando se utiliza alguna máquina Windows. También ofrece la posibilidad de compartir el puerto OpenVPN con otras aplicaciones utilizando TCP, por ejemplo, con HTTPS.

También sería interesante el desarrollo de alguna interfaz gráfica de usuario (GUI) que, además de ofrecer alguna manera más guiada y amigable de generar los archivos de configuración, ofreciera también la posibilidad de generar los certificados, claves privadas, CA y otros aspectos relacionados con la seguridad.

Anexos

Apéndice I: GUIs y otros Programas

OpenVPN GUI:

<http://openvpn.se/>

KVpnc:

<http://home.gna.org/kvpnc/en/index.html>

Tunnelblick:

<http://www.tunnelblick.net/>

OpenVPN-Admin:

<http://sourceforge.net/projects/openvpnadmin/>

OpenVPN-Control:

http://sourceforge.net/project/showfiles.php?group_id=152302

Webmin OpenVPN Admin Module:

http://www.openit.it/index.php/openit_en/soluzioni_gpl/openvpnadmin/

OpenVPN User Manager:

<http://www.mertech.com.au/mertech-products-openvpnusermanager.aspx>

My Certificate Wizard:

<http://mycert.sandbox.cz/>

TinyCA2:

<http://tinyca.sm-zone.net/>

XCA:

<http://sourceforge.net/projects/xca>

Apéndice II: Archivos de Configuración

En este apéndice se mostrarán los ficheros de configuración empleados en los escenarios implementados en este proyecto final de carrera, y algunos modelos proporcionados por el paquete OpenVPN 2.0.9 muy útiles.

El fichero de configuración del servidor implementado en el apartado 4.2, “servidor.conf”, se muestra a continuación:

```
local 212.128.44.254
port 1194
dev tun
proto udp
ifconfig 10.8.0.1 10.8.0.2
tls-server
dh dh1024.pem
ca ca.crt
cert servidor.crt
key servidor.key
comp-lzo
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
verb 6
```

El fichero de configuración del cliente implementado en el apartado 4.2, “cliente.conf”, se muestra a continuación:

```
dev tun
proto udp
remote 212.128.44.254 1194
ifconfig 10.8.0.2 10.8.0.1
tls-client
nobind
ca ca.crt
cert cliente.crt
key cliente.key
comp-lzo
resolv-retry infinite
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
verb 6
```

El fichero de configuración del servidor implementado en el apartado 4.3, “servidor.conf”, se muestra a continuación:

```
local 212.128.44.254
proto udp
dev tun
ca ca.crt
cert servidor.crt
key servidor.key
dh dh1024.pem
```

```
server 10.8.0.0 255.255.255.0
push "route 192.168.1.0 255.255.255.0"
ifconfig-pool-persist ipp.txt
client-to-client (opcional)
duplicate-cn (opcional)
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 6
```

El fichero de configuración del cliente implementado en el apartado 4.3, “cliente.ovpn”, se muestra a continuación:

```
client
dev tun
proto udp
remote 212.128.44.254 1194
nobind
resolv-retry infinite
persist-key
persist-tun
ca ca.crt
cert clienteX.crt (donde X = 1 ó 2)
key clienteX.key (donde X = 1 ó 2)
comp-lzo
verb 6
```

El fichero de configuración del servidor implementado en el apartado 5.2, “servidor.conf”, se muestra a continuación:

```
local 192.168.2.98
port 1194
proto udp
dev tun
plugin openvpn-auth-pam.so login
ca ca.crt
cert servidor.crt
key servidor.key
dh dh1024.pem
server 10.8.0.0 255.255.255.0
push "route 192.168.2.0 255.255.255.0"
ifconfig-pool-persist ipp.txt
client-to-client
duplicate-cn
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 6
```

El fichero de configuración del cliente implementado en el apartado 5.2, “cliente.ovpn”, se muestra a continuación:

```
client
dev tun
proto udp
```

```
remote 212.128.44.45 1194
resolv-retry infinite
nobind
persist-key
persist-tun
auth-user-pass
ca ca.crt
cert cliente.crt
key cliente.key
comp-lzo
status openvpn-status.log
verb 6
```

El fichero de configuración del servidor implementado en el apartado 5.3, “servidor.conf”, se muestra a continuación:

```
local 192.168.100.200
port 1194
proto udp
dev tun
plugin openvpn-auth-pam.so login
ca ca.crt
cert servidor.crt
key servidor.key
dh dh1024.pem
server 10.8.0.0 255.255.255.0
push "route 192.168.100.0 255.255.255.0"
ifconfig-pool-persist ipp.txt
client-to-client
duplicate-cn
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 6
```

El fichero de configuración del cliente implementado en el apartado 5.3, “cliente.ovpn”, se muestra a continuación:

```
client
dev tun
proto udp
remote 212.128.44.17 1194
resolv-retry infinite
nobind
persist-key
persist-tun
auth-user-pass
ca ca.crt
cert cliente.crt
key cliente.key
comp-lzo
status openvpn-status.log
verb 6
```

Fichero de configuración “server.conf” proporcionado por OpenVPN 2.0.9, muy útil para comenzar a configurar el servidor, suponiendo que se va a utilizar TLS como modelo de seguridad.

```
#####
# Sample OpenVPN 2.0 config file for
# multi-client server.
#
# This file is for the server side
# of a many-clients <-> one-server
# OpenVPN configuration.
#
# OpenVPN also supports
# single-machine <-> single-machine
# configurations (See the Examples page
# on the web site for more info).
#
# This config should work on Windows
# or Linux/BSD systems. Remember on
# Windows to quote pathnames and use
# double backslashes, e.g.:
# "C:\\Program Files\\OpenVPN\\config\\foo.key"
#
# Comments are preceded with '#' or ';'
#####

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one. You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap0" if you are ethernet bridging
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
```

```
# from the Network Connections panel if you
# have more than one.  On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt
cert server.crt
key server.key # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
#  openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh dh1024.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface.  Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0.  Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients.  Leave this line commented
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100
```



```
# Push routes to the client to allow it
# to reach other private subnets behind
# the server. Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
# iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN. This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
# ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients. There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
# group, and firewall the TUN/TAP interface
# for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
# modify the firewall in response to access
# from different clients. See man
# page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# the TUN/TAP interface to the internet in
# order for this to work properly).
# CAVEAT: May break client's network config if
# client's local DHCP server packets get routed
# through the tunnel. Solution: make sure
```

```
# client's local DHCP server is reachable via
# a more specific route than the default route
# of 0.0.0.0/0.0.0.0.
;push "redirect-gateway"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses. CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
;push "dhcp-option DNS 10.8.0.1"
;push "dhcp-option WINS 10.8.0.1"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names. This is recommended
# only for testing purposes. For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC # Blowfish (default)
;cipher AES-128-CBC # AES
;cipher DES-EDE3-CBC # Triple-DES
```

```
# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nobody

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "%Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it. Use one
# or the other (but not both).
;log          openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages. At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

Fichero de configuración “client.conf” proporcionado por OpenVPN 2.0.9, muy útil para comenzar a configurar los clientes, suponiendo que se va a utilizar TLS como modelo de seguridad. Este fichero de configuración corresponde a los clientes que se van a conectar al servidor cuyo fichero de configuración es el que se ha mostrado anteriormente (“server.conf”).

```
#####
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#                                             #
# This configuration can be used by multiple #
# clients, however each client should have  #
# its own cert and key files.               #
#                                             #
# On Windows, you might want to rename this #
# file so it has a .ovpn extension         #
#####

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one. On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap

# Are we connecting to a TCP or
# UDP server? Use the same setting as
# on the server.
;proto tcp
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote my-server-1 1194
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing. Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server. Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
```

```
;user nobody
;group nobody

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here. See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets. Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description. It's best to use
# a separate .crt/.key file pair
# for each client. A single ca
# file can be used for all clients.
ca ca.crt
cert client.crt
key client.key

# Verify server certificate by checking
# that the certificate has the nsCertType
# field set to "server". This is an
# important precaution to protect against
# a potential attack discussed here:
# http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server". The build-key-server
# script in the easy-rsa folder will do this.
;ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 3
```

```
# Silence repeating messages
;mute 20
```

Los dos ficheros de configuración que se mostrarán a continuación (“tls-office.conf” y “tls-home.conf”) son muy útiles para implementar el escenario en el que se conectan dos redes pertenecientes a oficinas, delegaciones, etc, separadas por Internet. Estos ficheros también son proporcionados por OpenVPN y son muy útiles para facilitar la realización de los ficheros de configuración de ambos extremos de la VPN (ambas delegaciones o una oficina y una casa como en este ejemplo).

El fichero “tls-office.conf” tiene el siguiente fichero de configuración:

```
#
# Sample OpenVPN configuration file for
# office using SSL/TLS mode and RSA certificates/keys.
#
# '#' or ';' may be used to delimit comments.

# Use a dynamic tun device.
# For Linux 2.2 or non-Linux OSes,
# you may want to use an explicit
# unit number such as "tun1".
# OpenVPN also supports virtual
# ethernet "tap" devices.
dev tun

# 10.1.0.1 is our local VPN endpoint (office).
# 10.1.0.2 is our remote VPN endpoint (home).
ifconfig 10.1.0.1 10.1.0.2

# Our up script will establish routes
# once the VPN is alive.
up ./office.up

# In SSL/TLS key exchange, Office will
# assume server role and Home
# will assume client role.
tls-server

# Diffie-Hellman Parameters (tls-server only)
dh dh1024.pem

# Certificate Authority file
ca my-ca.crt

# Our certificate/public key
cert office.crt

# Our private key
key office.key

# OpenVPN 2.0 uses UDP port 1194 by default
# (official port assignment by iana.org 11/04).
# OpenVPN 1.x uses UDP port 5000 by default.
# Each OpenVPN tunnel must use
```

```
# a different port number.
# lport or rport can be used
# to denote different ports
# for local and remote.
; port 1194

# Downgrade UID and GID to
# "nobody" after initialization
# for extra security.
; user nobody
; group nobody

# If you built OpenVPN with
# LZO compression, uncomment
# out the following line.
; comp-lzo

# Send a UDP ping to remote once
# every 15 seconds to keep
# stateful firewall connection
# alive. Uncomment this
# out if you are using a stateful
# firewall.
; ping 15

# Uncomment this section for a more reliable detection when a system
# loses its connection. For example, dial-ups or laptops that
# travel to other locations.
; ping 15
; ping-restart 45
; ping-timer-rem
; persist-tun
; persist-key

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3
```

Donde el fichero “office.up” es el que se muestra a continuación:

```
#!/bin/sh
route add -net 10.0.1.0 netmask 255.255.255.0 gw $5
```

El fichero “tls-home.conf” tiene el siguiente fichero de configuración:

```
#
# Sample OpenVPN configuration file for
# home using SSL/TLS mode and RSA certificates/keys.
#
# '#' or ';' may be used to delimit comments.

# Use a dynamic tun device.
# For Linux 2.2 or non-Linux OSes,
# you may want to use an explicit
# unit number such as "tun1".
```

```
# OpenVPN also supports virtual
# ethernet "tap" devices.
dev tun

# Our OpenVPN peer is the office gateway.
remote 1.2.3.4

# 10.1.0.2 is our local VPN endpoint (home).
# 10.1.0.1 is our remote VPN endpoint (office).
ifconfig 10.1.0.2 10.1.0.1

# Our up script will establish routes
# once the VPN is alive.
up ./home.up

# In SSL/TLS key exchange, Office will
# assume server role and Home
# will assume client role.
tls-client

# Certificate Authority file
ca my-ca.crt

# Our certificate/public key
cert home.crt

# Our private key
key home.key

# OpenVPN 2.0 uses UDP port 1194 by default
# (official port assignment by iana.org 11/04).
# OpenVPN 1.x uses UDP port 5000 by default.
# Each OpenVPN tunnel must use
# a different port number.
# lport or rport can be used
# to denote different ports
# for local and remote.
; port 1194

# Downgrade UID and GID to
# "nobody" after initialization
# for extra security.
; user nobody
; group nobody

# If you built OpenVPN with
# LZO compression, uncomment
# out the following line.
; comp-lzo

# Send a UDP ping to remote once
# every 15 seconds to keep
# stateful firewall connection
# alive. Uncomment this
# out if you are using a stateful
# firewall.
; ping 15

# Uncomment this section for a more reliable detection when a system
# loses its connection. For example, dial-ups or laptops that
# travel to other locations.
```



```
; ping 15
; ping-restart 45
; ping-timer-rem
; persist-tun
; persist-key

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3
```

Donde el fichero “home.up” es el que se muestra a continuación:

```
#!/bin/sh
route add -net 10.0.0.0 netmask 255.255.255.0 gw $5
```

OpenVPN también proporciona los mismos ficheros de configuración anteriores pero con el modelo de seguridad basado en claves pre-compartidas (static keys). Los ficheros se llaman “static-office.conf” y “static-home.conf”. Los ficheros “office.up” y “home.up” son los mismos que en el caso anterior.

El fichero “static-office.conf” tiene el siguiente fichero de configuración:

```
#
# Sample OpenVPN configuration file for
# office using a pre-shared static key.
#
# '#' or ';' may be used to delimit comments.

# Use a dynamic tun device.
# For Linux 2.2 or non-Linux OSes,
# you may want to use an explicit
# unit number such as "tun1".
# OpenVPN also supports virtual
# ethernet "tap" devices.
dev tun

# 10.1.0.1 is our local VPN endpoint (office).
# 10.1.0.2 is our remote VPN endpoint (home).
ifconfig 10.1.0.1 10.1.0.2

# Our up script will establish routes
# once the VPN is alive.
up ./office.up

# Our pre-shared static key
secret static.key

# OpenVPN 2.0 uses UDP port 1194 by default
# (official port assignment by iana.org 11/04).
# OpenVPN 1.x uses UDP port 5000 by default.
# Each OpenVPN tunnel must use
```

```
# a different port number.
# lport or rport can be used
# to denote different ports
# for local and remote.
; port 1194

# Downgrade UID and GID to
# "nobody" after initialization
# for extra security.
; user nobody
; group nobody

# If you built OpenVPN with
# LZO compression, uncomment
# out the following line.
; comp-lzo

# Send a UDP ping to remote once
# every 15 seconds to keep
# stateful firewall connection
# alive. Uncomment this
# out if you are using a stateful
# firewall.
; ping 15

# Uncomment this section for a more reliable detection when a system
# loses its connection. For example, dial-ups or laptops that
# travel to other locations.
; ping 15
; ping-restart 45
; ping-timer-rem
; persist-tun
; persist-key

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3
```

El fichero "static-home.conf" tiene el siguiente fichero de configuración:

```
#
# Sample OpenVPN configuration file for
# home using a pre-shared static key.
#
# '#' or ';' may be used to delimit comments.

# Use a dynamic tun device.
# For Linux 2.2 or non-Linux OSes,
# you may want to use an explicit
# unit number such as "tun1".
# OpenVPN also supports virtual
# ethernet "tap" devices.
dev tun

# Our OpenVPN peer is the office gateway.
```

```
remote 1.2.3.4

# 10.1.0.2 is our local VPN endpoint (home).
# 10.1.0.1 is our remote VPN endpoint (office).
ifconfig 10.1.0.2 10.1.0.1

# Our up script will establish routes
# once the VPN is alive.
up ./home.up

# Our pre-shared static key
secret static.key

# OpenVPN 2.0 uses UDP port 1194 by default
# (official port assignment by iana.org 11/04).
# OpenVPN 1.x uses UDP port 5000 by default.
# Each OpenVPN tunnel must use
# a different port number.
# lport or rport can be used
# to denote different ports
# for local and remote.
; port 1194

# Downgrade UID and GID to
# "nobody" after initialization
# for extra security.
; user nobody
; group nobody

# If you built OpenVPN with
# LZO compression, uncomment
# out the following line.
; comp-lzo

# Send a UDP ping to remote once
# every 15 seconds to keep
# stateful firewall connection
# alive. Uncomment this
# out if you are using a stateful
# firewall.
; ping 15

# Uncomment this section for a more reliable detection when a system
# loses its connection. For example, dial-ups or laptops that
# travel to other locations.
; ping 15
; ping-restart 45
; ping-timer-rem
; persist-tun
; persist-key

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3
```

Bibliografía

“*OpenVPN. Building and Integrating Virtual Private Networks*”, Markus Feliner.

“*Network Security with OpenSSL*”, John Viega, Matt Messier & Pravir Chandra.

“*VPNs Illustrated: Tunnels, VPNs and IPsec*”, Jon C. Snader

“*Internetworking with TCP/IP. Principles, protocols and architectures*”. Douglas E. Comer.

“*Apuntes de Laboratorio de Arquitectura de Redes de Computadores de 3º de Ingeniería Técnica de Telecomunicación, Especialidad Telemática de la UPCT*”, Jose Juan Sanchez Manzanares

“*Apuntes de Seguridad en Redes de 5º de Ingeniería de Telecomunicación de la UPCT*”, Maria Dolores Cano Baños

Referencias

<http://openvpn.net>

<http://www.openssl.org/>

<http://es.wikipedia.org/>

<http://vtun.sourceforge.net/>

<http://www.oberhumer.com/opensource/lzo/>

<http://www.kernel.org/pub/linux/libs/pam/>

<http://www.ietf.org/rfc/rfc2246.txt/>

<http://openvpn.se/>

<http://sites.inka.de/sites/bigred/devel/tcp-tcp.html/>

Glosario

AES: Estándar de Cifrado Avanzado – *Advanced Encryption Standard*

ACL: Lista de Control de Acceso – *Access Control List*

APT: Sistema de Gestión de Paquetes para GNU/Linux - *Advanced Packaging Tool*

ASLEAP: Protocolo Ligero de Autenticación Extensible de Cisco - *Cisco's Lightweight Extensible Authentication Protocol*

ATM: Modo de Transferencia Asíncrona – *Asynchronous Transfer Mode*

- CA:** Autoridad Certificadora – *Certificate Authority*
- CBC:** Cifrado por Bloques en Cadena – *Cipher Block Chaining*
- CFB:** Cifrado por Bloques con Realimentación – *Cipher Feedback*
- CHAP:** Protocolo de Autenticación por Desafío Mutuo – *Challenge Handshake Authentication Protocol*
- CoS:** Clase de Servicio – *Class of Service*
- CRL:** Lista de Revocación de Certificados - *Certificate Revocation List*
- CSR:** Petición de Firma de Certificados – *Certificate Signing Request*
- DHCP:** Protocolo Dinámico de Configuración de Anfitrión – *Dynamic Host Configuration Protocol*
- DES:** Cifrado de Datos Estándar – *Data Encryption Standard*
- DoS:** Ataque de Denegación de Servicio – *Denial of Service*
- EAP:** Protocolo de Autenticación Extensible – *Extensible Authentication Protocol*
- FTP:** Protocolo de Transferencia de Ficheros – *File Transfer Protocol*
- GPL:** Licencia Pública General de GNU – *GNU General Public License*
- GRE:** Protocolo Genérico de Encapsulamiento de Cisco – *Generic Routing Encapsulation*
- GUI:** Interfaz Grafica de Usuario para Programas – *Graphical User Interface*
- HMAC:** Código de Autenticación de Mensaje Basados en Resúmenes – *Hash Message Authentication Code*
- HTTP:** Protocolo de Transferencia de Hipertexto – *HyperText Transfer Protocol*
- IANA:** Agencia de Asignación de Números de Internet – *Internet Assigned Numbers Authority*
- IETF:** Grupo de Trabajo en Ingeniería de Internet – *Internet Engineering Task Force*
- IKE:** Protocolo para Establecer Asociaciones de Seguridad en IPsec – *Internet Key Exchange*
- IP:** Protocolo de Internet – *Internet Protocol*
- IPsec:** Seguridad para el Protocolo de Internet – *Internet Protocol Security*
- IPv4:** Versión 4 del Protocolo de Internet – *Internet Protocol version 4*
- IPv6:** Versión 6 del Protocolo de Internet – *Internet Protocol version 6*
- IPX:** Intercambio de Paquetes Interred – *Internetwork Packet Exchange*
- ISAKMP:** Protocolo de Establecimiento de Asociaciones de Seguridad y Claves Criptográficas en Internet – *Internet Security Association and Key Management Protocol*
- ISP:** Proveedor de Servicios de Internet – *Internet Service Provider*
- LAN:** Red de Área Local – *Local Area Network*
- LDAP:** Protocolo Ligero de Acceso a Directorios – *Lightweight Directory Access Protocol*

LEAP: Protocolo Ligero de Autenticación Extensible – *Lightweight Extensible Authentication Protocol*

LSP: Camino Basado en Etiquetas en MPLS – *Label Switched Path*

LZO: Algoritmo de Compresión de Datos – *Lempel Ziv Oberhumer*

L2F: Protocolo de Túnel de Capa 2 de Cisco – *Layer 2 Forwarding*

L2TP: Protocolo de Túnel de Capa 2 – *Layer 2 Tunneling Protocol*

MAC: Código de Autenticación del Mensaje – *Message Authentication Code*

MD5: Algoritmo de Resumen de Mensaje 5 – *Message Digest algorithm 5*

MPLS: Protocolo Basado en Etiquetas – *MultiProtocol Label Switching*

MS-CHAP: Versión de Microsoft de CHAP – *Microsoft Version CHAP*

NAT: Traducción de Direcciones de Red – *Network Address Translation*

OFB: Cifrado con Salida Retroalimentada – *Output FeedBack*

OSI: Modelo de Referencia de Interconexión de Sistemas Abiertos – *Open System Interconnection*

PAM: Modulo de Autenticación Reemplazable – *Pluggable Authentication Module*

PAP: Protocolo Simple de Autenticación – *Password Authentication Protocol*

PKCS: Estándar Criptográfico de Clave Pública – *Public Key Cryptography Standars*

POP3: versión actual del Protocolo de Correo Electrónico – *Post Office Protocol*

PPP: Protocolo Punto a Punto – *Point to Point Protocol*

PPTP: Protocolo de Túnel Punto a Punto – *Point-to-Point Tunneling Protocol*

PRF: Función Pseudo Aleatoria – *Pseudo Random Function*

PVC: Circuito Virtual Permanente – *Permanent Virtual Circuit*

QoS: Calidad de Servicio – *Quality of Service*

RADIUS: Protocolo de Autenticación y Autorización para Aplicaciones de Acceso a la Red – *Remote Authentication Dial-In User Server*

RC4: Algoritmo de Cifrado en Flujo Rivest Cipher 4 – *River Cipher 4 stream cipher algorithm*

RPM: Herramienta de Administración de Paquetes para Linux – *Red Hat Package Manager*

RTP: Protocolo de Transporte en Tiempo Real – *Rea-timel Transport Protocol*

SHA: Algoritmo de Hash Seguro – *Secure Hash Algorithm*

SMB: Protocolo de Red para Compartir Archivos e Impresoras – *Server Message Block*

SSH: Protocolo de Acceso Seguro a Máquinas Remotas – *Secure SHell*

SSL: Seguridad de la Capa de Transporte – *Secure Socket Layer*

TACACS+: Sistema de Control de Acceso del Controlador de Acceso a Terminales – *Terminal Access Controller Access Control System*

TCP: Protocolo de Control de Transmisión – *Transmission Control Protocol*

TELNET: Protocolo de Acceso a Máquinas Remotas – *TELEcommunication NETwork*

TLS: Seguridad de la Capa de Transporte – *Transport Layer Security*

UDP: Protocolo de Capa de Transporte Basado en Datagramas – *User Datagram Protocol*

VPN: Red Privada Virtual – *Virtual Private Network*

XINETD: Demonio Extendido de Internet – *eXtended InterNET Daemon*