Capítulo 4

Implementación de OpenVPN

OpenVPN es un software de código abierto que ha desmostrado tener un robusto diseño y un desarrollo continuo tanto por el núcleo principal de desarrolladores como por la comunidad formada alrededor de esta solución. Es lo suficientemente flexible para que se adapte a las necesidades de desarrollo de este trabajo. El siguiente capítulo muestra el trabajo llevado a cabo, su descripción detallada y los resultados obtenidos de dicho esfuerzo.

Capítulo 4 Implementación de OpenVPN

4.1 Objetivos y metas con OpenVPN

El principal objetivo de este trabajo es lograr la difusión, entre la comunidad universitaria, tanto la academica-estudiantil como el público en general, del protocolo de nueva generación. Así como fomentar la adopción de los mecanismos de transición necesarios actualmente para llevar a cabo la transición entre la anterior y la siguiente versión del protocolo de internet. Esto ayudará a generar las buenas prácticas entre los encargados del mantenimiento y soporte a las redes así como el publico en general.

4.2 Descripción de funciones

La implementación de esta VPN, proveerá al público usuario de las siguientes funciones:

1. Autenticación y verificación de credenciales validadas por el servidor.

2. Administración sobre el periodo convenido sobre el servicio.(sujeto a las políticas de seguridad acordadas previamente)

3. Acceso a la red virtual privada.

4. Soporte para Ipv6.

5. Limitados privilegios al usuario.

4.3 Estructura de operación y control

Se contará con un equipo que ofrecerá el servicio de conexión a la vpn, el cuál realizará estas funciones de acuerdo a la demanda que se genere y a la disposición posible. Este se encontrará alojado en las instalaciones de la Dirección General de Cómputo Académico (DGSCA) de acuerdo a los lineamientos acordados de disponibilidad y requerimientos de la dirección de conformidad a su conveniencia.

La parte del control estará definida por las necesidades y disponibilidades del Laboratorio de Tecnologías Emergentes (NetLab), así como de los responsables a cargo del área de IPv6. De acuerdo a estas circunstancias será como se maneje y definan las políticas de acceso, disponibilidad así como también los mecanismos de control que sean necesarios para el buen manejo y las buenas prácticas que se pudieran desarrollar con dicho servicio.

4.4 Instalación de OpenVPN y soporte para IPv6

Para la instalación de OpenVPN se procedió a descargar el paquete que contenga el código fuente correspondiente a dicho software. En este caso al momento de documentar esta parte, se encontraba en su versión 2.1.1. En la página oficial del proyecto encontraremos la liga correspondiente a las descargas tanto del código fuente como de los binarios ejecutables para la plataforma Windows http://www.openvpn.net/index.php/open-source/downloads.html.

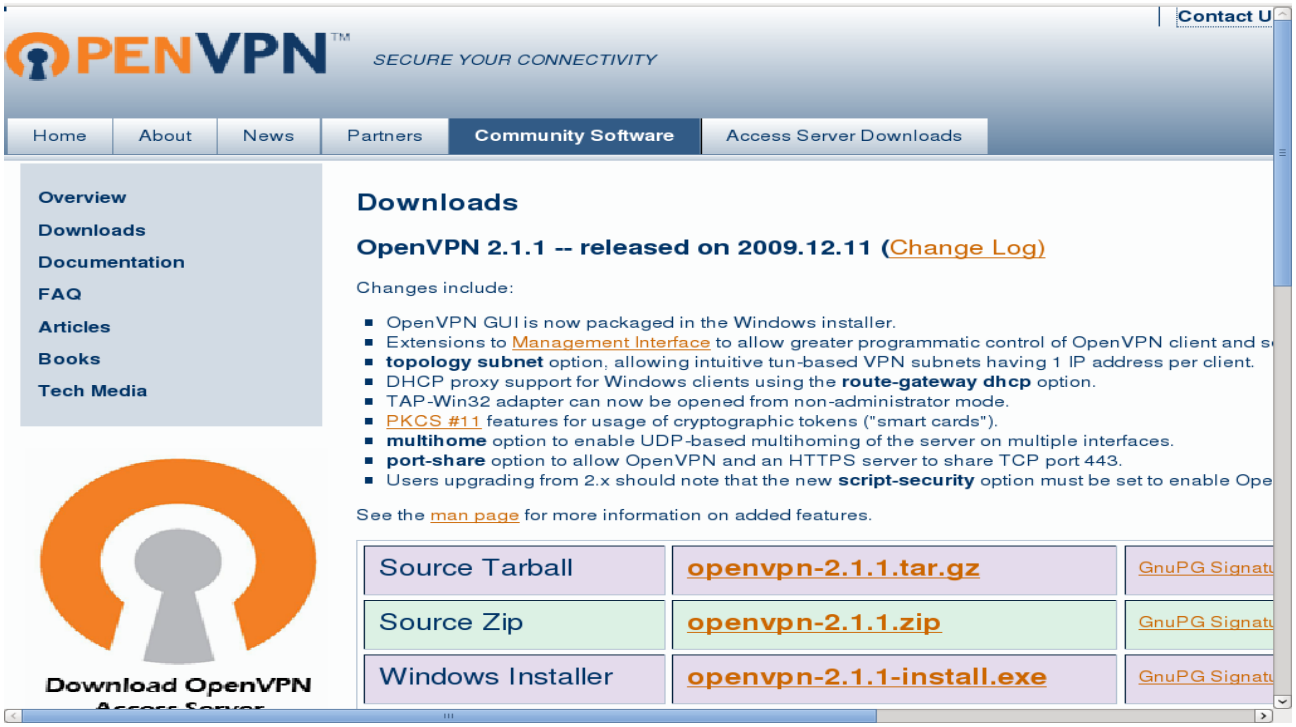

Imagen 5. "Sitio web oficial del proyecto OpenVPN"

Se procedió a descomprimir el paquete a fin de generar el directorio en donde se encontraban los distintos archivos del código fuente que eran necesarios para generar esta versión del software. El comando que se requiere para esto es el siguiente:

netlab@netlab:~/Downloads$ tar xvfz openvpn-2.1.1.tar.gz

de cuya respuesta obtenemos el resultado mostrado a continuación.

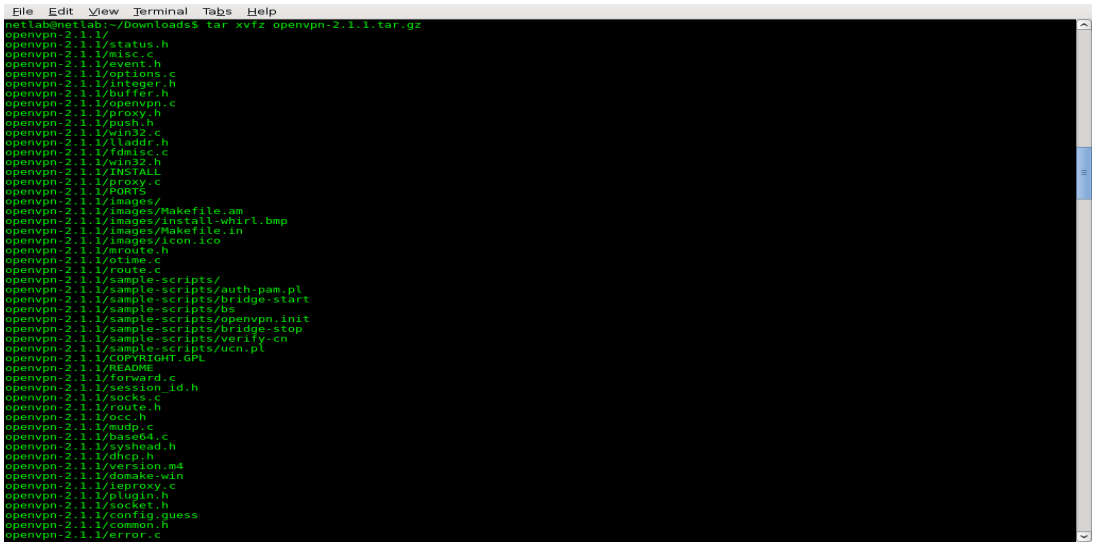

Imagen 6 “Captura de pantalla al descomprimir el paquete OpenVPN”

Esta acción nos creó automáticamente un directorio con todos los archivos necesarios para proceder a generar este software. En este directorio, que se llama openvpn-2.1.1, encontramos los siguientes archivos de configuración del código fuente.

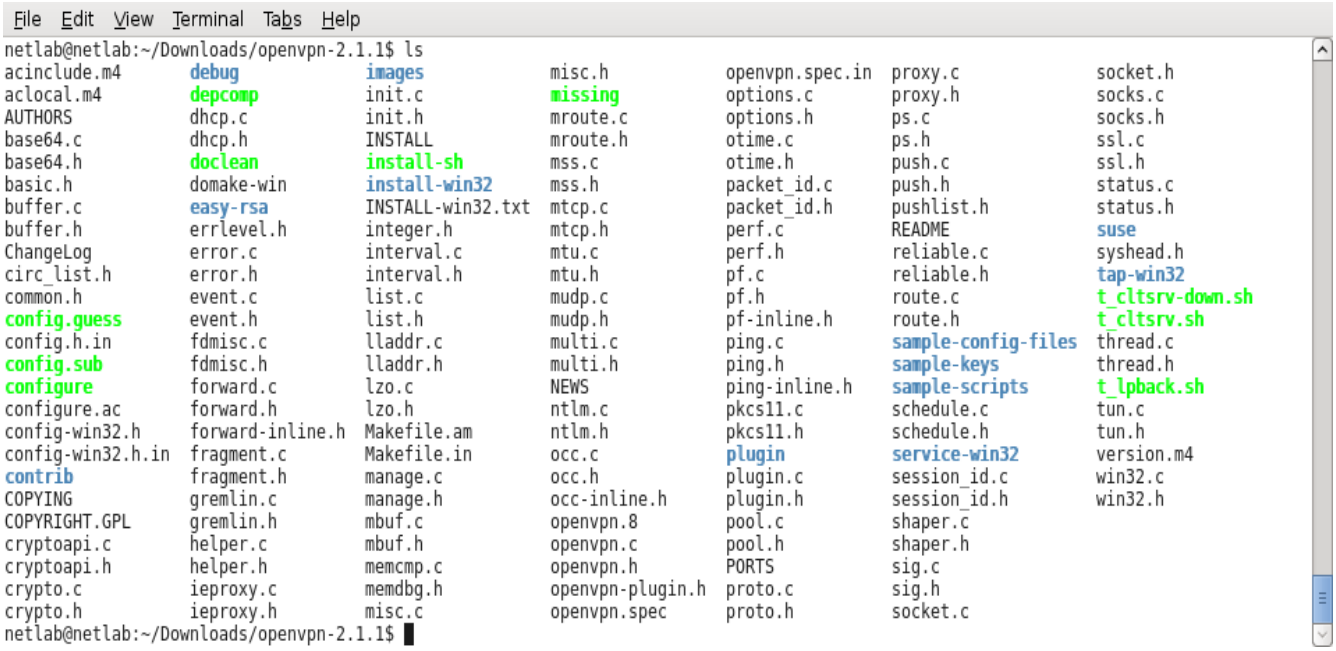

Imagen 7 “Listado de los paquetes del código fuente de OpenVPN”

Como se aprecia claramente, existen distintos tipos de archivos que corresponden a las diferentes funciones que se necesitan para crear el paquete. En este momento, se puede elegir crear el paquete tal cual se tiene, es decir seguir las instrucciones que se hallarán comúnmente en el archivo llamado INSTALL. Las cuales son:

- escribir en la línea de comandos *./configure* lo que debe entenderse como punto diagonal y teclear configure. Seguido de presionar el botón de Enter.
- Después teclear *make*. Seguido de presionar el botón Enter.
- Finalmente tecleamos *make install* . Y también presionamos el botón Enter.

Esto nos llevaría a obtener el paquete openvpn-2.1.1 tal cual se ofrece en la página web oficial del proyecto, pero para nuestros propósitos es necesario cursar otras vías. Una de ellas es la utilización de parches de software que los desarrolladores ofrecen a fin de mejorar o cambiar el comportamiento del software. Uno de ellos es ofrecido por Juan Jóse Ciarlante, el cual es posible descargarlo de la siguiente dirección http://github.com/jjo/openvpn-ipv6/downloads y otro se ofrece por parte de Bernard Schmidt y Gert Döring en http://www.greenie.net/ipv6. Obtendremos el siguiente paquete openvpn-2.1.1-20100307.tar.gz, el cual es el parche que requerimos aplicar al código fuente original para poder agregarle capacidades IPv6 al software principal. Lo hacemos con la siguiente instrucción :

patch <openvpn-2.1.1-20100307.patch

Esta acción nos dará como resultado tener un código que ya tiene algunas de las características que necesitamos. Modifica una serie de archivos para poder agregar las capacidades adicionales en el código fuente del software que se generó.

4.5 Pruebas de Interoperabilidad

Las pruebas que se llevaron a cabo exitosamente incluyen a los sistemas operativos tipo Unix. Estos son Debian 5 Lenny, Fedora 11 Leonidas y FreeBSD 8.0. En todos los casos estos funcionaron como servidores y clientes para ambas versiones del protocolo de internet, los sistemas Windows en sus versiones XP Professional Edition y Vista Home Edition fueron incapaces de lograr la conexión como cliente o servidor para IPv6, el soporte para IPv4 está garantizado. La Tabla 4 tiene el resumen de los resultados de las pruebas de conectividad para distintos sistemas operativos usados como clientes o servidores respectivamente.

Conectividad IPv6 VPN						
Cliente / Servidor	Windows Vista SP1	Windows XP SP3	FreeBSD 8	OpenBSD 4.6	Fedora 11	Debian 5
FreeBSD 8	Sin soporte	Sin soporte	----------	No funcionó	Sí	Sí
OpenBSD 4.6	Sin soporte	Sin soporte	No funcionó	----------	No funcionó	No funcionó
Fedora 11 Leonidas	Sin soporte	Sin soporte	Sí	No funcionó	----------	sí
Debian 5 Lenny	Sin soporte	Sin soporte	Sí	No funcionó	Sí	----------

Tabla 4.“Resumen de los resultados de la conectividad con IPv6 en la VPN. Modelo cliente-servidor punto a multipunto.”

4.6 Administración de fallas y cambios

A lo largo del desarrollo de este trabajo, se probaron distintos caminos entre los cuales destacan dos, el uso de scripts en ambos puntos de la conexión y el uso de código parchado a fin de generar el resultado deseado.

Con el primer camino, el uso de scripts, el proceso de implementación de este servicio se volvía engorroso para el administrador y los usuarios. Ambos tenían que tener conocimientos muy claros de lo que querían lograr y cómo hacerlo. Cualquier configuración incorrecta en alguno de los dos extremos de la conexión, cliente o servidor, y esta fallaba.

Para el caso del uso de parches aplicados al código original del proyecto, tenemos que decir que fue la opción viable para un servicio cómo el que se plantea implementar. Gracias al trabajo de estos desarrolladores alrededor del mundo fue que se tuvo un producto adecuado y viable. La administración del servicio es clara y simple de parte del servidor. El administrador responsable y enterado de lo que esta haciendo y tiene en sus manos, sabrá manejarlo. El usuario no tiene que tener grandes conocimientos de sistemas para poder hacer uso del servicio.

Se tenía planteado ofrecer un servicio multiplataforma que abarcara a las plataformas más usadas. Pero no fue posible tenerlo para el sistema MS Windows. La documentación que hay para este sistema esta fragmentada y en algunos casos es obsoleta o contradictoria, cómo la que se encuentra en línea en la página del proyecto OpenVPN. Las instrucciones de compilación que se encuentran en la página en línea pertenecen a versiones anteriores y las instrucciones que se pueden leer en el código fuente del programa no son del todo precisas.

Hay que añadir la gran cantidad de software adicional que es necesario para la generación de los ejecutables del software y el controlador que es necesario a fin de que funcione el programa en este sistema operativo. Específicamente el uso de bibliotecas de programación propietarias, llamadas DDK, que son parte del paquete de desarrollo de controladores para Windows (a) WDK. La versión que se solicita ya no esta disponible para su descarga en la página de Microsoft, lo cuál fue un impedimento para conseguir ambas partes del software, únicamente se tiene los ejecutables para el IPv4 no para IPv6, y no se pudo generar el controlador que soporte el IPv6.

Ventajas y desventajas de la VPN

- *Administración sencilla, centralizada, robusta y flexible.*
- *Bajo costo de implementación y mantenimiento.*
- *Alta escalabilidad y adaptación.*
- *Modular por diseño de origen y seguridad opr demanda.*
- *Archivos de configuración bien diseñados y bitacoras explícitas.*

- *Un solo puerto es usado por la VPN.*
- *Transparente a los usuarios.*
- *Protege los puntos finales de conexión.*

- *Requiere alta especialización por parte de los administradores del servicio.*
- *No es transparente a los servidores, requiere tener acceso autorizado por el firewall de los respectivos puntos de conexión, tanto punto a punto como punto a multipunto*
- *Ámplios conocimientos de los campos de redes, seguridad, administración y configuración.*
- *Constante actualización del conocimiento del producto y sus escenarios.*
- *En el producto libre la documentación está fragmentada, desarticulada y desactualizada.*
- *Sin la correcta administración y las políticas necesarias puede ser un boquete de seguridad.*

4.7 Generación de scripts y manual de conectividad

Los scripts se encontrarán en un archivo llamado *server.conf* o *client.conf* dado el caso, esto es por convención y para facilitar su ubicación a la vez de dar a entender su función de una manera clara y sencilla. Se ubicará preferentemente en los archivos de configuración que maneja el sistema, esto variará dependiendo de la elección del sistema operativo que sea el servidor, pero usualmente será algo como */etc/openvpn.*

El contenido de dicho archivo contiene las instrucciones necesarias para que el servidor provea las funcionalidades necesarias. Como se observa a continuación, esta es la configuración más básica par lograrlo:

```
port 1194
proto tcp
dev tun0
ca ca.crt
cert server.crt
key server.key
dh dh1024.pem
server-ipv6 10.8.0.0 255.255.255.0 2001:1218:1:6:42f:dd3a:b9b7:c0e9/64
ifconfig-pool-persist ipp.txt
comp-lzo
user nobody
group nobody
max-clients 10
keepalive 10 120
persist-key
persist-tun
status status-servidor.log
log-append bitacora-del-servidor.log
verb 9
reneg-sec 60
```

Cuandro 7 “Contenido del archivo server.conf”

En esta configuración, que se muestra en el cuadro 6, estamos habilitando los parámetros necesarios para que nuestro servidor pueda ofrecer conexión. Cada una de las opciones que se tienen se explican a continuación:

port 1194: nos habilita el puerto 1194, mismo que fue definido por la IANA como el estandar para esta software. Es posible utilizar cualquier otro, de acuerdo a las necesidades del administrador o las políticas que se hayan definido previamente.

proto tcp: habilita el protocolo a usar, que en este caso es tcp, pero existe otra opción que es udp. Dependerá de los requerimientos posteriores del servicio, el usar uno u otro.

dev tunX : define la interfaz a usar como medio de conexión, en este caso corresponde a la interfaz tun. También se puede usar la interfaz *tapX*. Los requerimientos del servicio determinarán cuál de estas es la más adecuada para esta implementación.

ca ca.crt: solicita el certificado de autenticación que el servidor manejará para firmar los certificados de los clientes a fin de llevar un control de los usuarios que hagan uso del servidor.

cert server.crt : maneja el certificado propio del servidor, al hacer el manejo de autenticación este es necesario para diferenciarlo de los clientes.

key server.key: controla la llave privada del servidor. A través del manejo de las llaves es cómo se autentica a los clientes del servidor.

dh dh1024.pem: define cual será el parámetro de seguridad a manejar. La cantidad de bits que manejará el algoritmo diffie-hellman para verificar la autenticidad de los certificados y las llaves.

server-ipv6: esta sentencia indica claramente que se trata del servidor y que además estamos manejando la parte IPv6. Se tienen que definir ambas direcciones IPv4 e IPv6, con sus respectivas máscaras de red.

Ifconfig-pool-persist: esta instrucción maneja el archivo de texto ipp.txt, en el cuál se almacenarán las direcciones asignadas a los clientes que manden peticiones de conexión al servidor.

comp-lzo: define la compresión que se utilizará en la conexión entre el cliente y el servidor.

u*ser nobody*: en particular esta opción sólo aplica a los sistemas tipo Unix verdaderos. Disminuye los privilegios del servicio ya que no permite usuarios en el grupo de usuarios de este servicio.

group nobody: se usa junto con la anterior instrucción para administrar de manera más segura el servicio.

max clients: limita la cantidad de clientes que serán usuarios del servicio.

keepalive: es una directiva auxiliar para simplificar la expresión de –**ping** y –**ping-restart** en las configuraciones de modo servidor.

persist-tun: evita cerrar/reabrir el dispositivo TUN/TAP o ejecutar/cerrar scripts a través de **SIGUSR1** o reinicios de –**ping-restart**.

persist-key: no relee archivos de llaves a través de –**ping-restart**.

log-append: crea la bitácora con los mensajes de conexión, intercambio de llaves, autenticación, etc. Muy útil si se necesita depurar o encontrar errores.

status: genera un archivo con el estado de la conexión.

verb: indica la generación de mensajes explícitos acerca de los estados de los procesos involucrados en la conexión.

reneg-sec: fija el tiempo de renegociación durante la conexión, según las necesidades. El default es 60 segundos.

El contenido de dicho archivo para los clientes contiene las instrucciones necesarias para que el servidor provea las funcionalidades necesarias. Como se observa a continuación, esta es la configuración más básica par lograrlo:

```
port 1194
proto tcp
dev tun0
ca ca.crt
cert client.crt
key client.key
comp-lzo
keepalive 10 120
persist-key
persist-tun
status status-client.log
log-append bitacora-del-cliente.log
verb 9
reneg-sec 60
```

Cuadro 8 “Cuadro del archivo client.conf”

En esta configuración, que se muestra en el cuadro 7, estamos habilitando los parámetros necesarios para que el cliente pueda obtener conexión del servidor VPN. Cada una de las opciones que se tienen se explican a continuación:

port 1194: nos habilita el puerto 1194, mismo que fue definido por la IANA como el estándar para este software. Es posible utilizar cualquier otro, de acuerdo a las necesidades del administrador o las políticas que se hayan definido previamente.

proto tcp: habilita el protocolo a usar, que en este caso es tcp, pero existe otra opción que es udp. Dependerá de los requerimientos posteriores del servicio, el usar uno u otro.

dev tunX : define la interfaz a usar como medio de conexión, en este caso corresponde a la interfaz tun. También se puede usar la interfaz *tapX*. Los requerimientos del servicio determinarán cuál de estas es la más adecuada para esta implementación.

ca ca.crt: solicita el certificado de autenticación que el servidor manejará para firmar los certificados de los clientes a fin de llevar un control de los usuarios que hagan uso del servidor.

cert client.crt : maneja el certificado propio del cliente, al hacer el manejo de autenticación este es necesario para diferenciarse del servidor y autenticarse ante el.

key client.key: controla la llave privada del cliente. A través del manejo de las llaves es cómo se autentica el cliente ante el servidor.

comp-lzo: define la compresión que se utilizará en la conexión entre el cliente y el servidor.

keepalive: es una directiva auxiliar para simplificar la expresión de –**ping** y –**ping-restart** en las configuraciones de modo servidor.

persist-tun: evita cerrar/reabrir el dispositivo TUN/TAP o ejecutar/cerrar scripts a través de **SIGUSR1** o reinicios de –**ping-restart**.

persist-key: no relee archivos de llaves a través de –**ping-restart**.

log-append : crea la bitácora con los mensajes de conexión, intercambio de llaves, autenticación, etc. Muy útil si se necesita depurar o encontrar errores.

status: genera un archivo con el estado de la conexión.

verb: indica la generación de mensajes explícitos acerca de los estados de los procesos involucrados en la conexión.

r*eneg-sec* : fija el tiempo de renegociación en el proceso de conexión, según las necesidades. El default es 60 segundos.

4.8 Realimentación a la propuesta

A fin de mejorar la propuesta se buscó cambiar su diseño de la implementación original. Al principio de la misma, se tenía planeado realizarla a través de una serie de scripts del lado del servidor y del cliente. A lo largo de los más dos años que este trabajo duró, aparecieron desarrolladores independientes del núcleo original de desarrollo del OpenVPN, que comenzaron a ofrecer sus parches al código fuente original a fin de que éste pudiera ofrecer características IPv6, que la comunidad de usuarios había estado solicitando. Con el uso de estos parches se logró el objetivo principal del planteamiento de la tesis, el cual es un servicio de una VPN con soporte para IPv6.