Universidad de Buenos Aires

Facultades de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Maestría en Seguridad Informática

Trabajo Final de Maestría

Implementación, Gestión y Soporte de VPNs Open Source a Gran Escala

Autor:
Andrés Villegas

Director de Trabajo Final:
Hugo Pagola

Entrega: Octubre de 2015

Cohorte 2014

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Tesis vigente y que se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

FIRMADO
Andrés Villegas
DNI: 28.414.685

Resumen

Debido a la globalización y al abaratamiento de las comunicaciones, la interacción entre organizaciones y la integración de sistemas y servicios informáticos de terceros ha crecido muy rápidamente en los últimos años. Estos cambios traen aparejados un gran desafío para el área de seguridad informática de las organizaciones ya que implican nuevas amenazas relacionadas con la confidencialidad e integridad de los datos que viajan en esas comunicaciones y con la disponibilidad de los servicios prestados a través de dichos enlaces.

En el presente trabajo se intentará mostrar la implementación de un subproyecto exitoso basado en VPNs SSL/TLS de software libre, usando OpenVPN, y desarrollo propio de una organización que, hizo viable un proyecto de gran envergadura con problemas presupuestarios.

La implementación del presente proyecto implicó el despliegue de una infraestructura particular, que trajo aparejado distintos tipos de problemas de gestión, administración y soporte de esos enlaces seguros. A lo largo del documento se brindará un análisis profesional sobre los problemas y las soluciones implementadas en cada caso.

Palabras clave

Conexiones seguras, VPN, autentificación, autorización, certificados digitales, disponibilidad de servicio, integridad de datos, confidencialidad en comunicación, gestión, soporte.

Índice

Introducción

La globalización, la masificación en las comunicaciones, la digitalización de procesos y la distribución geográficas de potenciales clientes impulsan a la tecnología a ser el soporte necesario para la implementación de estas comunicaciones con otras organizaciones para la integración de servicios informáticos de terceros que sean la base para generar un mayor valor agregado. El área de TI y el área de seguridad informática tienen un gran desafío por delante, ya que deben brindar el soporte necesario para alcanzar estos nuevos objetivos de manera segura. Es fundamental en estas comunicaciones que se asegure no comprometer la integridad y la confidencialidad de los datos transmitidos y brindar una infraestructura tolerante a fallas que asegure la continuidad de servicios ante imprevistos.

Las relativamente nuevas tecnologías de comunicación y la seguridad en comunicaciones brindada por Redes Privadas Virtuales (VPN) facilita y hace viables procesos distribuidos que hasta hace no mucho requerían el servicios de logística para el traslado de documentación física. La eficiencia alcanzada por la inmediatez en las comunicaciones digitales y la seguridad brindada impulsa a las organizaciones a abandonar estos viejos procesos y las empuja a renovarlos.

En el presente Trabajo Final de Maestría se mostrará en detalle un caso de éxito de implementación de VPNs open source que se inició como una mejora para los clientes más importantes de la organización y que terminó siendo uno de los pilares principales para la renovación integra del proceso completo para todos los clientes.

A lo largo del desarrollo del Trabajo se muestra como fue madurando el proyecto. En el primer capítulo se detallan las especificaciones de los requerimientos iniciales. Luego, en el segundo capítulo, se incluye el análisis y el diseño de los procedimientos que darán sustento a la solución. Después, en el tercer capítulo, se especifican los detalles del desarrollo e implementación de la solución. Finalmente, el cuarto capítulo detalla una

propuesta de mejora, un problema no previsto y su solución y un tema a tener en cuenta en el mantenimiento a largo plazo.

Este proyecto fue planificado, desarrollado e implementado por mí y otras dos personas. Luego de la primera versión se contrataron otras cinco personas que fueron las encargadas de realizar las primeras instalaciones. Hoy en día dos de los integrantes iniciales continuamos haciendo el soporte técnico de segundo nivel, la administración de los servicios y su mantenimiento, mientras que para el soporte de primer nivel se cuenta con cuatro personas que dan el soporte telefónico a problemas de conectividad y nuevas instalaciones.

Requerimientos y especificaciones

En un primer momento el requerimiento hablaba de conectar los principales clientes, eran 30 sitios aproximadamente con unos 10 puestos de trabajo cada uno. Este requerimiento mutó, muy rápidamente, hacia la posibilidad de conectar todos los clientes y eliminar por completo el proceso de registros no informatizado que se hacía previamente en papel. Este nuevo escenario representaba una solución que permitiera conectar más de mil quinientos sitios y aproximadamente tres mil puestos de trabajo.

Como el tema presupuestario era un impedimento importante para alcanzar el objetivo, se pensó en instalar concentradores VPNs en cada sitio cliente con arquitectura estándar (x86) y software open source que permitiera que cada cliente pueda comprar o usar computadoras propias para sumarse a la red de VPNs sin tener que pagar licencias de software adicional y/o hardware específico. Estos concentradores además de brindar el acceso a los puestos instalados localmente en los sitios clientes, contribuirían en el procesamiento que implica la autentificación de los tres mil puestos de trabajo que se encontrarían tras ellos.

Adicionalmente, era imperioso tener una manera de solucionar problemas remotamente y dar soporte a los operadores remotos sin tener que viajar a los sitios clientes. Para ellos se decidió crear un CD de instalación que permitiera a los clientes con escasos conocimientos de informática poder ser su propio soporte técnico con una simple asistencia telefónica. La administración de estos concentradores sería exclusiva de la organización central y en ningún caso sería compartida con el cliente, por lo que se decidió implementar medidas de seguridad adicionales para verificar que los concentradores no sean modificados o accedidos. Estas medidas serán desarrolladas a lo largo de este documento.

A continuación se detallan los requerimientos funcionales y de seguridad que fueron determinados para el presente proyecto.

Requerimientos funcionales

RF1. Permitir la conexión desde los aproximadamente tres mil puestos de trabajo remoto a la aplicación web de toma de registros.

RF2. Tener un monitoreo de los sitios conectados, no conectados y con algún tipo de problema.

RF3. Brindar las herramientas necesarias para poder dar un soporte remoto completo de la conectividad.

RF4. Brindar un sistema de instalación que sea lo suficientemente sencillo para que cualquier operador con la mínima información sobre la red que usa pueda instalar o reinstalar completamente los sistemas de VPN.

RF5. Desarrollar un modo de instalación personalizada que permita al cliente autentificarse, registrar la instalación y descargar las credenciales de VPN necesarias.

RF6. Bloquear las instalaciones y reinstalaciones no autorizadas.

RF7. Permitir la generación de clientes a través de una interfaz de usuario cómoda, registrando nombre del cliente, ubicación, cantidad de puestos y clasificación (nacional, regional o internacional).

RF8. Automatizar la generación de archivos requeridos para la instalación de modo que estos puedan ser generados por el operador y publicados para que sean descargados por el cliente remoto a través de Internet.

RF9. El módulo de gestión de VPN debe permitir la modificación de datos de los clientes, pudiendo registrar, contacto técnico, dirección postal, teléfono y geolocalización del cliente.

RF10. Se debe permitir la generación de nuevos puestos de trabajo para un cliente ya instalado.

RF11. Brindar al operador la posibilidad de habilitar la reinstalación de un cliente, pudiendo adicionalmente revocar las credenciales generadas previamente.

RF12. Contar con un módulo de auditoría general en el que se deje registro

de las conexiones, desconexiones, instalación, reinstalaciones y otras operaciones de cada cliente que se permita realizar en el módulo de Gestión de VPNs. Este módulo deberá registrar fecha y hora, identificador del cliente, acción realizada, operador que intervino (si hubiere) y otros datos complementarios.

RF13. Brindar un tablero de estadística que permita ver el progreso de instalación de los clientes y un resumen de los estados de conexión de estos.

RF14. Ofrecer un mapa que muestre los clientes geolocalizados, identificando el estado de conexión de cada uno.

Requerimientos de seguridad

RS1. Asegurar la privacidad y autentificación de la conexión remota que viaja por Internet desde el sitio remoto hasta el concentrador VPN central.

RS2. Debido a que las redes cliente pueden no ser seguras, es preciso que el túnel VPN que asegura la privacidad y autentificación de la conexión comience en el puesto, o sea, desde la misma interfaz de red del puesto de trabajo hasta la interfaz de red del concentrador VPN central.

RS3. No permitir la descarga de archivos requeridos para la instalación a usuarios no autentificados.

RS4. Asegurar las conexiones hechas durante la instalación para descargar las credenciales de modo que el servicio no esté disponible para aquellos que no empleen el CD generado por la organización. Adicionalmente estas conexiones deben ser seguras, garantizando confidencialidad y autentificación en las transferencias.

RS5. Garantizar que las credenciales de usuarios locales y las reglas de firewall instaladas en el concentrador VPN remoto no han sido modificadas después de la instalación.

RS6. Garantizar que el hardware usado durante la instalación es el mismo

usado en cada conexión, en caso de cambiar de hardware deberá autorizarse o reinstalarse el concentrador VPN remoto.

RS7. Durante la conexión, y solamente después de validar todos los requisitos, se habilitará el acceso en el firewall que permita a los puestos de trabajo del cliente operar contra la aplicación web de registro.

RS8. Tener un clúster de servidores VPN configurados en alta disponibilidad que permita garantizar que ante la caída de uno de los servidores otro servidor del clúster brindará el servicio asegurando la disponibilidad de la conectividad en todo momento.

RS9. Tener redundancia de servidores de instalación que permitan ante la falla de uno tener alta disponibilidad y que no se impida realizar la instalación por la falla de un servidor.

RS10. Brindar además redundancia de enlaces que asegure que ante la caída de uno de ellos los clientes puedan conectarse por otro sin necesidad de modificar las configuraciones de conexión VPN. Para cumplir con este requerimiento se deberá contar con dos o más servicios de conectividad de diferentes proveedores.

Detalle técnico de OpenVPN

Sobre SSL/TLS

SSL/TLS es el protocolo de seguridad más ampliamente implementado en el mundo hoy [1]. Esto no implica una garantía total de seguridad pero si nos asegura su constante testeo y mejora, ya que su popularidad hace que los más reconocidos criptógrafos trabajen e investiguen sobre sus vulnerabilidades y aporten su trabajo para mejorarlo.

SSL fue desarrollado originalmente por Netscape Communications Corporation, más tarde Microsoft incorporó un desarrollo similar en sus sistemas y finalizando los 90s IETF tomó control del desarrollo de SSL y le cambio el nombre a TLS (Transport Layer Security) en un intento de consolidar los diferentes desarrollos para lograr un estándar abierto. TLS equivale a SSLv3 con algunos parches y mejoras menores. A lo largo del presente trabajo se hará referencia a SSL, TLS o SSL/TLS sin distinción [2].

Específicamente SSL/TLS usa una de las mejores tecnologías de encripción conocidas para autentificar los extremos de la comunicación VPN. Esta forma de encripción se la conoce como encripción asimétrica y, básicamente, funciona de la siguiente manera: ambos extremos tienen un par de claves, conocidas como clave pública y clave privada generadas a partir de un algoritmo matemático [3]. Debido a este ingenioso algoritmo lo cifrado mediante el uso de una de las claves sólo puede ser descifrado por el uso de la otra clave y viceversa. Cada usuario de VPN debe tener una clave privada, que sólo él conozca y una pública que distribuya para que todos la tengan.

De esta manera, la encripción asimétrica puede usarse para la autentificación bidireccional y el intercambio de claves. Supongamos un ejemplo simple: El cliente A, genera una número aleatorio, se lo envía a B, B lo cifra con su propia clave privada, y lo devuelve a A, A puede descifra el mensaje usando la clave pública de B, si el mensaje coincide con el enviado la identidad de B ha sido validada, ya que sólo B tiene su clave privada; este

procedimiento es conocido como firma digital. Para el intercambio de claves A podría enviar una nueva clave a B cifrándola con la clave pública de B, de modo que sólo B podría descifrar el mensaje.

Ahora el problema aparece cuando el número de clientes crece y todos deben mantener actualizadas las claves públicas de todos. Al crecer el número de clientes la complejidad administrativa se dispara de manera exponencial. La solución a este problema se da mediante la implementación de una Infraestructura de Clave Pública (PKI). Dicha implementación, explicada someramente, se basa en la generación de un certificado para la Autoridad Certificante (CA) que será usado para firmar digitalmente los certificados de los clientes en quien confía.

Un certificado no es más que una clave pública, más cierta información preestablecida respecto a la identidad del usuario que usará dicha clave pública, firmada digitalmente por un tercero de confianza, en nuestro caso la Autoridad Certificante. Ahora cada cliente sólo debe tener la clave pública de la CA en quien confía para poder validar el certificado de cualquier otro cliente firmado digitalmente por ésta.

Ya hemos dicho que SSL/TLS es protocolo de seguridad más ampliamente usado en todo el mundo. Se usa para transacciones seguras, para operaciones financieras, para comercio electrónico, incluso lo usamos todos los días para acceder al homebanking. Esta última aplicación del protocolo se basa en que todos los navegadores web modernos incluyen los certificados de las Autoridades Certificantes más importantes del mundo. El proceso es transparente para nosotros, pero el navegador valida que el certificado del banco esté firmado digitalmente por alguna de las Autoridades Certificantes que él conoce y en las que confía. A partir de esto se realiza un intercambio de claves y se genera un túnel SSL sobre el que viajan los mensajes HTTP. Esto brinda a los clientes del banco dos cosas: certeza de que la conexión está siendo realizada contra el banco (autentificación del servidor) y además, confidencialidad e integridad en el intercambio de información.

Seguridad de OpenVPN

La decisión de usar OpenVPN fue tomada después de analizar algunas alternativas como IPSec y su implementación OpenS/WAN, que descartamos por su complejidad de instalación y configuración. OpenVPN brinda un nivel de seguridad más que aceptable basada en SSL/TLS y tiene una simplicidad de administración muy grande y una alta flexibilidad de posibles configuraciones; lo que nos permitirá cumplir con todos los requisitos especificados.

Específicamente, OpenVPN tiene dos formas de autenticar a los clientes: llaves estáticas precompartidas y mediante el uso de certificados SSL/TLS.

Si se configura el uso de llaves estáticas precompartidas, una clave estática es compartida antes de iniciar la conexión. Esta clave única es utilizada por ambos extremos para la autentificación y para la encripción/desencripción de la comunicación. Esta alternativa tiene el gran problema y debilidad en la distribución de claves, por ese motivo fue descartada para nuestra implementación.

La configuración TLS, usa certificados SSL/TLS para la autentificación y el intercambio de claves de encripción. En este modo, la sesión es establecida con autentificación bidireccional, esto quiere decir que tanto servidor como cliente deben autentificarse usando la clave privada de su propio certificado firmado por una misma Autoridad Certificante (CA). Si la autentificación es satisfactoria se generan de manera aleatoria las claves de encripción/desencripción y HMAC, y se intercambian sobre la encripción SSL/TLS. Se deben notar dos cosas, tanto cliente como servidor proveen parte del material aleatorio y ninguna clave simétrica es usada bidireccionalmente. O sea, cada extremo de la conexión tiene cuatro claves, una para el envío cifrado, una para descifrar lo recibido, una para generar los HMAC y uno para comprobar los HMAC recibidos [4].

HMAC (Hash Message Authentication Code) es un código enviado

junto con cada mensaje que sirve para validar que el mensaje no fue alterado. Básicamente es una función hash donde interviene el uso de una clave que sólo conocen los extremos de la conexión; se usa una clave para evitar que cualquier intruso que modifique o genere un mensaje pueda generar un HMAC válido y así adulterar la comunicación. Cualquiera de los extremos que reciba un paquete con un HMAC inválido descartará el paquete automáticamente.

Los paquetes cifrados contienen los tres campos siguientes: HMAC (de los dos campos siguientes), Vector Inicializador y el sobre cifrado. El sobre cifrado tiene dentro un número de secuencia y los carga de datos (payload data). Está claro que el Vector Inicializador y el HMAC viajan en texto plano [4].

Las librerías OpenSSL provee varias APIs con diferentes funciones de encripción. OpenSSL EVP es una interfaz API de programación de alto nivel que provee funciones de cifrado y descifrado simétrico [5][6]. Específicamente para OpenVPN, OpenSSL EVP provee las funciones de HMAC, encripción y desencripción, y mediante la configuración de OpenVPN se puede elegir el algoritmo de cifrado, los tamaños de las llaves y el algoritmo de hash para el HMAC. Por defecto, OpenVPN usa BlowFish CBC para cifrado y SHA-1 como algoritmo de hash para calcular el HMAC [4].

Adicionalmente OpenVPN provee la alternativa de usar la opción *tls-auth* que nos permite definir una clave estática precompartida que se usará para el HMAC durante la negociación inicial de TLS, impidiendo así, ataques de *buffer overflow* de OpenSSL u otros ataques de denegación de servicio (DoS). El atacante no podrá siquiera iniciar una negociación TLS ya que sus paquetes serán descartados por no poder calcular los HMAC correctos [7].

OpenVPN como proceso

OpenVPN trabaja en espacio de usuario dentro del sistema operativo. Lo que significa que puede correrse como un usuario de bajos privilegios y

que no requiere modificaciones complejas en el Kernel del sistema operativo. Además, este proceso no interferirá con los procesos de sistema operativo lo que provee una mejora en la estabilidad y seguridad respecto a VPNs que trabajan a nivel de Kernel como es IPSec [2].

Algunas funcionalidades aplicadas en OpenVPN

A continuación se detallarán funcionalidades que serán usadas en el proyecto. Como se verá, algunas son provistas por OpenVPN y otras no, para estas últimas deberemos buscar alternativas.

Rendimiento

El procesamiento requerido para cifrar y descifrar el tráfico de red, requiere uso de ciclos de CPU. Por lo que se deduce de manera directa que el rendimiento dependerá del hardware. Se han realizado pruebas sobre un Pentium III de 500Mhz de cuatro procesadores con 1Gb RAM y no se ha detectado una degradación de rendimiento en una prueba de estrés con 15 clientes simultáneos. Mientras que la carga del servidor tuvo un incremento menor y despreciable. Esta prueba nos permite asegurar que cualquier cliente que quiera sumarse a nuestra red de VPNs podrá usar una máquina “vieja” como concentrador VPN y no verá afectado el rendimiento de la red. Por otro lado, como concentrador central de VPNs hemos planificado usar dos servidores virtuales de 4 procesadores de 2600Mhz y 6Gb RAM cada uno para atender los mil quinientos concentradores remotos. Se espera que esta configuración sea suficiente y no afecte el rendimiento de red.

Facilidad de configuración

OpenVPN tiene una configuración por defecto simple y fuerte que encaja prácticamente para la mayoría de las implementaciones [2]. Pero además, provee una flexibilidad que permite adaptarlo a cualquier requerimiento manteniendo los niveles de seguridad. Para poder cumplir con

los requerimientos, nos apoyaremos mucho en desarrollo de scripts propios.

Balanceo de carga

Para alcanzar este requerimiento se puede usar dos alternativas: el balanceo de carga a través de reglas IPTables en el servidor OpenVPN o simplemente desde la configuración del OpenVPN cliente mediante la definición de varias alternativas de conexiones a servidores remotos y la opción *remote-random*. Las reglas de las tablas NAT de IPTables nos permiten realizar un balanceo con comportamiento round-robin [8], mientras que la opción de realizar el balanceo en el cliente OpenVPN la rotación es aleatoria [9].

Alta disponibilidad

OpenVPN no provee esta funcionalidad de forma nativa, por lo que ya tenemos previsto la configuración de un clúster de servidores gestionados por Pacemaker y Corosync que proveen Cluster Resource Manager (CRM) y Heartbeat [10].

Gestión centralizada

OpenVPN no provee ninguna herramienta de gestión centralizada de clientes, por este motivo el presente proyecto prevé el desarrollo de una aplicación de gestión centralizada que tenga un módulo de gestión de clientes que permita la generación y almacenamiento seguro de certificados clientes, la revocación y renovación de estos certificados. Esta aplicación también brindará información muy importante para el soporte remoto de la conectividad.

Análisis y Diseño

Topología de red

Debido a que las redes remotas son ajenas a la organización no es posible definir ninguna especificación al respecto. Por este motivo se buscó definir una configuración de red que sea lo más flexible posible y que pueda adaptarse a cualquier configuración de red remota.

Para conseguir esta flexibilidad se decidió utilizar un cliente VPN por puesto de trabajo, esta decisión no sólo nos permite asignar una IP definida por nosotros a cada puesto para poder identificar unívocamente sus peticiones, sino que también nos protege de posibles filtraciones y configuraciones de seguridad deficientes que puedan existir en las redes de los clientes.

Cada puesto de trabajo se conectará con un túnel VPN a un concentrador VPN que estará en la misma red de trabajo. Para conocer en qué IP de la red se encuentra el concentrador VPN los puestos de trabajo consultarán una zona de DNS destinada a registrar las IPs privada de la interfaz física de cada concentrador VPN. Dicho concentrador VPN tendrá dos procesos de OpenVPN, uno servidor para atender a los puestos de trabajo y otro cliente que se conectará contra la organización de registro para brindarle la conectividad a los puestos locales. El concentrador VPN ruteará los paquetes de un túnel a otro realizando las operaciones de descifrado y cifrado constantemente. Ningún dispositivo que no tenga un cliente VPN podrá rutear paquetes a través del concentrador VPN.

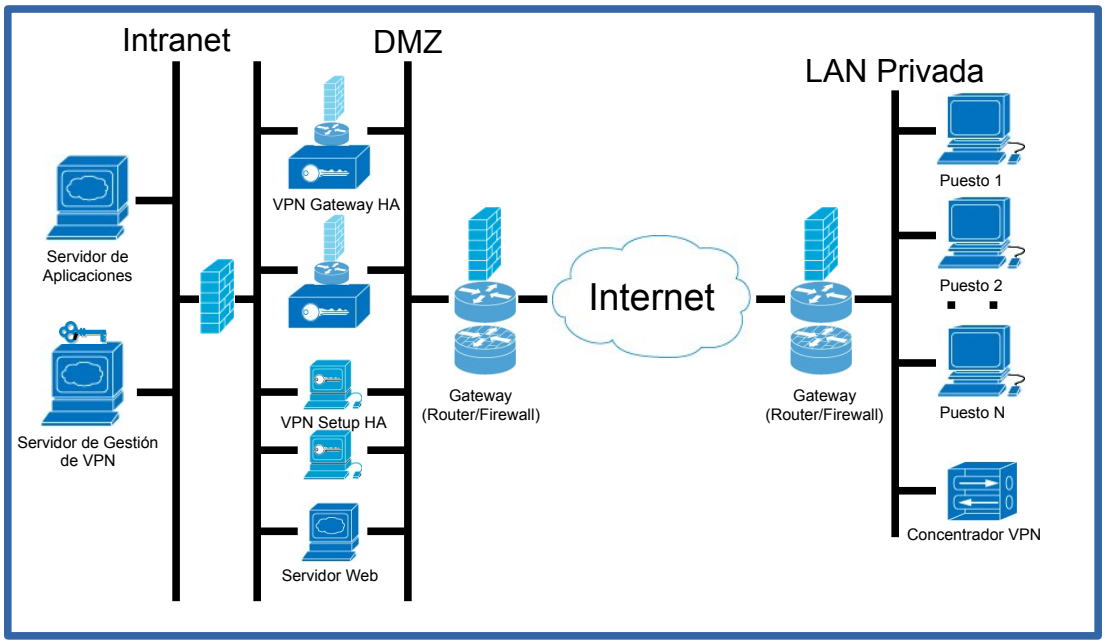
El servidor central de VPN ubicado en la organización estará detrás de un firewall que administra los diferentes enlaces de cada Proveedor de Internet (ISP). Tendrá por lo tanto nateados ciertos puertos de una IP pública de cada enlace hacia los puertos de servicio de OpenVPN. De igual manera, el servidor que brindará el servicio de instalación escuchará nateado en un puerto de una IP pública de cada enlace. Este servidor central de VPN será el encargado de actualizar la zona de DNS que guarda las IPs privadas de

cada uno de los concentradores VPN.

En la DMZ de la organización también está ubicado el servidor web con la página institucional de la organización. Este servidor web será el usado para publicar en él, ocasionalmente y bajo demanda, los archivos necesarios para la instalación de los clientes.

Dentro de la red Intranet de la organización, ubicada detrás de otro firewall se encuentra la aplicación web de registro. Debido a que ésta es el núcleo del negocio están definidos para este servidor los más altos niveles de seguridad. Dentro de esa red de servidores estará ubicado el servidor de Gestión de VPNs con el cual se dará soporte y se crearán nuevos clientes.

A continuación se muestra un diagrama esquemático de la topología de red definida:



Estructura de datos

De cada cliente se almacenarán diferentes variables, estos datos estarán guardados en una base de datos *MySQL*. A continuación se brinda un listado de las variables y una explicación de qué se guardará en cada una:

- *idvpn*: Identificador de la VPN contendrá 3 letras, 4 números y un

código verificador.

- *startup*: Indicará si el cliente fue instalado o no.
- *ip_tun*: La IP que toma la interfaz VPN que conecta contra el servidor VPN central de la organización.
- *cmdclavehw*: Contendrá el comando usado para comprobar el hardware del concentrador VPN.
- *clavehw*: Resultado de ejecutar el *cmdclavehw* en cada concentrador VPN.
- *clavehwmd5*: El hash MD5 del campo *clavehw*.
- *shadowmd5*: Un hash MD5 obtenido a partir de los archivos */etc/passwd* y */etc/shadow* y de las reglas de firewall de cada concentrador VPN.
- *clicacrt*: El certificado de la CA que usará el VPN cliente de cada concentrador VPN para conectarse. Comúnmente esta CA será igual en todos los clientes, salvo que se esté realizando un cambio paulatino de certificados.
- *clicertcrt*: El certificado correspondiente al cliente VPN de cada concentrador VPN.
- *clicertkey*: La clave privada del certificado del cliente VPN de cada concentrador VPN.
- *clitakey*: La clave precompartida usada para cifrar la conexión desde el concentrador VPN al VPN central, durante la fase de negociación del túnel SSL.
- *srvcacrt*: Será el certificado de la CA usado en el servidor VPN del concentrador VPN.
- *srvcertcrt*: El certificado firmado por la CA anterior, que usará el servidor OpenVPN del concentrador VPN.
- *srvcertkey*: La clave privada del certificado anterior.
- *srvtakey*: La clave precompartida usada para cifrar el inicio de conexión entre los puestos de trabajo y el concentrador VPN, durante

toda la fase de negociación del túnel SSL.

- *srvdh*: Contiene los parámetros Diffie-Helman que usará el servidor VPN de cada concentrador VPN.
- *subred*: Inicio del rango de red en el que se encontrarán los puestos de cada concentrador VPN.
- *mascara*: La máscara de red del rango anterior.
- *preexec*: Opcionalmente guardará comandos que serán ejecutados antes de instalar el servidor.
- *postexec*: Opcionalmente guardará comandos que serán ejecutados después de la instalación y antes de reiniciar el servidor.
- *fecha*: La fecha y hora de la última instalación.
- *latitud*: Latitud en formato decimal de la ubicación del cliente.
- *longitud*: Longitud en formato decimal de la ubicación del cliente.
- *estado*: Indicará si el sitio cliente está conectado, desconectado, en proceso de conexión, si falló alguna validación o si aún no fue instalado.
- *ubicación*: Una descripción de donde se encuentra el cliente.
- *nombre*: Título identificador del cliente.
- *dirección*: Dirección postal del cliente.
- *teléfono*: Teléfono del cliente.
- *contacto*: Nombre del contacto técnico del cliente.
- *versión*: Opcionalmente se usará si deben aplicarse parches en los concentradores para saber que parches tiene instalado y cuáles no.
- *ip_local*: Posterior a la instalación guardará la IP física del concentrador VPN, ésta será usada para actualizar el DNS que consultarán los puestos de trabajo.
- *máscara_local*: Máscara de red de la interfaz física de red del concentrador VPN.
- *gw_local*: IP de la puerta de enlace de la interfaz física de red del concentrador VPN.

- *server*: Identificador del servidor al que hizo la última conexión.
- *zona*: Podrá tener tres valores: NACIONAL, REGIONAL o INTERNACIONAL.
- *licencia*: Código aleatorio de 3 grupos de 4 alfanuméricos que se usarán para validar la instalación.

La estructura de datos resultante de la base de datos se muestra en el Anexo 1. Como se puede ver existe una tabla con gran cantidad de campos que contiene todos los datos relevantes de cada cliente. Debido a que esta tabla contiene los certificados y las credenciales de cada uno de los clientes, se han implementado rigurosas políticas de seguridad para el acceso a estos datos.

Algo importante de destacar es que se intentó realizar un desarrollo seguro, para lo cual cada parte de la aplicación tiene un usuario y contraseña de conexión a la base de datos diferente, cada uno con sus propios privilegios. Debido a que una filtración de datos de esta tabla comprometería la seguridad de todas las VPNs y obligaría a la revocación de todos los certificados ahí almacenados, se decidió implementar el uso de vistas que muestren sólo una porción de los campos de la tabla de modo que únicamente un usuario con permiso de *SELECT* en la tabla *clivpn* podrá leer los campos que contienen los certificados. De este modo, cada aplicación o módulo tiene acceso a los campos de datos que son estrictamente necesarios y nada más, reduciendo así los riegos de una filtración de datos debida a una debilidad de programación de las aplicaciones y módulos. Las vistas que serán usadas, los usuarios y sus privilegios se encuentran al final del Anexo 1.

Proceso de alta de nuevo cliente

Una de las tareas más importantes será la posibilidad de dar de alta nuevos clientes. Durante este proceso se generarán todas las claves que estarán almacenadas en el concentrador VPN y una para cada uno de los

puestos de trabajo que inicialmente tenga dicho cliente. También se definirán parámetros de configuración del servidor OpenVPN del concentrador VPN, como por ejemplo, el rango de red que usará ese servidor para asignar direcciones a sus puestos de trabajo.

El proceso recibirá los datos mínimos del cliente, cantidad de puestos iniciales y zona donde está ubicado el cliente. El inicio del proceso generará un identificador de VPN para el cliente y le calculará el dígito verificador. También asignará un rango de red eligiendo el siguiente rango de 256 direcciones (clase C) disponible. Luego generará aleatoriamente una licencia de instalación que consta de tres grupos separados por un guión de cuatro alfanuméricos cada uno. Posteriormente se procederá a generar las claves SSL.

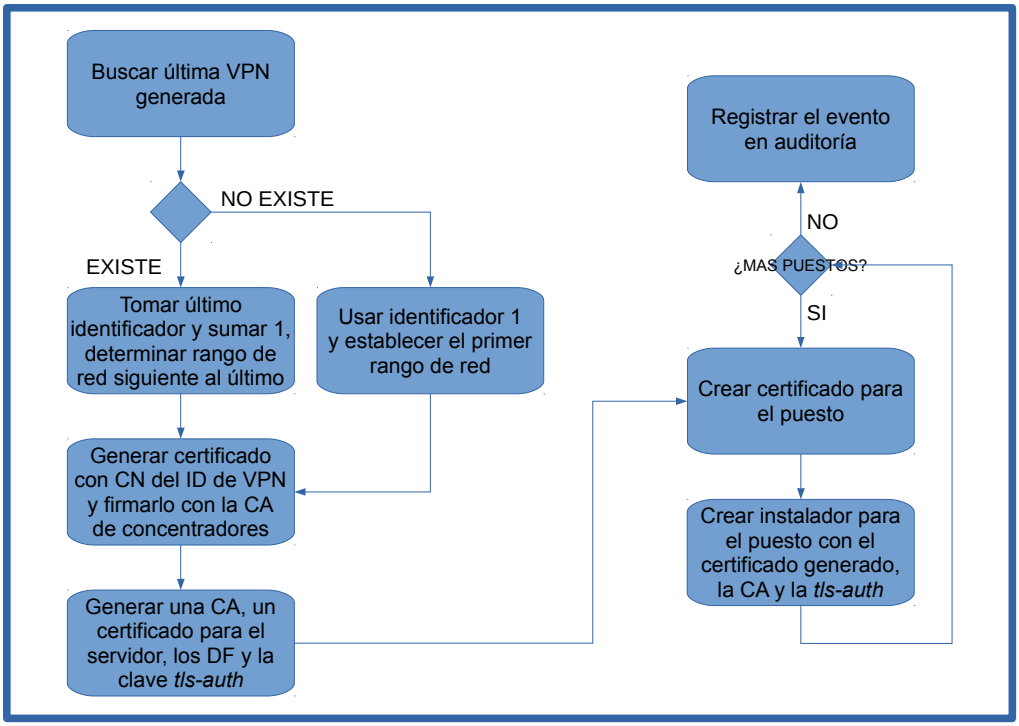
Lo que resta del proceso de creación de claves SSL podría dividirse en dos etapas, una por cada túnel VPN. Para el túnel que conecta al concentrador VPN contra nosotros se generará un certificado (y su clave privada). Este certificado será firmado por la Autoridad Certificante (CA) de concentradores VPNs.

La segunda etapa es durante la cual se generarán y almacenarán todos los parámetros necesarios para realizar el túnel que unirá los puestos de trabajo con el concentrador VPN. En esta etapa se generará una Infraestructura de Clave Pública (PKI) completa, ya que comenzará por generar un par de claves pública y privada el cual se convertirá en un certificado autofirmado que será la CA usada para firmar y validar todos los certificados del túnel entre los puestos de trabajo y el concentrador VPN. Luego se generará un certificado para el servidor VPN firmado por dicha CA. Ambos certificados y la clave privada del certificado del servidor serán almacenados en la base de datos. Adicionalmente se generarán los parámetros Diffie Helman y la clave pre compartida *tls-auth* que será usada durante la negociación de los túneles VPN.

Una vez almacenados todos los datos, parámetros y archivos necesarios para iniciar el servidor en el concentrador VPN, se generarán los

instaladores de los VPN de los puestos de trabajo de plataforma Windows. Para esto se generarán tantos certificados como puestos iniciales hayan sido solicitados. Estos certificados serán firmados por la CA recientemente creada. Luego mediante el uso de la herramienta *nsis* se creará un instalador del OpenVPN GUI para Windows para cada puesto. Cada instalador tendrá incorporado su certificado correspondiente, el certificado de la CA recientemente creada, la clave pre compartida *tls-auth* y la configuración del OpenVPN cliente.

Al finalizar el proceso de generación del nuevo cliente se registrará el evento de alta en la tabla de auditoría registrado el usuario del operador que realizó la tarea.



Procedimiento de instalación de un cliente

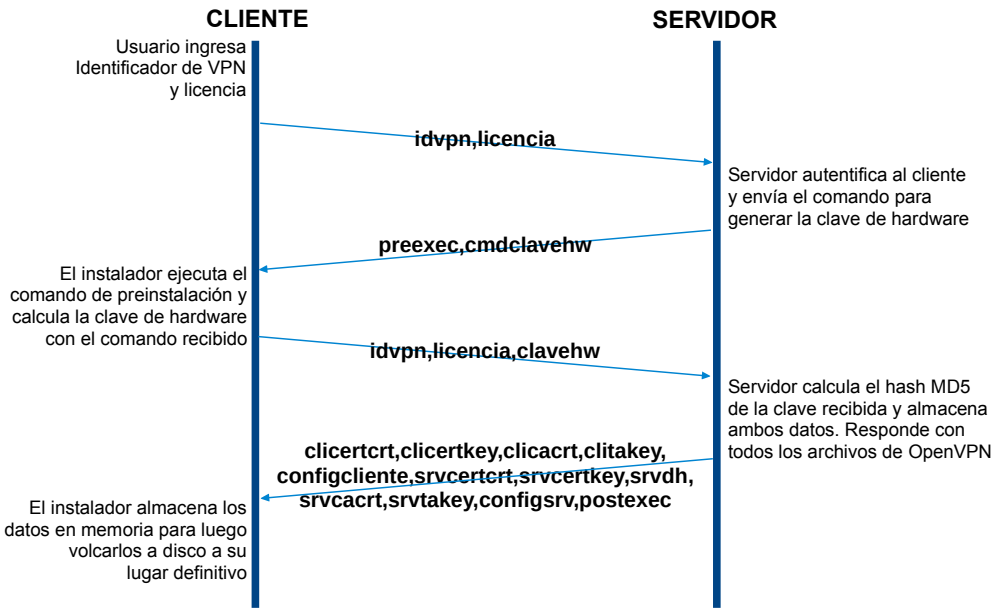
La instalación de un cliente comenzará por la configuración y puesta en marcha de su concentrador VPN. Para esta tarea se aprovisionará un CD booteable, que al correrlo mediante un asistente de instalación solicitará al operador los parámetros de configuración de la red. Para ello primero elegirá

si quiere configurar por DHCP o no. En caso afirmativo, el instalador intentará obtener la configuración de red mediante DHCP si es satisfactorio guardará la configuración y realizará una prueba de conectividad contra el VPN central, en caso exitoso seguirá al segundo paso, en caso contrario volverá a empezar. Si elige configurar manualmente (no por DHCP) se solicitará que ingrese: Dirección IP, Máscara de red, Puerta de enlace y Dirección IP del servidor DNS. Al finalizar la introducción de estos parámetros los mismos serán guardados y se procederá a probar la conectividad contra el VPN central. En caso exitoso seguirá con el segundo paso, en caso contrario volverá a empezar.

Una vez configurada la red y habiendo realizado la prueba de conectividad exitosamente, se solicitará el identificador de la VPN y su licencia correspondiente. Este par de datos viajará por HTTPS hacia el servidor de instalación, el cual validará los datos. Si la respuesta es satisfactoria el asistente de instalación ejecutará los comandos de preinstalación que reciba en la respuesta.

Posteriormente el instalador internamente calculará las claves de hardware usando el comando recibido para ello y enviará el resultado junto con el identificador de VPN y la licencia. En la respuesta, recibirá todos los archivos de configuración del OpenVPN para conectar como cliente contra el VPN central y también los necesarios para levantar el servidor de VPN local.

Esquemáticamente el protocolo de comunicación en la instalación se podría ver así:

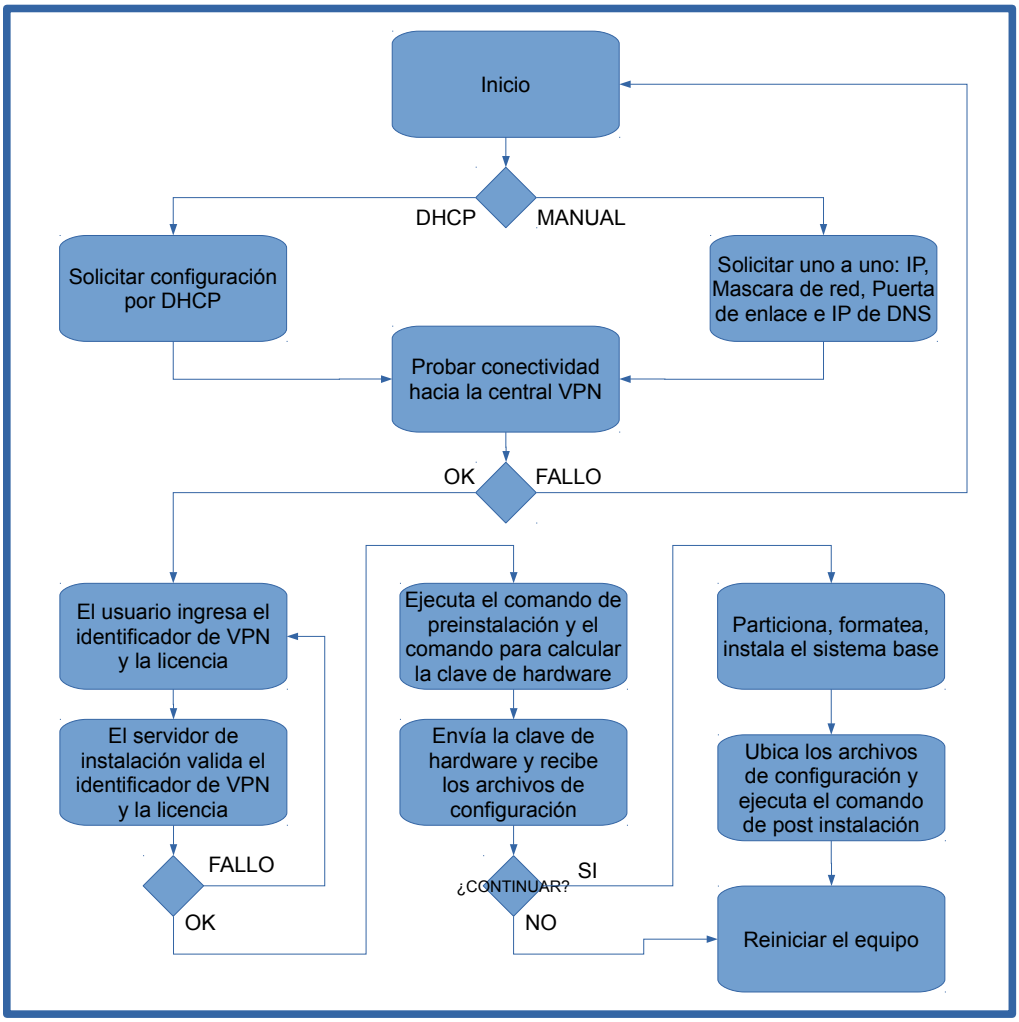


A continuación, el asistente alertará al usuario que, para completar la instalación, se eliminará toda la información que pueda tener la PC donde se está ejecutando la instalación. Si el usuario confirma que desea continuar, el sistema de instalación borrar la tabla de particiones del disco primario de la PC, creará una tabla de particiones con una única partición, le hará un formato con sistema de archivos *ext4*, lo montará y descomprimirá un empaquetado que contendrá un sistema operativo Linux Debian estándar con los paquetes necesario preinstalados. Dentro de esta partición generará un archivo *loop* que será configurado como memoria de intercambio (*swap*) para el linux. Y finalmente actualizará la configuración *grub* para que éste pueda bootear

Antes de reiniciar, el proceso de instalación volcará los archivos de configuración de OpenVPN recibidos, junto con las claves y demás parámetros para que los servicios inicien junto con el sistema. También guardará la configuración de red configurada durante la instalación en el sistema instalado. Finalmente previo a reiniciar el sistema ejecutará los comandos recibidos de postinstalación.

Del lado del servidor de instalación, autentifica todas las conexiones con el par de datos identificador de VPN y licencia, y dependiendo del contenido de la petición reconoce si es el primero o el segundo paso. El

primer paso es trivial ya que sólo lee los campos *preexec* y *clavehw* y los envía. El segundo paso calcula el hash MD5 de la clave de hardware recibida y almacena tanto la clave como el hash. Luego, recolecta todas las claves, configuraciones y parámetros para enviar como respuesta. Al finalizar este proceso registra el evento relacionado en la tabla de auditoría.



Procedimiento de habilitar reinstalación de un cliente

Este procedimiento brinda a los operadores de soporte la posibilidad de autorizar la reinstalación de un concentrador VPN. O sea, un cliente con problemas de conectividad en el concentrador VPN podrá reinstalar el equipo luego de ser autorizado. El operador tendrá dos opciones, solamente habilitar la reinstalación que implicará cambios en la base de datos; o en

caso de haber sido comprometidos los certificados podrá optar por la revocación de los mismos y su regeneración.

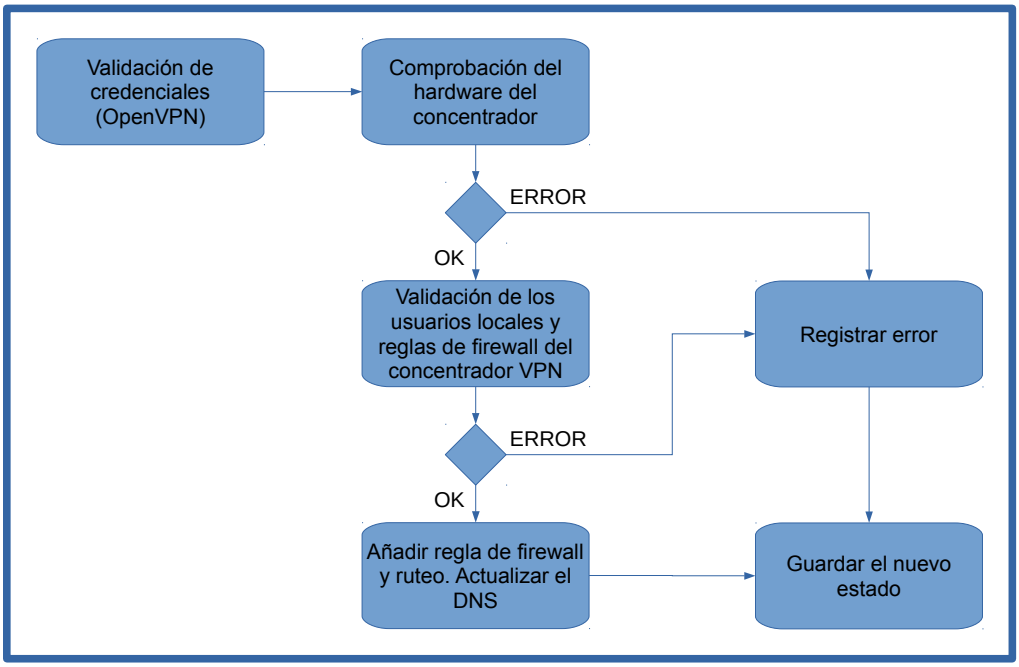
Finalizada la operación el proceso registrará el evento en la tabla de auditoría registrando el usuario del operador que realizó la acción.

Proceso de conexión

Cuando un concentrador VPN inicie una conexión hacia la organización, el servidor central de OpenVPN validará sus credenciales, o sea sus certificados. Este proceso es propio de OpenVPN y lo hace durante la negociación del túnel SSL. Si el concentrador VPN emplea certificados válidos para la conexión, el túnel quedará establecido. De cualquier manera aún no podrá realizar peticiones, debido a que el firewall en el servidor central de VPN no se lo permitirá.

Haciendo uso del túnel ya establecido un proceso de validación comprobará, que el hardware instalado en el concentrador VPN es el mismo que fuera registrado al momento de la instalación. Si la validación anterior es exitosa, se comprobará que las credenciales de usuarios locales y las reglas de firewall almacenadas en el concentrador VPN no hayan sido alteradas respecto a las registradas durante la instalación del mismo. En caso de que alguna de las comprobaciones falle se registrará el evento y se actualizará el estado del cliente para reflejar el problema.

En caso de que los chequeos realizados hayan sido exitosos el proceso anterior agregará una regla de firewall que permita el acceso desde los puestos de trabajo del cliente al servidor de registro. Además, se añadirá la ruta a la red de VPNs de los puestos del cliente en el ruteo. Luego, usando la IP de LAN de la interfaz del concentrador VPN, se actualizará el registro DNS que consultarán los puestos de trabajo para poder conectarse. Finalmente, se registrará el evento de conexión del cliente reflejando su nuevo estado en la base de datos. A continuación se muestra un diagrama de flujo del proceso:



Proceso de desconexión

Cuando el OpenVPN debe desconectar un cliente, cualquiera sea el motivo: éste da de baja el túnel establecido, se produce por inactividad en el túnel o simplemente recibe la orden de matar una conexión; el servidor dispara un proceso muy sencillo que hará tres cosas:

- Borrar la regla de firewall (si la hubiese) que permitía conectarse a los puestos del concentrador VPN que acaba de desconectarse.
- Borrar la ruta hacia los puestos de trabajo de dicho concentrador VPN.
- Registrar el evento y el nuevo estado del cliente.

Funciones de soporte para validación de un cliente

En caso de que un concentrador VPN falle las comprobaciones posteriores a la conexión. Ya sea que hayan cambiado algo del hardware de concentrador VPN, que una eventual instalación de un software en el concentrador haya creado nuevos usuarios o que se hayan agregado reglas de firewall, se ofrecerán herramientas de soporte para que los operadores

del panel de Gestión de VPNs puedan realizar el registro de los nuevos valores que permitan al cliente seguir operando sin tener que reinstalar el servidor.

Estas funciones utilizarán el túnel ya establecido para recuperar el nuevo valor de la comprobación que falló, luego almacenarán el valor en la base de datos y forzarán la desconexión del cliente para que éste reinicie el proceso de conexión y comprobación. Todas estas operaciones dejarán registrado el evento en la tabla auditoría y dicho registro incluirá el usuario del operador que realizó la tarea.

Funciones de descarga o publicación de credenciales

El módulo de Gestión de VPN ofrecerá la función de descargar los medios digitales necesarios para la instalación de un cliente. De esta manera, permitirá a un operador del módulo descargar el *ISO* del CD de instalación de un cliente para grabarlo, o descargar uno o más instaladores de puestos de trabajo de un cliente en un archivo tipo *RAR*. Adicionalmente dará la posibilidad de publicarlo en la página web institucional de la organización para que el cliente pueda remotamente acceder a los medios y hacer uso del mismo.

La publicación se realizará en un servidor de público acceso por lo cual se deberán arbitrar los medios y las acciones necesarias para restringir el acceso a los archivos sólo a la persona indicada.

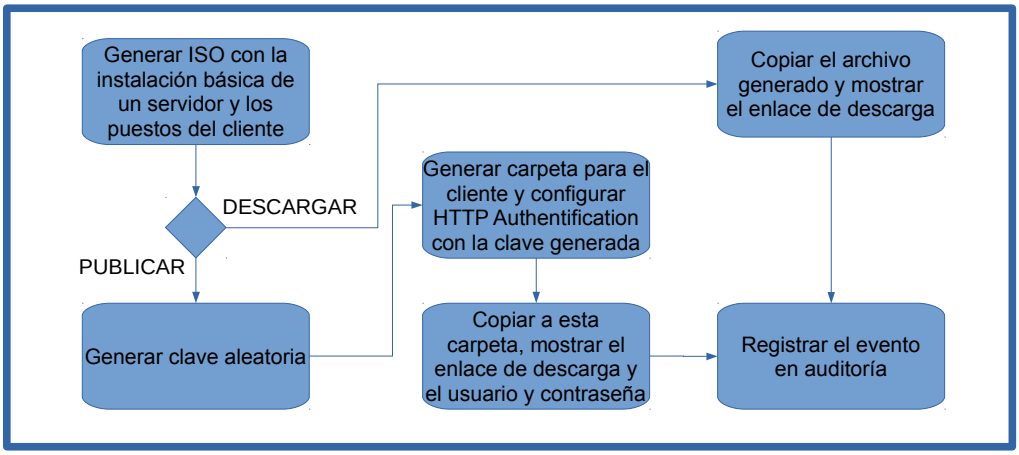
Descarga o publicación de un CD de instalación

En caso de tratarse de la instalación completa de un cliente esta función generará el *ISO* del CD de instalación, correspondiente al cliente especificado recopilando la instalación básica de un servidor y los instaladores VPN para los puestos de trabajo de dicho cliente. Adicionalmente el operador tendrá dos opciones, descargarlo desde el módulo de Gestión de VPNs o publicarlo en la página web institucional de la

organización bajo la protección de usuario y contraseña.

Si se decide descargar desde el módulo de Gestión de VPNs el archivo en cuestión será ubicado en una carpeta de dicho servidor y se mostrará el enlace al mismo. En caso de haberse solicitado publicar en el sitio institucional, se creará una carpeta en éste último, dedicada especialmente para el cliente. Se generará una clave aleatoria y se implementará el método *HTTP Authentication* para restringir el acceso sólo al usuario dueño de CD de instalación. Luego se copiará el archivo a la carpeta en cuestión y se mostrará en pantalla el usuario y la contraseña que deberán ser trasmitidos al contacto técnico que realizará la descarga.

En cualquier caso, ya sea descarga o publicación el proceso registrará el evento en la tabla de auditoría almacenando el usuario del operador que realizó la tarea.

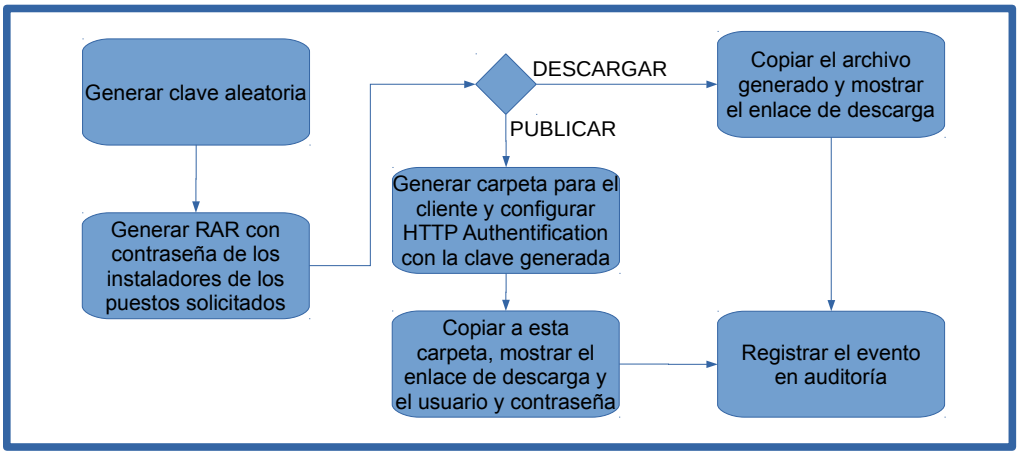


Descarga de instaladores de puestos de trabajo de un cliente

Si lo solicitado fue uno o más instaladores de VPN para puestos de trabajo de un cliente, el módulo de Gestión de VPN generará un archivo *RAR* que contenga los instaladores seleccionados cifrados con una clave aleatoria. Este cifrado es realizado debido a que en la actualidad, la mayoría de servidores *SMTPs* no permiten el envío de archivos ejecutables ya sean que viajen solos o empaquetados en un *RAR*, *ZIP*, *TAR*, etc. De cualquier modo existen servidores *SMTPs* que tampoco admiten archivos cifrados (por

ejemplo gmail.com), por este motivo, la aplicación ofrecerá la alternativa de publicar el archivo en el servidor web institucional, protegido con usuario y contraseña, de manera análoga a como lo hace con los archivos *ISOs*.

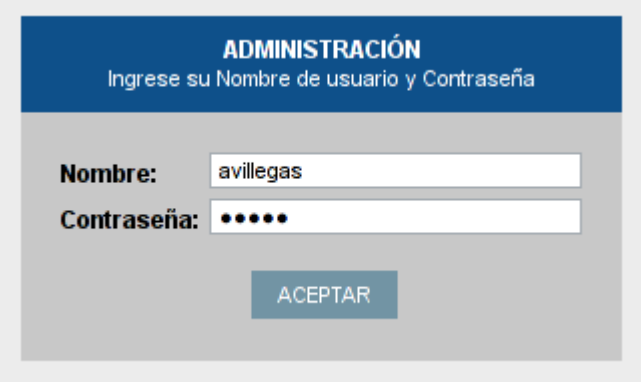
Al finalizar la operación este proceso registrará el evento en la tabla de auditoría almacenando el usuario del operador que realizó la tarea.



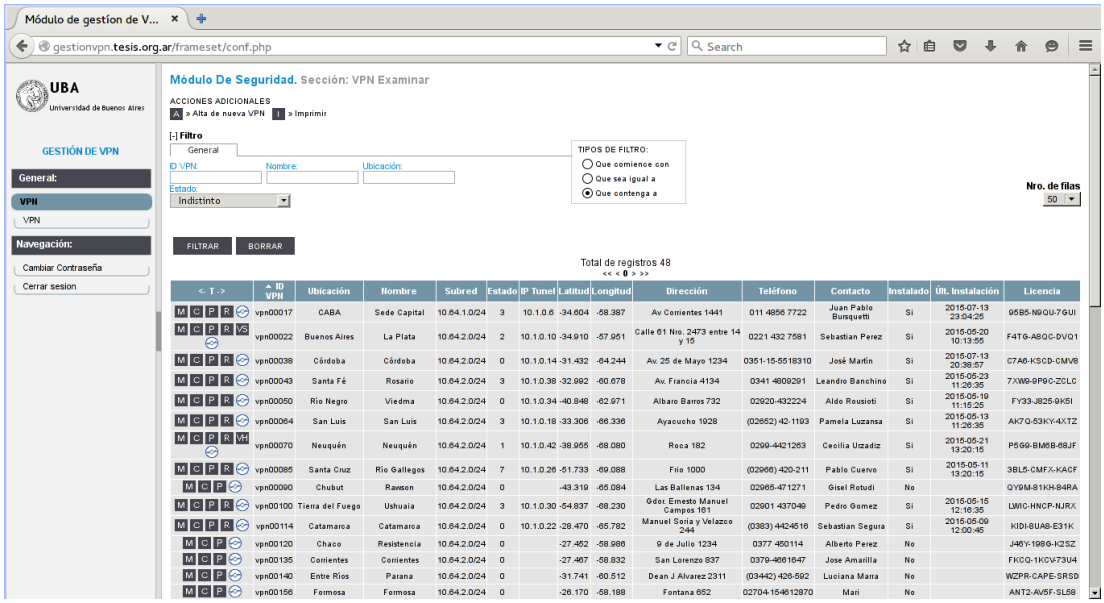
Desarrollo e implementación

Aplicación de gestión de clientes

La interfaz web de gestión de VPN fue desarrollada en *PHP* y almacena los datos en una base de datos *MySQL*. Dicha interfaz requiere que el operador se autentifique mediante usuario y contraseña, en una interfaz similar a la siguiente:



Al pasar exitosamente la fase de autentificación se muestra en la pantalla un listado de los clientes VPN existentes, y unas opciones de filtro de búsqueda. Cada fila de la tabla muestra los datos más relevantes, para el soporte, de cada cliente y en la primera columna de la tabla una serie de botones que habilitan al operador a realizar diferentes tareas sobre un cliente en particular. Además de la tabla y las opciones de búsqueda, se muestran dos enlaces, uno para dar de alta un nuevo cliente y otro para imprimir un listado tipo reporte de todas las VPNs existentes hasta el momento.



Los botones que aparecen o pueden aparecer son los siguientes:

M: Nos permite cambiar los datos modificables de un cliente.

C: Nos permite consultar los eventos registrados sobre el cliente, visualizando todos los registros de la tabla auditoría relacionados con dicho cliente.

P: Nos muestra un listado de los puestos generados y nos permite generar más puestos, así como descargar un paquete de ellos o publicarlo en la web institucional para su descarga remota.

: Nos permite generar el *ISO* del CD de instalación del cliente. Ofrece descargarlo o publicarlo en la web institucional para su descarga remota.

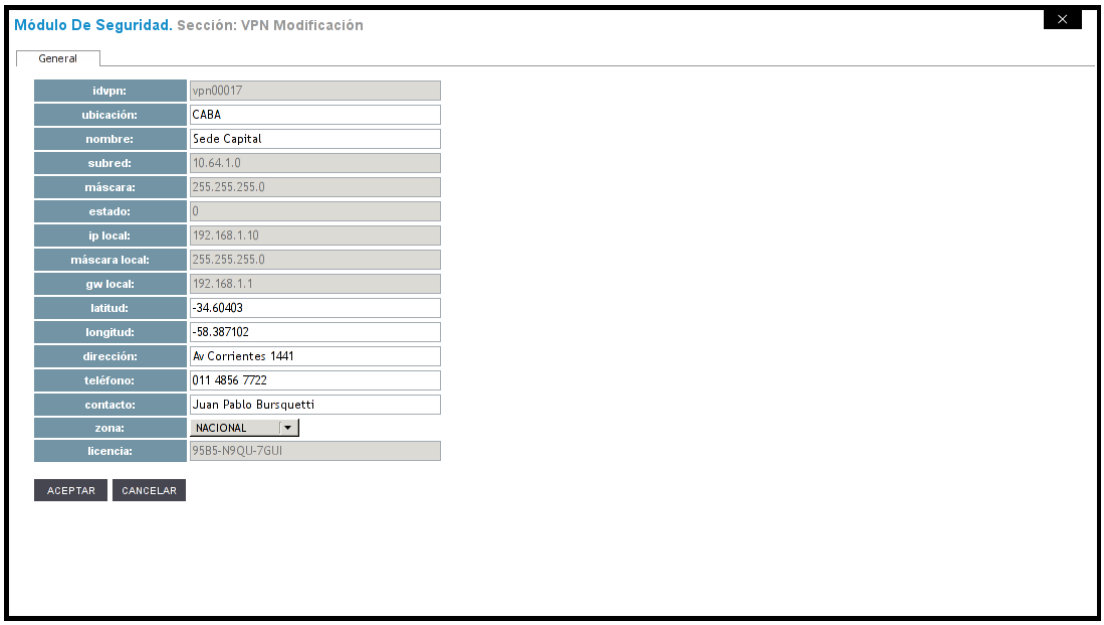
R: Botón opcional, sólo aparece si el cliente está instalado. Permite al operador habilitar la reinstalación del cliente. También ofrecerá la opción de revocar el certificado en cuestión y generarle nuevos certificados.

VH: Botón opcional, sólo aparece si el cliente está en estado 1, o sea, con problemas de validación de la clave de hardware. Permite al operador validar el hardware usado por el concentrador VPN del cliente.

VS: Botón opcional, sólo aparece si el cliente está en estado 2, o sea,

con problemas de validación del hash de los archivos de credenciales y las reglas de firewall del concentrador VPN. Permite al operador validar las credenciales de usuarios locales y las reglas de firewall instaladas en el concentrador VPN del cliente.

El botón para modificar M nos permitirá cambiar algunos de los atributos de los clientes, pero no todos. A continuación se muestra la interfaz para la modificación de un cliente:



Los atributos modificables son el nombre, la ubicación, la geolocalización (latitud y longitud), la dirección, el teléfono, el contacto y la zona. Al realizar una modificación se guardará el evento en la tabla de auditoría.

Generación de clientes y credenciales

Al hacer click en “Alta de nueva VPN”, nos muestra una pantalla como la siguiente:

Para continuar se deben ingresar:

- nombre: es la denominación que identifica de manera fácil al cliente.
- ubicación: nos permite ingresar un dato adicional sobre el lugar dónde está el cliente.
- cantidad de puestos: será el número inicial de puestos de trabajo que tendrá el cliente en cuestión.
- zona: El combo de selección nos permite elegir entre 3 posibles NACIONAL (para clientes dentro del país), REGIONAL (para cliente de Sudamérica/América) e INTERNACIONAL (para el resto del mundo). Este dato es simplemente de naturaleza estadística.

Una vez enviado este formulario completo se verifica que contenga valores del tipo de dato correcto y en caso afirmativo, realiza tres tareas:

- Llamar al script python /usr/local/bin/crearVPN.py con tres parámetros: nombre, ubicación y zona. El script retornará el identificador de VPN generado.
- Llamar al script bash /vpn/puestos/generar.sh pasando el identificador de VPN (recientemente recibido) y el número de puestos de trabajo inicial que fueron solicitados.
- Finalmente insertará el evento en la tabla auditoría.

Ambos scripts y otros relacionados pueden verse en el Anexo 2. Como se puede apreciar en el código, el script /usr/local/bin/crearVPN.py en su línea 80 llama a un script bash /vnp/ssl/cert.sh y le pasa un como parámetro el identificador de VPN generado. Este script genera el certificado del cliente del concentrador VPN y lo firma por la CA de concentradores VPN. También se puede ver en la línea 85 que llama a otro script ubicado en /vpn/puestos/crearca.sh el cual recibe un único parámetro, el identificador del VPN. Este último script genera la PKI del cliente. Todos los archivos de esa PKI quedarán guardados en la carpeta /vpn/puestos/keys/[ID VPN]/.

Ambos bash scripts (/vnp/ssl/cert.sh y /vpn/puestos/crearca.sh) y el otro llamado por el módulo de Gestión de VPNs para generar los puestos de trabajo del cliente (/vpn/puestos/generar.sh) utilizan un paquete de gestión de llaves RSA que provee OpenVPN conocido como *easy-rsa*. Este paquete provee varios scripts basados en OpenSSL que facilitan las tareas de gestión de llaves SSL RSA. [11]

En la línea 28 del script /vpn/puestos/generar.sh se usa la herramienta *makensis*, que es provista por el paquete *nsis* (Nullsoft Scriptable Installer System). Esta herramienta nos permite generar un instalador para plataforma Windows que incluya el OpenVPN GUI y además integre los certificados y la configuración de la conexión. [12]

CD y proceso de instalación

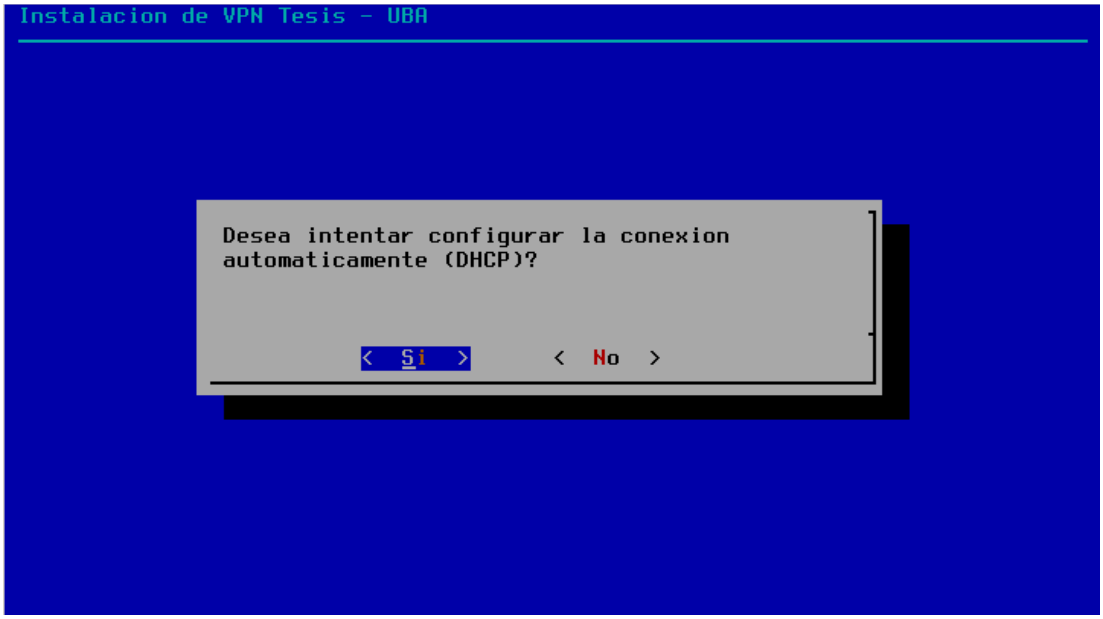
El servicio de instalación que brinda las configuraciones a los concentradores VPN está implementado sobre un servidor Web que atiende comunicaciones cifradas mediante el uso de SSL y que está configurado de manera que requiere un certificado firmado por una CA particular en el cliente para atenderlo. Esta medida de seguridad reduce el universo de potenciales atacantes al servicio.

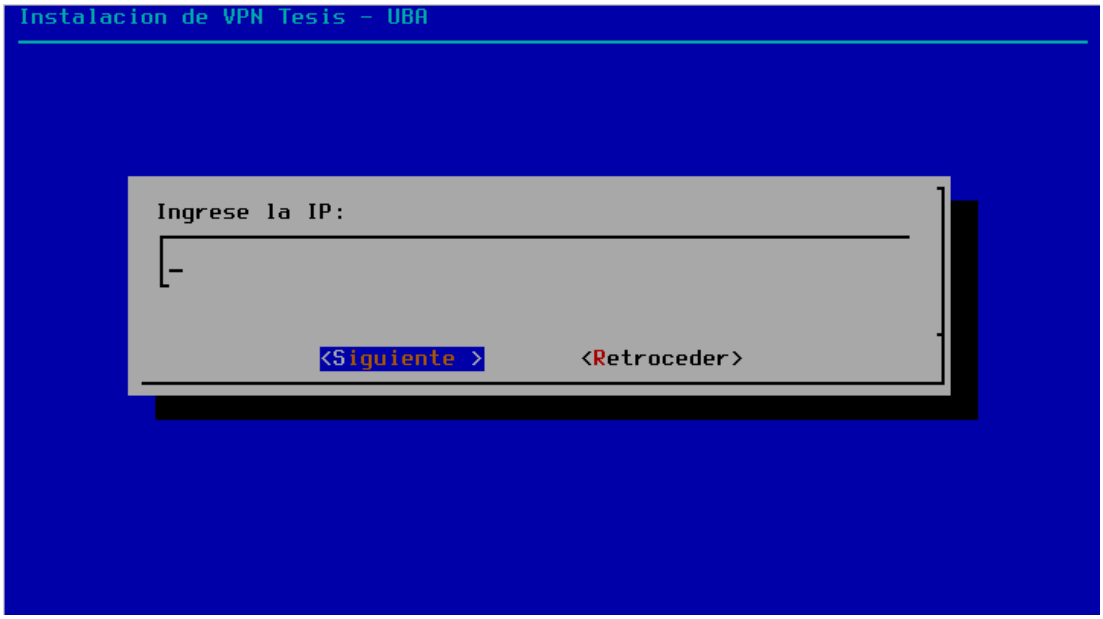
Específicamente el servicio es un *cgi-bin* que dispara un script hecho en Python que interactúa con el instalador, también desarrollado en Python, ubicado en los CDs de instalación. Para comunicarse ambos scripts utilizan

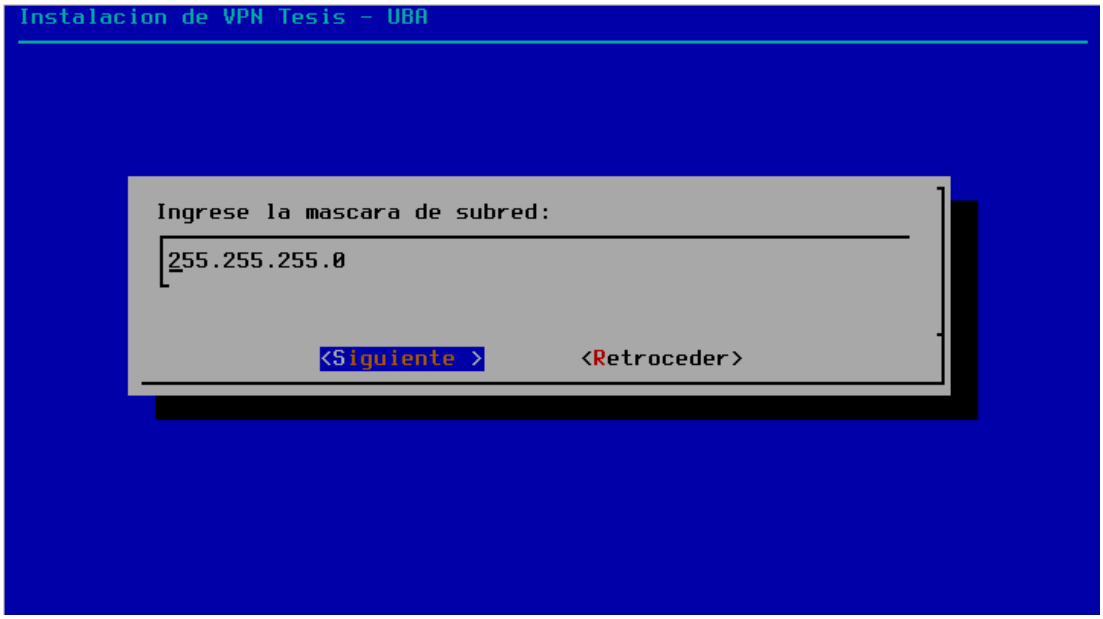
el protocolo HTTP sobre SSL. El instalador del cliente envía peticiones HTTP con sus variables de autentificación y respuestas a los mensajes del servidor usando peticiones HTTP de método POST. El servidor usa el cuerpo de la respuesta HTTP para devolver al cliente los posibles errores y las variables de configuración que requiere el cliente. Ambos scripts pueden encontrarse en el Anexo 3.

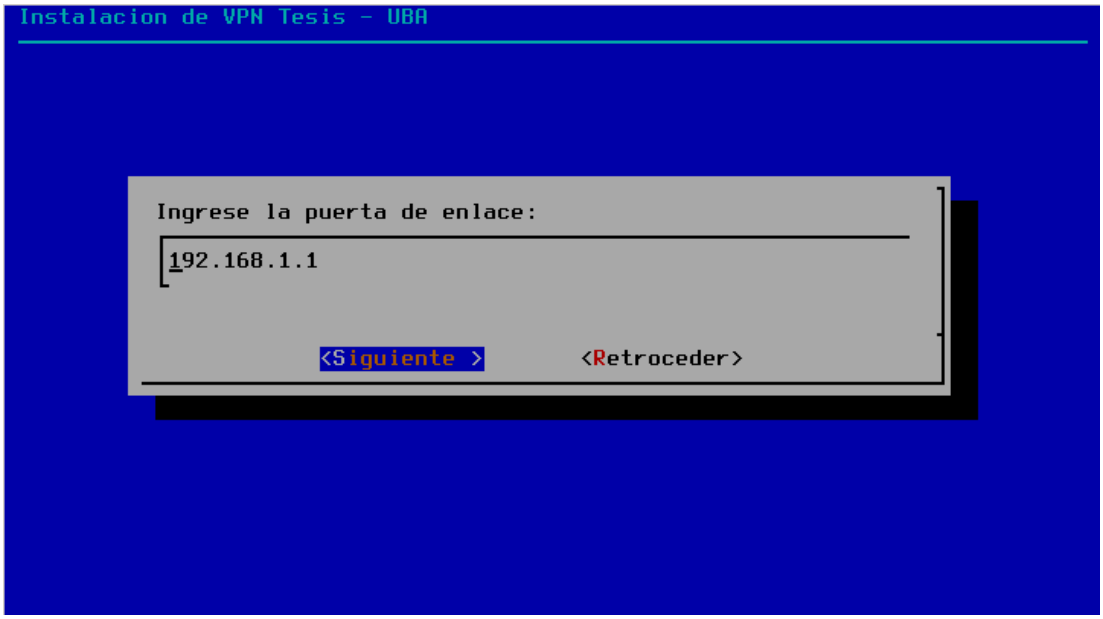
Como puede verse el script alojado en el CD es muy corto y todas sus funciones son ejecutadas mediante el uso de una librería Python (libvpn.py). Esto se debe a que como Python es un lenguaje interpretado se deseó ocultar lo más posible el código del instalador y lo que realmente viaja en el CD es la librería pre-compilada por Python (libvpn.pyc) y no su código fuente. Esta medida no es completamente efectiva ya que mediante ingeniería reversa es fácil volver al código fuente, pero una vez más, es una medida para achicar el universo de posibles atacantes.

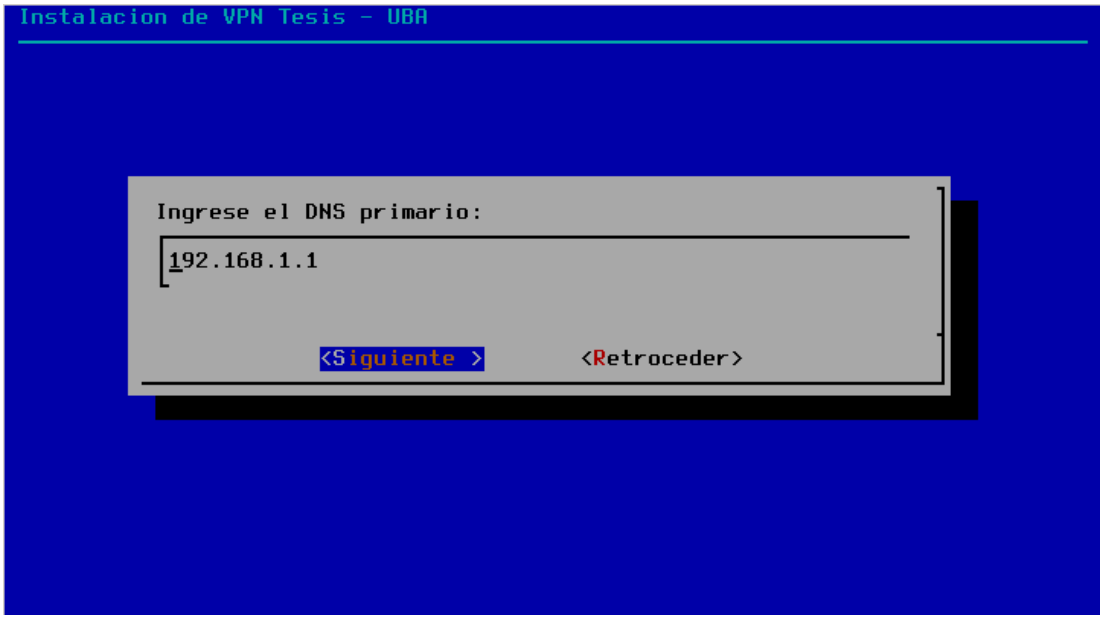
Al bootear con el CD, el instalador usando el comando *dialog,* interactúa con el usuario para obtener la configuración de red. En las siguientes imágenes puede verse como lo hace:



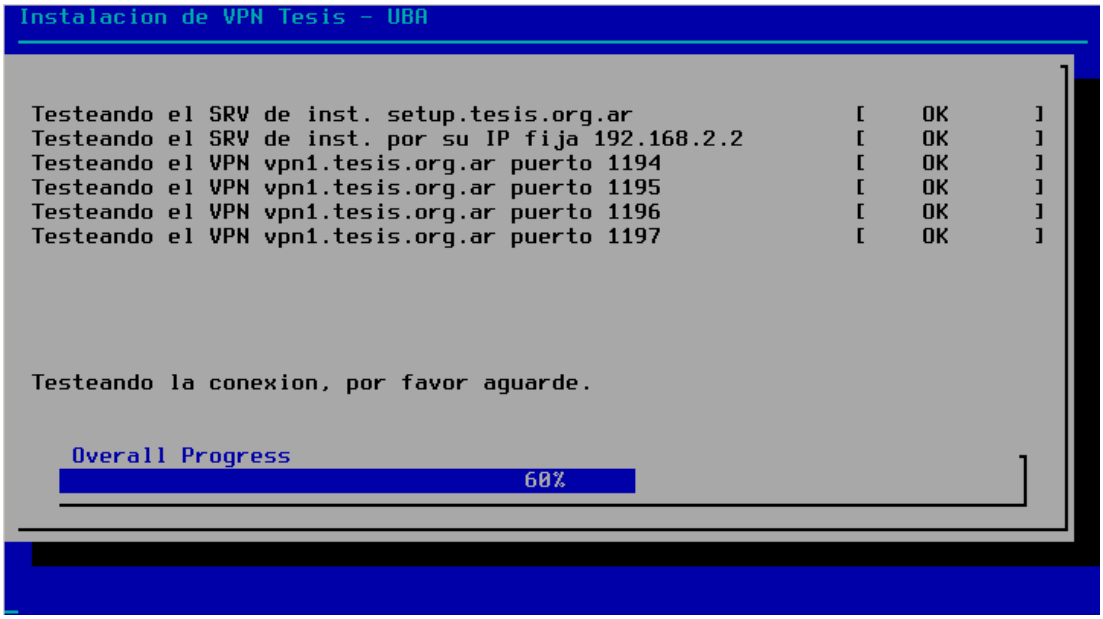

Instalacion de VPN Tesis - UBA
Ingrese la IP:
<Siguiente >
<Retroceder>


Instalacion de VPN Tesis - UBA
Ingrese la mascara de subred:
255.255.255.0
<Siguiente >
<Retroceder>

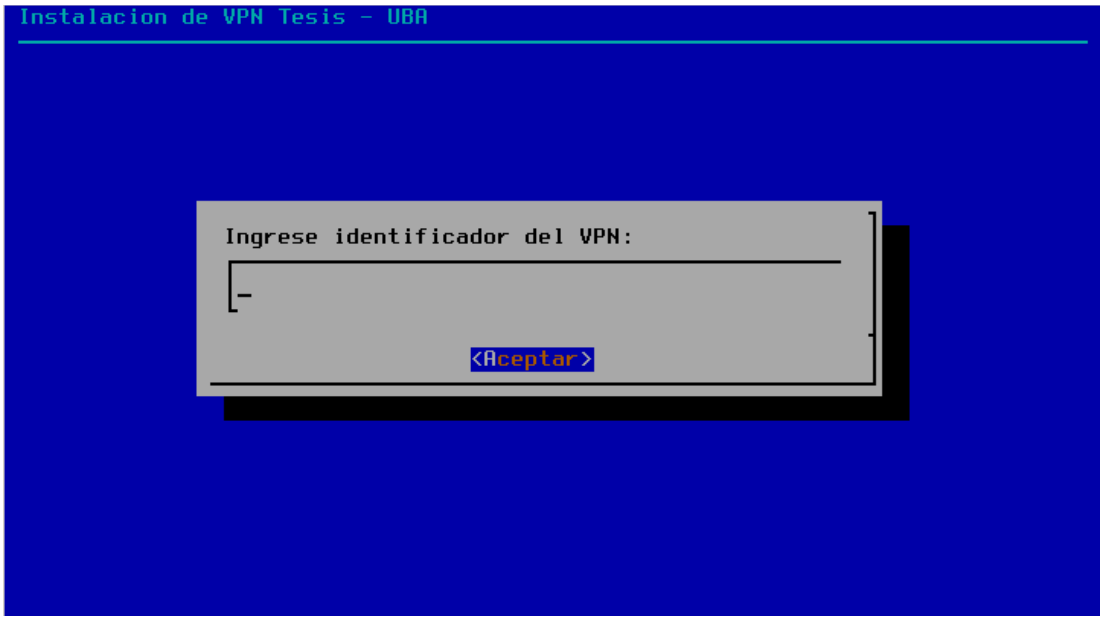




Una vez se tiene la configuración completa ésta es aplicada al sistema y se realiza una prueba de conectividad tanto hacia el Servidor Web de instalación como hacia cada uno de los puertos posibles de OpenVPN. Como se puede ver en la siguiente imagen:



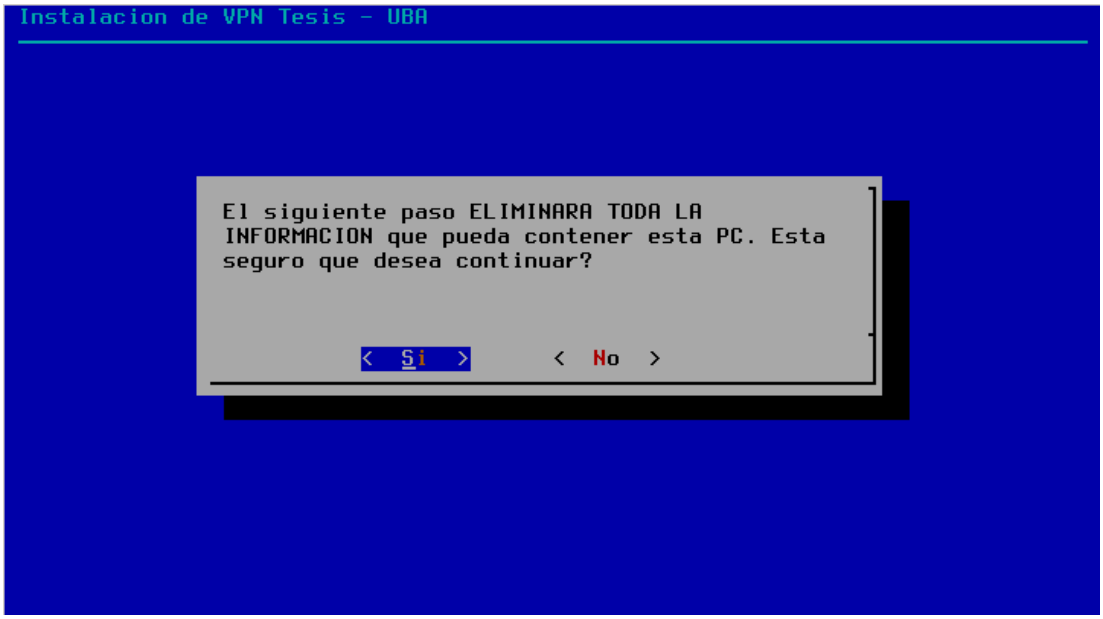
En caso de no llegar al Servidor Web de instalación o a ninguno de los puertos de OpenVPN se alertará de la situación y no se permitirá continuar con la instalación. Si por el contrario, supera la prueba de conectividad se solicita el identificador de VPN y la licencia correspondiente:



Luego mediante el uso del comando *wget* se envía al servidor estos datos. El *cgi-bin* (/var/www/cgi-bin/alta) evalúa si la petición incluye la clave de hardware, en caso no haber sido enviada sabe que es el primer mensaje de la instalación entonces valida el identificador de VPN y su licencia, y en caso de autentificarlo, lee de la base de datos los parámetros que debe enviarle al cliente (preexec y cmdclavehw).

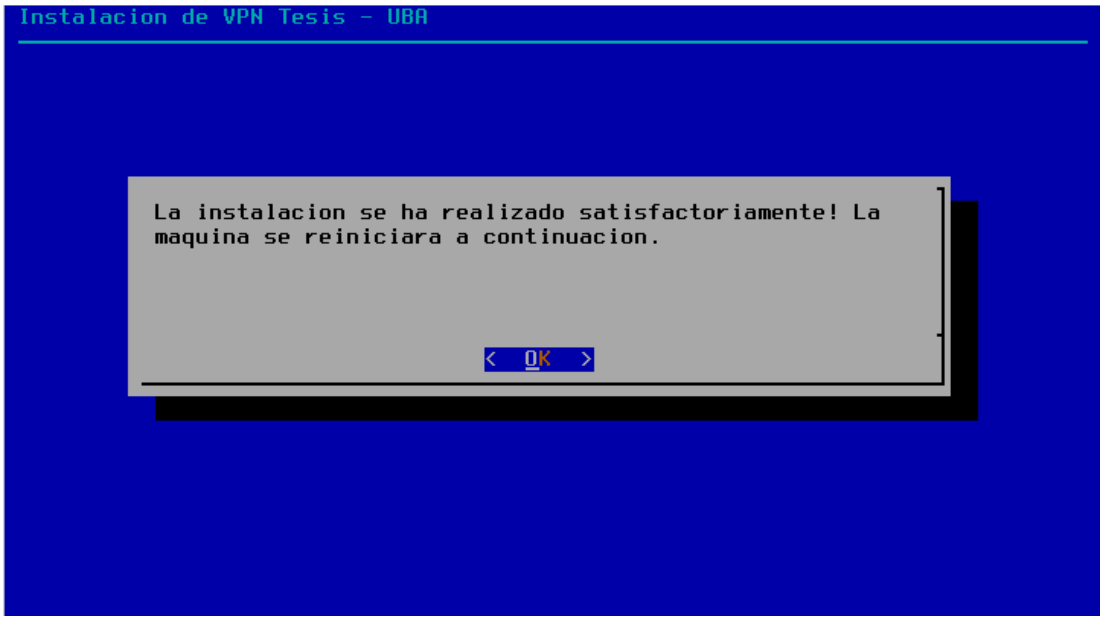
El cliente lee el contenido de la respuesta al *wget*, ejecuta ambos comandos y genera otra petición que incluye el resultado del comando de la clave de hardware. El servidor en esta ocasión lee la variable clavehw e interpreta que es el segundo mensaje de la instalación. En ese momento vuelve a validar el identificador de VPN y la licencia recibidos y en caso correcto almacena la clave de hardware y su hash MD5. A continuación, lee todos los parámetros de configuración que debe enviar, registra la acción en la tabla de auditoría y envía la respuesta.

El cliente una vez más lee el resultado obtenido del *wget* pero aún no lo guarda al disco. En este momento advierte al usuario que eliminará todo el contenido del disco rígido:



Si el usuario da su consentimiento entonces el instalador procede a particionar el disco, formatearlo para luego descomprimir una imagen de un Linux Debian Jessie preinstalado. Posteriormente realiza un *grub-install* para modificar el *Master Boot Record (MBR)* del disco, lo que permitirá bootear el Linux instalado. Finalmente, antes de reiniciar, guarda la configuración recibida en el último *wget* en las ubicaciones correspondientes y la configuración de red usada para que sea configurada por defecto en el equipo. Estas acciones se muestran al usuario de la siguiente manera:

Al finalizar el proceso si no ocurrió ningún problema se le informará al usuario mediante una pantalla como la siguiente:



Proceso de conexión, autentificación y post conexión

Luego de reiniciar el concentrador VPN iniciará una conexión VPN contra el servidor central de OpenVPN. Este último, primero validará el certificado del cliente y en caso exitoso, el servidor OpenVPN ejecutará el script connect.py. Tanto la configuración del servidor OpenVPN central y el script connect.py se puede encontrar en el Anexo 4.

El script connect.py es el encargado de agregar la ruta de los puestos de trabajo del cliente al sistema. Para conocer de qué red se trata consulta la base de datos, luego de agregar la ruta registra el evento del inicio de conexión. Todavía el concentrador no ha completado las validaciones necesarias; para ello, este script dispara otro llamado post-connect.py en segundo plano y termina su ejecución sin errores, lo que le indicará a OpenVPN que termine de levantar el túnel y deje activa la conexión del cliente. El script post-connect.py, como se puede ver en el Anexo 4, mediante una conexión SSH, validará la clave de hardware y de credenciales de usuarios y reglas de firewall sean la mismas que la

registradas al momento de la instalación. En caso de coincidir los valores de estas claves con las registradas este script realizará las siguientes tareas:

- Registrar el final de la instalación en caso de ser la primera conexión.
- Actualizar los DNS para registrar la IP física del concentrador VPN (registro que será consultado por los puestos de trabajo para poder conectarse).
- Habilitar el acceso en el firewall hacia la aplicación de registro desde todos los puestos de trabajo del cliente.

En caso de fallar alguna de las comprobaciones sólo se registrará el evento en la tabla de auditoría y en el estado del cliente.

Inversamente, al producir la desconexión de un concentrador VPN se ejecutará el script disconnect.py (Anexo 4) que se encargará de eliminar las rutas agregadas previamente y de quitar los permisos de firewall, además de registrar el evento y modificar el estado del cliente.

Herramientas de soporte

El operador del módulo de Gestión de VPNs tiene a disposición, la información sobre el estado de conexión de cada uno de los clientes. En caso de ocurrir algún problema durante la conexión de uno de los clientes, ya sea que no pasó la validación de hardware o la validación de las credenciales de usuario y reglas de firewall, el estado reflejará el problema en cuestión.

Ante la falla de validación de hardware o de credenciales de usuario y reglas de firewall, el operador puede forzar la actualización de los valores registrados, para que al iniciar una nueva conexión, el cliente pase las verificaciones establecidas. Estas herramientas son Validar Shadow (credenciales de usuario y reglas de firewall) y Validar Hardware.

En caso que el concentrador VPN haya tenido un desperfecto, no encienda, no esté más disponible o cualquier otro caso donde se deba reinstalar el mismo, el operador dispone de la herramienta para habilitar la

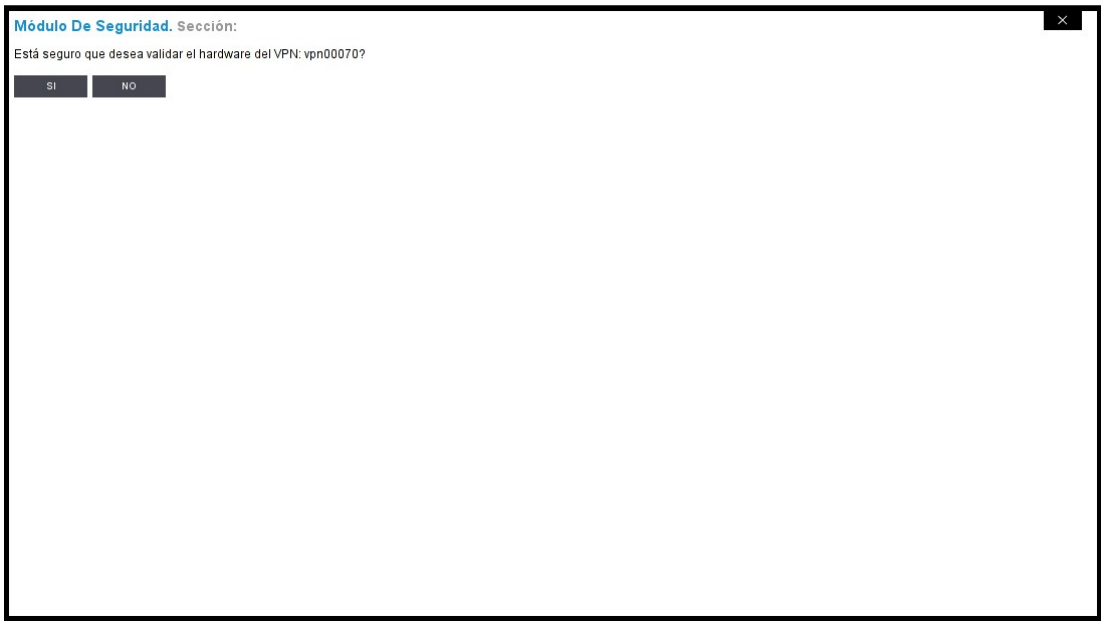
reinstalación del concentrador VPN.

Complementariamente, para brindar más información al usuario, el concentrador VPN viene con un usuario de diagnóstico que al loguearse brinda en pantalla información sobre estado de conexión del concentrador.

Validar Hardware

Suponiendo que se realice algún cambio en el hardware del concentrador VPN, ya sea un reemplazo de la placa de red o cualquier otro cambio. Durante el siguiente inicio de conexión el concentrador VPN no podrá validar el hardware y el script post-connect.py no habilitará el firewall para que los puestos de trabajo puedan alcanzar la aplicación de registro. El estado del cliente tendrá valor 1 y se habilitará un botón para validar el hardware, como el siguiente: VH.

El operador podrá optar por solicitar al cliente que reinstale el concentrador VPN en el nuevo hardware o podrá directamente validar el nuevo hardware sin tener que reinstalar. En caso que opte por la segunda opción hará uso del botón Validar Hardware que mostrará una pantalla como la siguiente:



Si confirma la acción se ejecutará el script updateHard.sh que recibirá

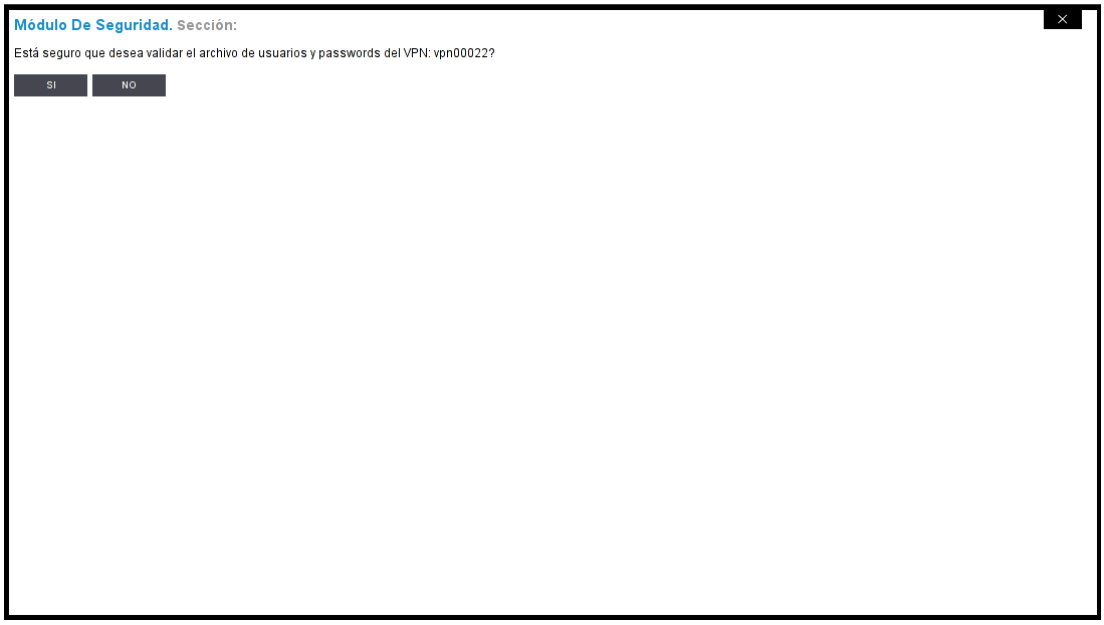
un único parámetro, el identificador de VPN. El script realizará una conexión SSH hacia el concentrador VPN para obtener los nuevos valores de hardware, actualizará los valores de la base de datos y matará la conexión desde la terminal del OpenVPN para forzar la reconexión del cliente. El script updateHard.sh se puede encontrar en el Anexo 5.

Luego de llamar al script el módulo de Gestión de VPNs registrará en la tabla de auditoría la acción realizada guardando el usuario del operador que solicitó validar el nuevo hardware.

Validar Shadow (credenciales de usuario y reglas de firewall)

Ante algún cambio en los usuarios y/o contraseñas del concentrador VPN o una modificación en sus reglas de firewall, el mismo no pasará las validaciones realizadas al comienzo de conexión. Este tipo de cambios puede darse ante la instalación de algún paquete en el concentrador, o simplemente ante una modificación no autorizada con intenciones desconocidas.

El operador podrá identificar el problema fácilmente, ya que el valor del estado del cliente será 2 y aparecerá el botón para validar el nuevo conjunto de usuarios, contraseñas y reglas de firewall:  . Si el operador conoce una razón válida para habilitar el nuevo conjunto de usuarios, contraseñas y reglas de firewall puede validarlo usando este botón. En caso contrario deberá optar por forzar una reinstalación del concentrador VPN. Si opta por validarlo verá una pantalla como la siguiente:

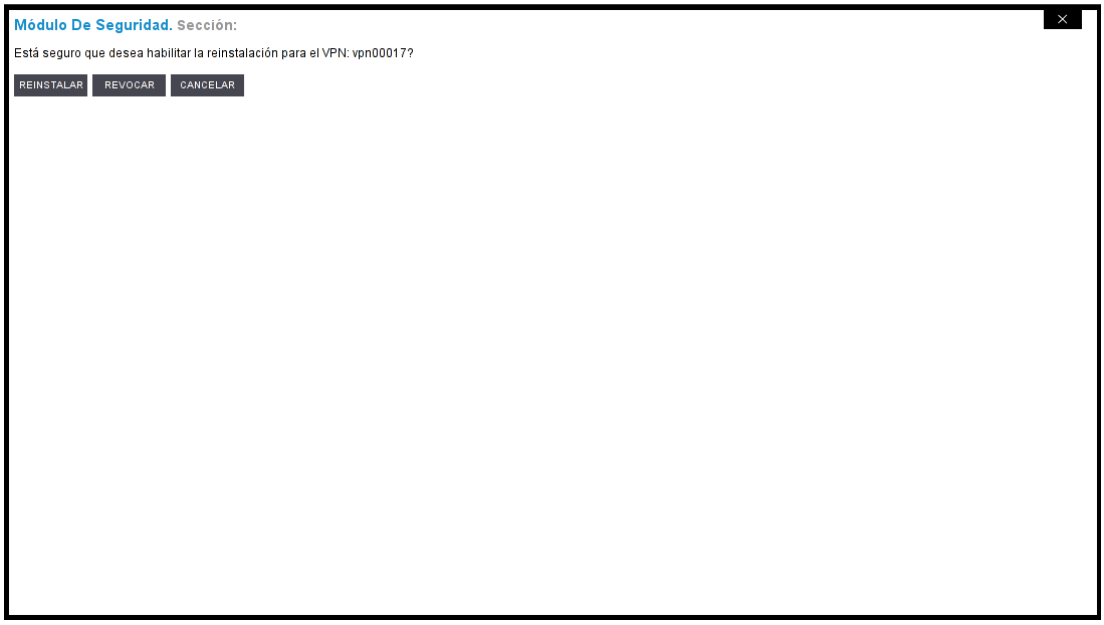


Si se confirma la acción se ejecutará el script updateShadow.sh que se puede ver en el Anexo 5. Este script calculará el nuevo valor de shadowmd5 luego de conectarse por SSH al concentrador VPN y lo guardará en la base de datos. Luego matará la conexión del cliente en cuestión para forzar la reconexión del mismo.

El módulo de Gestión de VPN luego de ejecutar el script registrará en la tabla de auditoría la acción realizada guardando el usuario que lo solicitó.

Habilitar reinstalación

Cuando exista un problema concreto que requiera cambiar de máquina de concentrador VPN, o en cualquier caso que se modifique la configuración de red del cliente (mudanza, migración, etc) será necesario reinstalar el concentrador VPN. Esta acción no podrá realizarse sin antes obtener autorización para ello, el operador podrá usar el botón R destinado a habilitar la reinstalación del cliente que mostrará una pantalla como la siguiente:



Esta pantalla le permite al operador optar entre la reinstalación del cliente manteniendo las credenciales o adicionalmente revocar los certificados anteriores y generar nuevos certificados. La elección de la opción de Reinstalar sólo cambiará valores en la base de datos que permitan al cliente realizar la instalación, si en cambio se opta por Revocar se llamará al script recrearVPN.py (Anexo 5), que incluirá el certificado en cuestión a la Lista de Revocación (CRL), generará un nuevo certificado y una nueva licencia para el cliente y los guardará en la base de datos.

En cualquier caso la acción realizada será registrada junto con el usuario del operador que lo solicitó en la tabla de auditoría.

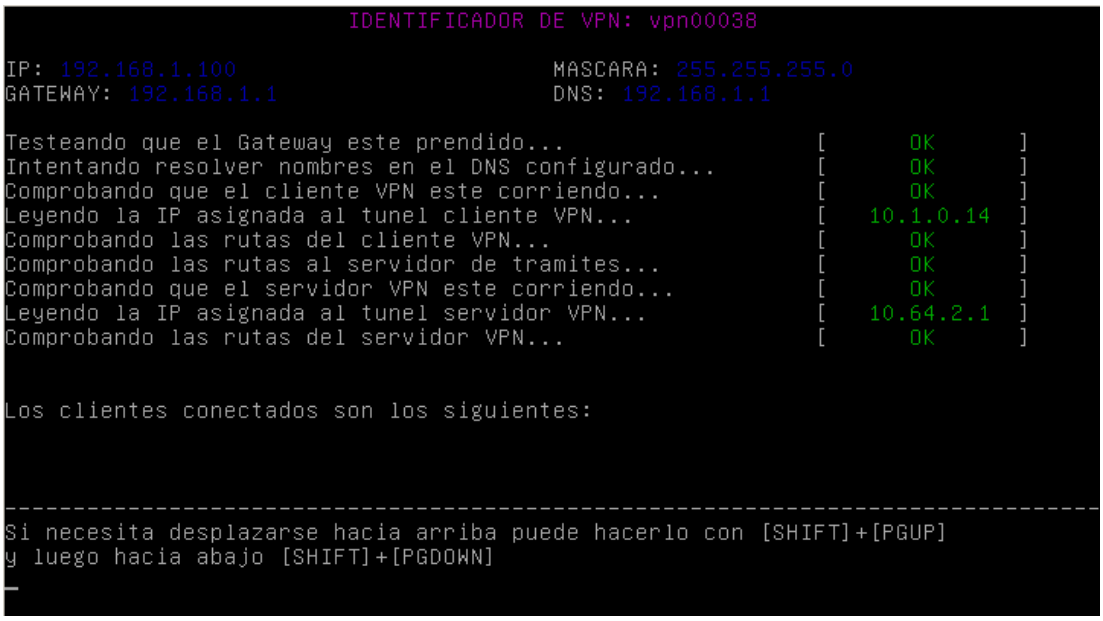
Usuario de diagnóstico

Por otra parte, se provee una herramienta que cada usuario puede usar para diagnosticar el estado de conectividad del concentrador VPN, para eso existe un usuario preinstalado en el mismo. Las credenciales para realizar este diagnóstico son:

```
Usuario: test
Contraseña: test
```

Al loguearse con estas credenciales en un concentrador VPN, éste realizará un diagnóstico y mostrará el resultado en pantalla. A continuación

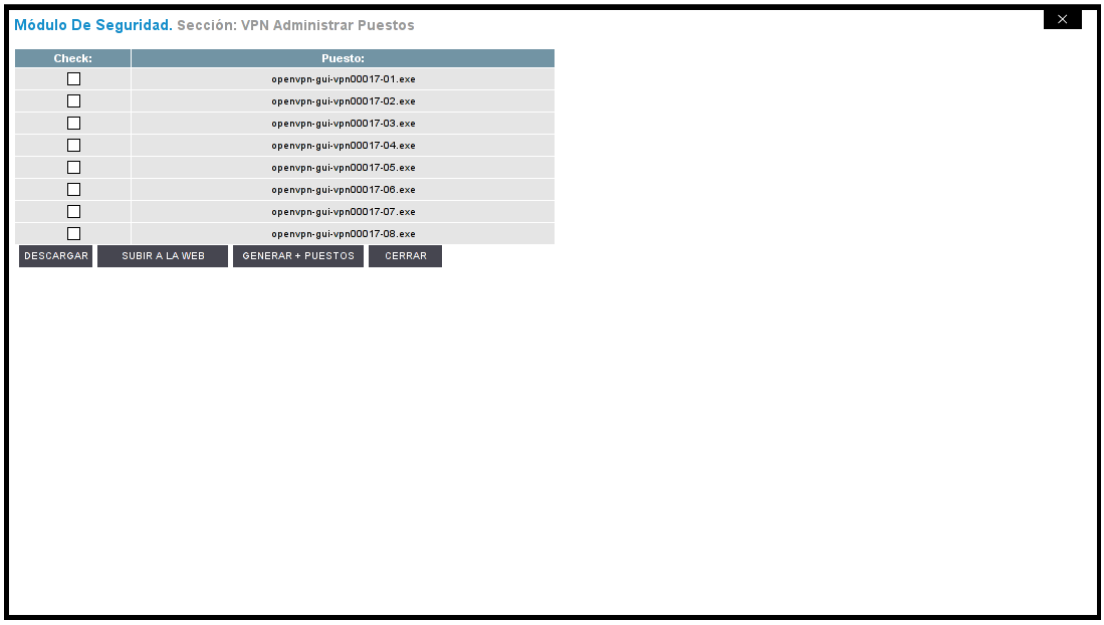
se muestra un ejemplo:



En caso de fallar alguno de los puntos, el mismo será mostrado en color rojo para resaltar el problema. Esta herramienta es muy útil para que los operadores de soporte puedan obtener información técnica del estado del servidor y puedan decidir el mejor rumbo para solucionar posibles problemas.

Descargar, publicar y generar puestos de trabajo

Para la gestión de puestos de trabajo que puede tener cada cliente hay un botón P que nos provee una interfaz donde ver el listado de puestos ya generados para ese cliente:

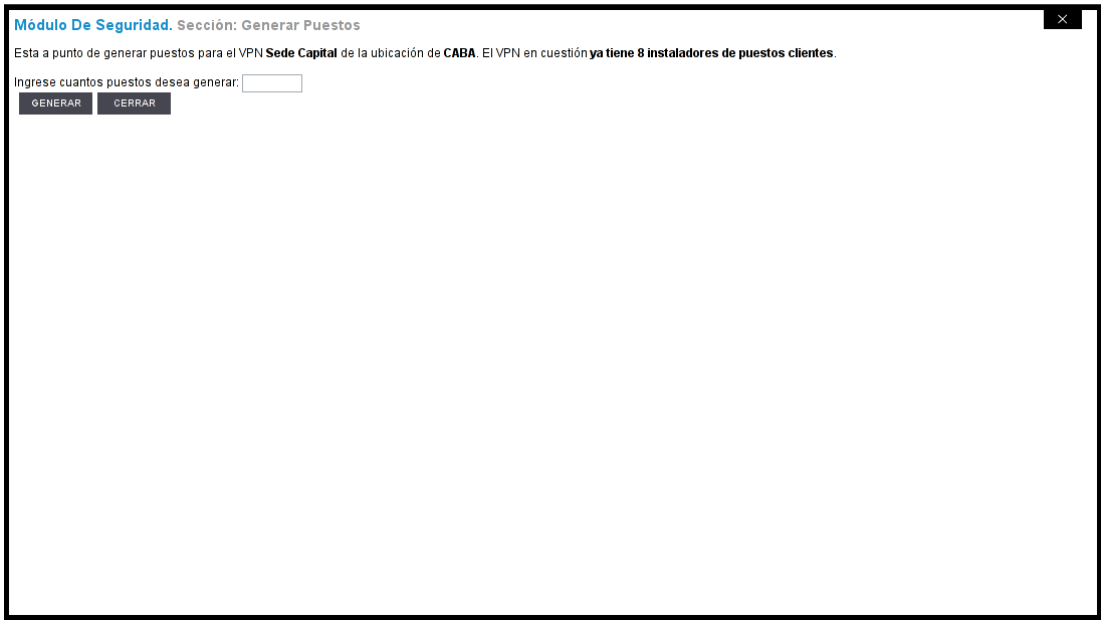


En esa interfaz podemos seleccionar, uno o varios puestos y clickeando en el botón “Descargar” se generará un archivo *RAR* para que lo descarguemos. El mismo será cifrado con una contraseña aleatoria y se mostrará el enlace en una pantalla similar a la siguiente:

Lo mismo sucederá si usamos el botón “Subir a la Web”, pero en este caso el archivo *RAR* será ubicado en una carpeta del servidor Web público y será restringido el acceso a un usuario y contraseña. Esto se realiza mediante la generación de un archivo *.htaccess* de *Apache* para esa

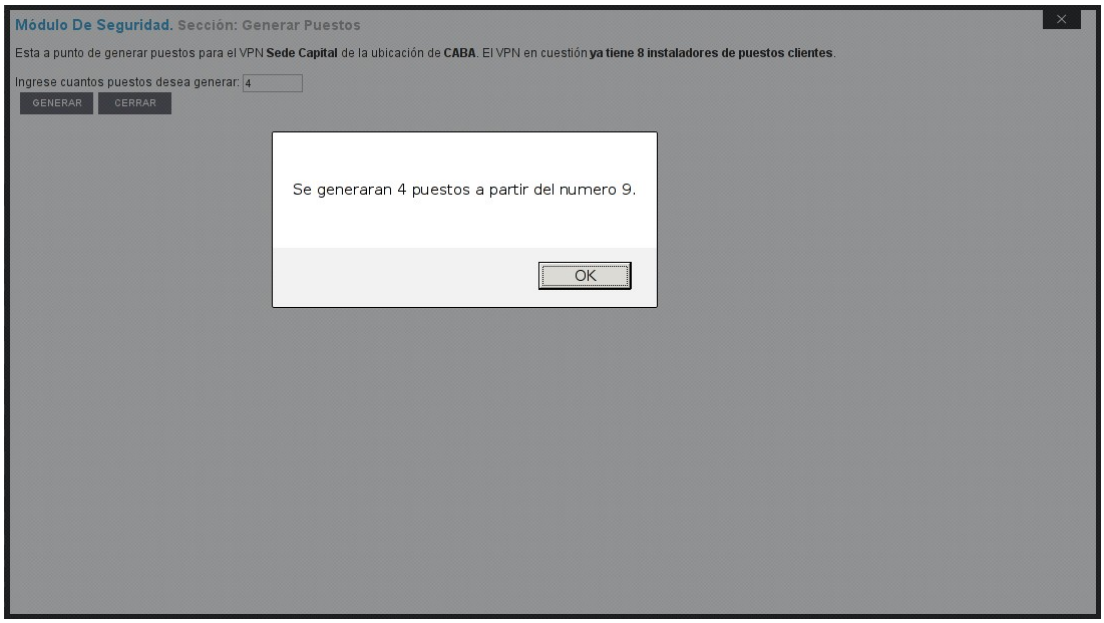
carpeta, como se puede ver en el script subirPuestos.sh en Anexo 5 [13]. Las credenciales y el enlace serán mostrados al operador como se ve en la pantalla siguiente:

Si lo que se requiere es generar nuevos puestos, se hará click en el botón “Generar + Puestos” el cual mostrará al usuario una interfaz similar a la siguiente:



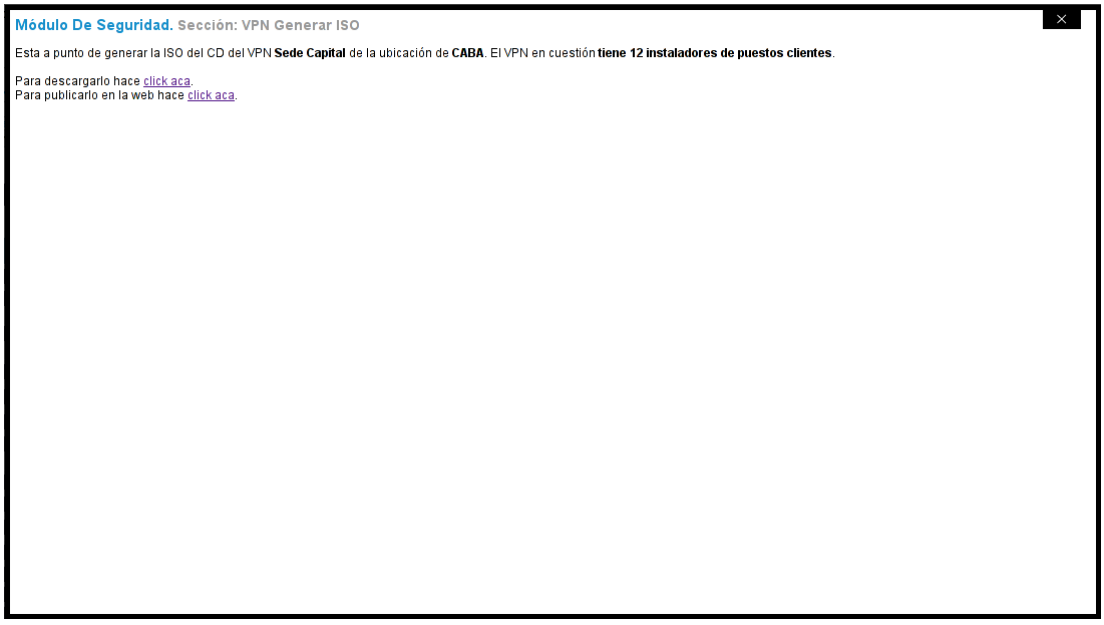
El operador ingresará la cantidad de puestos que se requieren generar. Al confirmarla mediante el botón “Generar” se mostrará un cartel

como el siguiente y al cabo de un instante estarán disponibles los nuevos puestos.



Descargar y publicar CD de instalación

Al hacer uso del botón para generar el CD de instalación nos lleva a una interfaz similar a la siguiente:

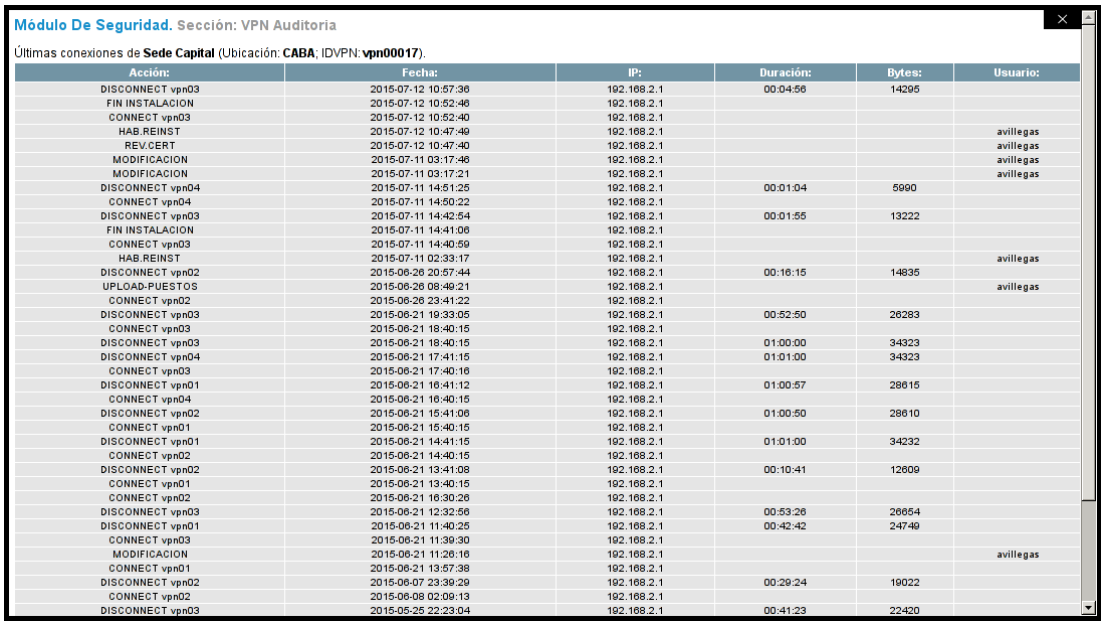


Si el operador elige descargar el *ISO* para grabar el CD él mismo, se mostrará una página con el enlace al *ISO* como la siguiente:

Si el operador elige publicar el *ISO* para que sea el cliente quien lo grabe, se generará en el servidor Web público una carpeta con restricciones de acceso a la que sólo podrá acceder la persona que conozca las credenciales. En esa carpeta se copiará el *ISO*. De manera análoga a como se hace con los puestos, para poder restringir el acceso se implementa *HTTP Authentication* mediante un archivo *.htaccess* de *Apache* como se puede ver en el script subirISO.sh del Anexo 5 [13]. Al operador se le mostrará el enlace al *ISO* y las credenciales necesarias para la descarga, como se ve en la siguiente interfaz:

Módulo de auditoría

Desde el botón  se puede acceder a todos los eventos del cliente en cuestión. Al hacerlo se muestra una página similar a la siguiente:



Módulo De Seguridad. Sección: VPN Auditoria

Últimas conexiones de **Sede Capital** (Ubicación: **CABA**; IDVPN: **vpn00017**).

Acción:	Fecha:	IP:	Duración:	Bytes:	Usuario:
DISCONNECT vpn03	2015-07-12 10:57:36	192.168.2.1	00:04:56	14295	
FIN INSTALACION	2015-07-12 10:52:46	192.168.2.1			
CONNECT vpn03	2015-07-12 10:52:40	192.168.2.1			
HAB.REINST	2015-07-12 10:47:49	192.168.2.1			avillegas
REV.CERT	2015-07-12 10:47:40	192.168.2.1			avillegas
MODIFICACION	2015-07-11 03:17:46	192.168.2.1			avillegas
MODIFICACION	2015-07-11 03:17:21	192.168.2.1			avillegas
DISCONNECT vpn04	2015-07-11 14:51:25	192.168.2.1	00:01:04	5990	
CONNECT vpn04	2015-07-11 14:50:22	192.168.2.1			
DISCONNECT vpn03	2015-07-11 14:42:54	192.168.2.1	00:01:55	13222	
FIN INSTALACION	2015-07-11 14:41:06	192.168.2.1			
CONNECT vpn03	2015-07-11 14:40:59	192.168.2.1			
HAB.REINST	2015-07-11 02:33:17	192.168.2.1			avillegas
DISCONNECT vpn02	2015-06-26 20:57:44	192.168.2.1	00:16:15	14835	
UPLOAD-PUESTOS	2015-06-26 08:49:21	192.168.2.1			avillegas
CONNECT vpn02	2015-06-26 23:41:22	192.168.2.1			
DISCONNECT vpn03	2015-06-21 19:33:05	192.168.2.1	00:52:50	26283	
CONNECT vpn03	2015-06-21 18:40:15	192.168.2.1			
DISCONNECT vpn03	2015-06-21 18:40:15	192.168.2.1	01:00:00	34323	
DISCONNECT vpn04	2015-06-21 17:41:15	192.168.2.1	01:01:00	34323	
CONNECT vpn03	2015-06-21 17:40:16	192.168.2.1			
DISCONNECT vpn01	2015-06-21 16:41:12	192.168.2.1	01:00:57	28615	
CONNECT vpn04	2015-06-21 16:40:15	192.168.2.1			
DISCONNECT vpn02	2015-06-21 15:41:06	192.168.2.1	01:00:50	28610	
CONNECT vpn01	2015-06-21 15:40:15	192.168.2.1			
DISCONNECT vpn01	2015-06-21 14:41:15	192.168.2.1	01:01:00	34232	
CONNECT vpn02	2015-06-21 14:40:15	192.168.2.1			
DISCONNECT vpn02	2015-06-21 13:41:08	192.168.2.1	00:10:41	12609	
CONNECT vpn01	2015-06-21 13:40:15	192.168.2.1			
CONNECT vpn02	2015-06-21 16:30:26	192.168.2.1			
DISCONNECT vpn03	2015-06-21 12:32:56	192.168.2.1	00:53:26	26654	
DISCONNECT vpn01	2015-06-21 11:40:25	192.168.2.1	00:42:42	24749	
CONNECT vpn03	2015-06-21 11:39:30	192.168.2.1			
MODIFICACION	2015-06-21 11:26:16	192.168.2.1			avillegas
CONNECT vpn01	2015-06-21 13:57:38	192.168.2.1			
DISCONNECT vpn02	2015-06-07 23:39:29	192.168.2.1	00:29:24	19022	
CONNECT vpn02	2015-06-08 02:09:13	192.168.2.1			
DISCONNECT vpn03	2015-05-25 22:23:04	192.168.2.1	00:41:23	22420	

Como se puede ver a simple vista no todos los registros de auditoría tienen los mismos atributos, cada tipo de evento tiene campos propios. Todos los registros de auditoría guardan fecha y hora, servidor que interviene en la acción, identificador de VPN del cliente e IP remota (puede

ser la del cliente o la del operador dependiendo de la acción). A continuación se detallan los diferentes eventos que son registrados y qué atributos adicionales tienen:

- INICIA INSTALACIÓN: Se produce luego de enviar los archivos de configuración y certificados durante la instalación.
- CONNECT: Se produce cuando un concentrador VPN inicia una conexión VPN hacia el VPN central. El registro guarda adicionalmente la instancia de OpenVPN que lo atiende.
- FIN INSTALACIÓN: Se produce cuando el concentrador VPN conecta por primera vez después de la instalación. Guarda los mismos registros que el evento CONNECT.
- DISCONNECT: Se produce cuando un concentrador VPN cierra la conexión. Este registro tiene adicionalmente, la duración de la conexión que está cerrando y los bytes transmitidos en la misma.
- HAB.REINST: Se produce cuando un operador habilita la reinstalación de un concentrador VPN. Guarda el usuario del operador que solicitó la operación.
- REV.CERT: Ocurre cuando un operador revoca los certificados de un concentrador VPN. Se guarda adicionalmente, el usuario de dicho operador.
- VALIDA HARD: Se registra al forzar la actualización de la clave de hardware de un cliente. Este registro tiene además el usuario que realizó la operación.
- VALIDA SHADOW: Se registra al forzar la actualización de la clave de shadow (hash de credenciales de usuarios locales y reglas de firewall) de un concentrador VPN. Este registro tiene además el usuario que realizó la operación
- GENERO [n] PUESTOS: Es generado cuando un operador solicita la generación de n puestos para un cliente. Registra el usuario que lo realizó.
- MODIFICACIÓN: Este evento se registra al realizar una modificación

en los datos de un cliente. Se guarda también el usuario que lo ejecuta.

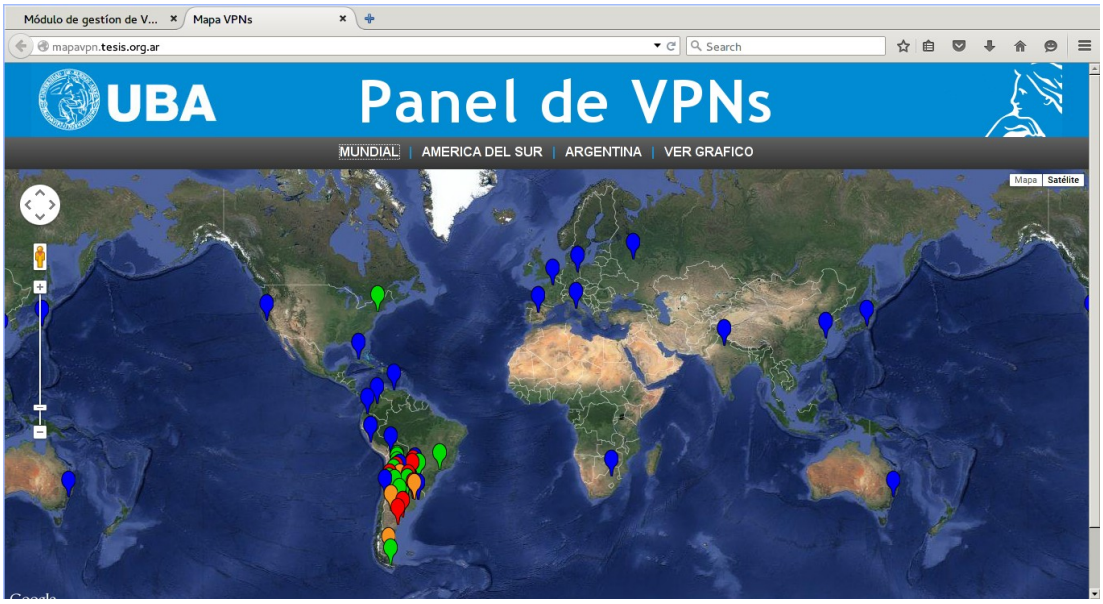
- ALTA: Se produce cuando se genera un nuevo cliente. Guarda el usuario que lo genera, el nombre y la ubicación ingresados. Será siempre el primer evento de cada cliente.
- DOWNLOAD-PUESTOS: Este evento se genera cuando un operador selecciona uno o más instaladores VPN de puestos de trabajo de un cliente para descargarlos. Registra el usuario del operador.
- UPLOAD-PUESTOS: Ídem anterior, sólo que sucede cuando se elige publicar el paquete en la página Web institucional.
- DOWNLOAD-CD: Sucede cuando un operador genera la imagen *ISO* de un concentrador VPN para descargarla. Registra el usuario del operador.
- UPLOAD-CD: Ídem anterior, pero el operador elige publicarla en la página Web institucional para su descarga remota. También registra el usuario del operador.

Tablero de estadísticas y monitoreo

El módulo de Gestión de VPN de desarrollo propio y open source nos brinda la posibilidad de futuros desarrollos y adaptaciones a las necesidades propias. En nuestro proyecto el tablero de estadística y monitoreo estaba incluido desde el principio pero podría suceder que naciera un nuevo requerimiento, que haciendo pequeñas modificaciones, puedan cumplirse. Esta flexibilidad no está siempre disponible en otros sistemas de VPN de código propietario.

La solución que desarrollamos incluye una visión gráfica de la ubicación de todos los concentradores VPN, que muestra además el estado de la conexión de cada uno. A continuación se muestran algunas capturas de pantalla de ejemplo.

Visión MUNDIAL

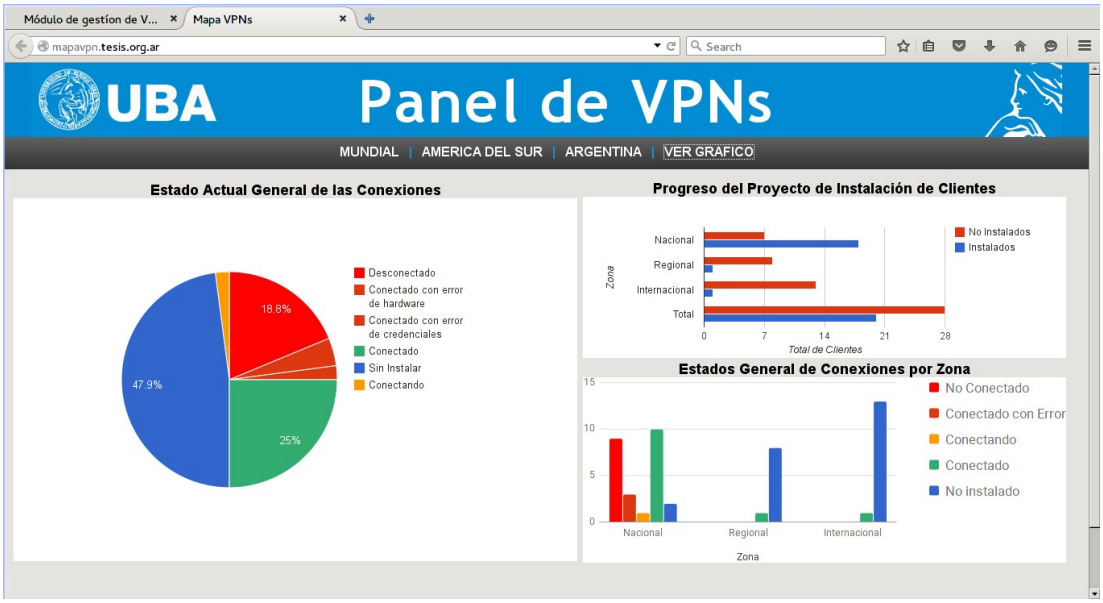


Visión AMÉRICA DEL SUR:

Visión ARGENTINA:

Como se puede ver cada globo de los mapas representa un concentrador VPN. Su color está relacionado con el estado de su conexión. El color verde indica que el cliente está conectado, naranja que se encuentra con algún problema de validación, rojo que está desconectado y azul que aún no fue instalado.

A continuación se puede ver un ejemplo de la opción VER GRÁFICO. Estos cuadros muestran el progreso de instalación, y una estadística general, y por región, del estado de conexión de los clientes.



Configuración OpenVPN

La configuración del OpenVPN puede encontrarse en el Anexo 4. Por razones que son explicadas más adelante se decidió iniciar cuatro instancias de OpenVPN, cada una escucha en un puerto distinto, con un puerto de gestión diferente y entregando IPs a sus clientes en un rango de red distinto.

Se eligió que OpenVPN use UDP, debido a que no se recomienda hacer un túnel TCP sobre TCP porque duplica los controles de flujo del protocolo y, por lo tanto, es más lento. La recomendación es, por consiguiente, hacer el túnel sobre UDP [14].

La interfaz usada es tipo *tun (dev-type)*, debido a que la interfaz *tap* permitiría la interacción entre concentradores VPN sin intervención de OpenVPN central y esta funcionalidad no es requerida. Cada instancia de OpenVPN tiene un rango /20 destinado a atender los concentradores. Este rango tiene 4096 direcciones posibles pero debemos tener en cuenta que cada conexión VPN genera una subred /30 por lo cual utiliza 4 direcciones. Así, los rangos asignados a cada instancia le permiten atender 1024 clientes cada una. Si el número de cliente continúa creciendo se podrían extender estos rangos o simplemente levantar más instancias de OpenVPN y aplicar parches a los concentradores para que hagan uso de los nuevos posibles destinos.

La opción *tls-auth* ya fue explicada previamente, sirve para prevenir ataques de denegación de servicio (DoS) y de buffer-overflow de OpenSSL. Define una clave estática precompartida, la cual es usada durante la fase de negociación SSL por parte de OpenVPN.

Una opción muy importante usada en la configuración es *client-config-dir* que especifica el directorio de dónde se leerán las opciones particulares de cada concentrador VPN. Al momento de generar un cliente se guarda en ese directorio un archivo cuyo nombre es el *common name (CN)* del certificado del cliente. Dentro de ese archivo se especifica una opción para asociar internamente la ruta de los puestos de trabajo del cliente a esa conexión, esta ruta se define con la opción *iroute*.

Las opciones *ping* y *ping-restart* son usadas para monitorear el estado de conexión de cada cliente y dar de baja dicha conexión si se produce una inactividad de más de 60 segundos. Adicionalmente, se usa compresión (*comp-lzo*) para reducir el uso de ancho de banda.

En la configuración OpenVPN se definen los scripts que serán ejecutados cuando un cliente se conecta y cuando se desconecta, usando las opciones *client-connect* y *client-disconnect* respectivamente, Para que OpenVPN nos permita usar scripts programados por nosotros debemos usar la opción *script-security* con valor 2. Durante la conexión se fuerza al cliente a que incorpore las rutas de las redes de servicio. Así, usando la opción *push,* cada concentrador VPN incorpora las rutas a las redes 192.168.3.0/24 y 10.1.0.0/16 a través del túnel VPN.

Otra opción fundamental usada, es la que especifica la Lista de Revocación de Certificados (CRL) que será evaluada en cada conexión. La CRL en cuestión se define mediante el uso de la opción *crl-verify*, esta opción es la que nos permite revocar certificados.

Alta disponibilidad de los servicios

La instalación del servidor de VPN central es realizada por duplicado, formando un clúster de dos servidores que pueden brindar el servicio en alta disponibilidad. El sistema usado para armar el clúster que se auto controle y migre los servicios de uno a otro es Corosync Pacemaker. Este gestor de clúster ofrece la posibilidad de hacer un clúster de alta disponibilidad activo pasivo o de balanceo de carga. Si bien *a priori* la segunda opción es mejor, genera inconvenientes de ruteo en la infraestructura, incluso usando ruteo dinámico. Analizando en profundidad no se obtiene un gran rédito de una implementación con balanceo de carga, ya que no existe un problema de carga o saturación de uno de los nodos, por todo esto se implementó un clúster de alta disponibilidad activo pasivo.

La configuración de Corosync implementada puede encontrarse en el Anexo 6. Como se puede ver existen una condición para que un nodo pueda

tomar el servicio, debe tener ping contra la puerta de enlace de Internet. Si cumple con esa condición podrá tomar el servicio, en ese caso por las reglas definidas, primero enviará un correo electrónico informando la migración del servicio, luego levantará la IP móvil *failover-ip-dmz*. Esta dirección IP es a las que apuntan los servicios (desde Internet) y a la que apuntan las rutas de VPN (desde Intranet). Finalmente, si no ocurrió ningún problema con las dos primitivas anteriores, levantará el servicio de OpenVPN. [10]

Otros servicios muy importantes son los que brinda el servidor de instalación, ya que aunque tal vez pueda entenderse que la instalación podría no ser considerada un servicio crítico, en estos servidores corre el motor de base de datos que es utilizado por el servidor central de VPN en cada conexión y es el motor que alimenta de información al sistema de gestión de VPN. Por esta razón el servidor de instalación es implementado por duplicado para formar un clúster activo pasivo que brinde sus servicios a través de una IP móvil. Para que ambos servidores tengan los datos consistentes y que ambos puedan tener en cualquier momento la IP de servicio, los motores de *MySQL* serán configurados en un sistema de replicación Master-Master. Mediante esta configuración ambos motores pueden ejecutar actualizaciones en los datos y estos cambios serán replicados al otro servidor.

En el Anexo 6 se muestra la configuración de Corosync usada para asignar la IP de servicio entre ambos servidores. En esta configuración se aplica un ping a la puerta de enlace para testear el servicio de red y además se encarga de monitorear que el servicio de *MySQL* esté ejecutándose. Si ambas comprobaciones son exitosas, el servidor se encuentra en condiciones de tener la IP de servicio. En cada migración de la IP el servicio de Corosync enviará un correo electrónico avisando del evento.

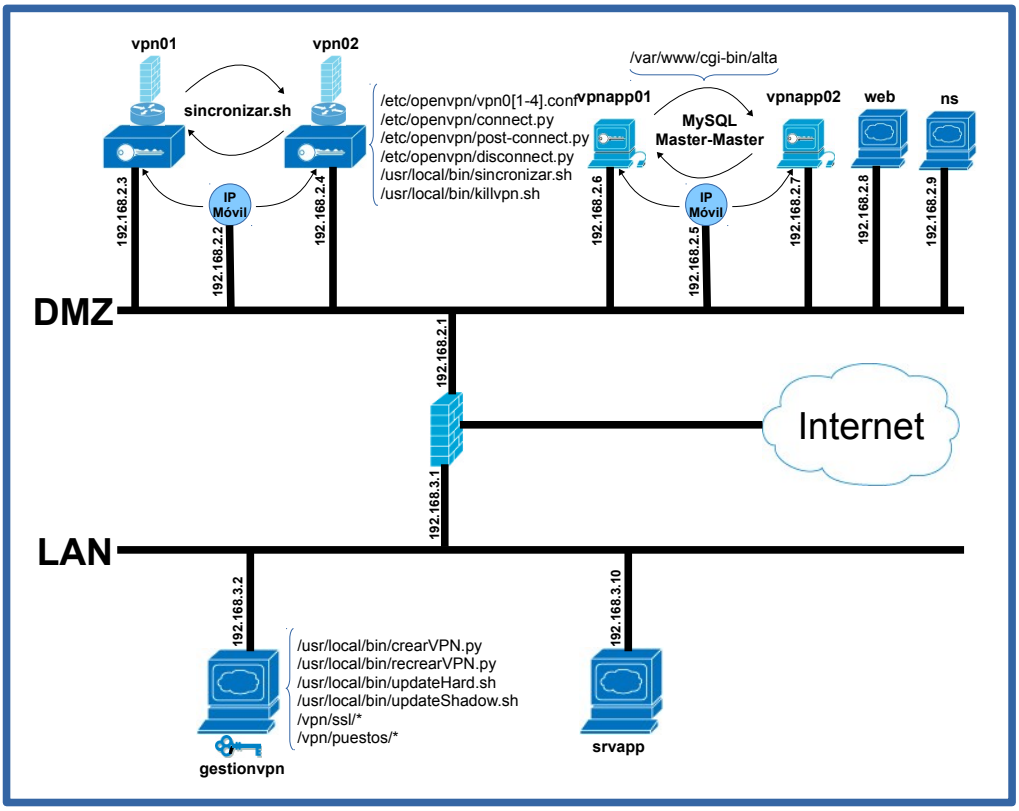
Interacción entre los servidores

La base para poder interactuar entre los servidores y que los scripts funcionen como fueron desarrollados, es que todos los servidores de la

infraestructura de VPN tengan la resolución de DNS de los nombres de dominio de aquellos servidores con los que interactúa. Si el servicio de DNS no incluye estos nombres se debe agregar las siguientes líneas al archivo /etc/hosts de todos los servidores con las IPs que correspondan:

```
192.168.2.2       vpn
192.168.2.3       vpn01
192.168.2.4       vpn02
192.168.2.5       vpnapp
192.168.2.6       vpnapp01
192.168.2.7       vpnapp02
192.168.2.8       web www.tesis.org.ar tesis.org.ar
192.168.2.9       ns.tesis.org.ar
192.168.3.2       gestionvpn gestionvpn.tesis.org.ar
192.168.3.10      srvapp srvapp.tesis.org.ar
```

La topología de red y la ubicación de cada script pueden verse en el siguiente gráfico:



La mayoría de los scripts interactúa con el motor de base de datos, por lo cual se deberá habilitar en el firewall el acceso desde el servidor gestionvpn (Intranet) hacia el puerto TCP 3306 de la IP móvil del vpnapp (DMZ). También será necesario el acceso desde el servidor gestionvpn hacia la IP móvil del servidor vpn (DMZ) y hacia el rango de IPs de los

concentradores VPN (alcanzable a través del mismo servidor) al puerto TCP 22 de SSH. El servidor gestionvpn dispone de una llave SSH-RSA con la cual se autentifica de manera desatendida, tanto en los servidores vpn01 y vpn02, como en los concentradores VPN y es usada por los scripts como veremos a continuación:

- Durante la creación de un nuevo cliente, el servidor gestionvpn crea los certificados, los guarda en la base de datos ubicada en vpnapp y luego (línea 100 de crearVPN.py) crea en el servidor vpn el archivo de configuración de ruteo específico para ese cliente.
- El script updateHard.sh lee la información del concentrador VPN de la base de datos (vpnapp) y luego se conecta al concentrador VPN en cuestión para recuperar la nueva información del hardware. Esto lo hace usando una llave SSH (línea 19). Luego actualiza los valores en la base de datos y en la línea 34, usando una conexión SSH, ejecuta el script killvpn.sh (Anexo 5) en el servidor vpn para que desconecte al cliente en cuestión.
- El script updateShadow.sh lee la información del concentrador VPN de la base de datos (vpnapp) y luego se conecta al concentrador VPN en cuestión para recuperar la nueva información de las credenciales de usuarios y reglas de firewall. Esto lo hace usando una llave SSH como se ve en la línea 14 del script. Luego actualiza los valores en la base de datos y en la línea 25, usando una conexión SSH, ejecuta el script killvpn.sh (Anexo 5) en el servidor vpn para que desconecte al cliente en cuestión.
- Cuando se revocan los certificados de un cliente, el servidor gestionvpn después de agregar al certificado a la lista de revocación (CRL), copia la nueva crl.pem al servidor vpn usando la llave SSH como se ve en la línea 41 del script. Luego por SSH ejecuta la sincronización el clúster vpn y desconecta al cliente en cuestión como se ve en la línea 42 del script.

  Respecto a los scripts subirPuestos.sh y subirISO.sh, estos no

interactúan directamente con el servidor web (DMZ) sino que usan dos carpetas locales, en el servidor gestionvpn, para poner los archivos que serán accedidos desde Internet. Para eso, el servidor web usará el módulo proxy de Apache y el servidor gestionvpn servirá por HTTP estas carpetas. Para eso en la configuración del dominio virtual www.tesis.org.ar del servidor web (DMZ) se deben incluir las siguientes líneas de configuración:

```
ProxyRequests off
ProxyPass        /isos/ http://gestionvpn.tesis.org.ar/isospub/
ProxyPassReverse /isos/ http://gestionvpn.tesis.org.ar/isospub/
ProxyPass        /puestos/ http://gestionvpn.tesis.org.ar/ppub/
ProxyPassReverse /puestos/ http://gestionvpn.tesis.org.ar/ppub/
```

Y en el servidor gestionvpn dentro de la carpeta principal del dominio virtual gestionvpn.tesis.org.ar hay dos enlaces simbólicos que apuntan a la carpeta donde los scripts dejan los archivos que serán accedidos por la web institucional:

```
lrwxrwxrwx 1 root  root  14 mar  2  2013 isospub -> /vpn/isospub/
lrwxrwxrwx 1 root  root  14 mar  2  2013 ppub -> /vpn/puestospub/
```

Y otros dos para los archivos que no serán publicados en Internet sino que serán los accedidos por el operador de la plataforma de gestión VPN:

```
lrwxrwxrwx 1 root  root  14 mar  2  2013 isos -> /vpn/isos/
lrwxrwxrwx 1 root  root  14 mar  2  2013 puestos -> /vpn/puestos/
```

Para prevenir llenar los discos de almacenamiento con imágenes ISO o clientes VPN para puestos de trabajo ya descargados, se ha configurado, además, el siguiente CRON que eliminará las carpetas viejas.

```
52 23 * * * find /vpn/isos/ -type d -mtime +3 -delete
52 23 * * * find /vpn/isospub/ -type d -mtime +15 -delete
52 23 * * * find /vpn/puestos/ -type d -mtime +3 -delete
52 23 * * * find /vpn/puestospub/ -type d -mtime +15 -delete
```

Por último, el script post-connect.py debe modificar la zona de DNS para actualizar los registros que guardan las direcciones de los concentradores VPN. Como se puede ver en la línea 61 interactúa con el servidor de dominio de la zona tesis.org.ar usando el comando *nsupdate*. Para poder realizar los cambios utiliza una llave HMAC-MD5 de 512bits ubicada en /etc/openvpn/Ktesis.org.ar.key. Este comando modifica la zona pública de tesis.org.ar para actualizar la IP del concentrador VPN en el registro correspondiente (vpnNNNNN.vpns.tesis.org.ar). Este registro es el

consultado por los puestos de trabajo de ese cliente para poder iniciar la conexión contra el concentrador VPN.

Problemas encontrados y alternativas de mejora

Límite de clientes y caída de las conexiones

La primera aproximación a la solución final sólo contemplaba un proceso OpenVPN servidor en el VPN central y los clientes conectando estáticamente a este único proceso. Al año de estar en producción comenzamos a notar que los túneles se desconectaban, tenían caídas simultáneas a intervalos irregulares. O sea, esporádicamente (una vez por semana aproximadamente) notábamos que el mapa de conexiones se ponía todo rojo y seguidamente comenzaban a renegociar todos los clientes.

Este problema nos costó mucho diagnosticarlo, debido a que no podíamos provocar el problema para estudiarlo y lo veíamos como un suceso estocástico que no podíamos predecir. Adicionalmente *a posteriori* no encontrábamos en los logs indicios del motivo. Por lo cual la solución a la que arribamos la encontramos mediante prueba-error, sospechábamos que estaba relacionado con la cantidad de clientes, ya que cuando empezó a suceder el problema habíamos alcanzado los 600/700 concentradores VPN instalados y estábamos creciendo muy rápidamente en instalaciones nuevas. Esto nos hacía pensar en posibles límites de cantidad de sockets o archivos abiertos por el proceso de OpenVPN e intentamos por varios métodos subir esos límites, pero las caídas volvían a suceder. También hicimos intentos de incrementar la memoria RAM del servidor que por ese entonces estaba en 2Gb y se notaba al límite, pero tampoco dio resultados.

Finalmente, nos dimos por vencidos y encaramos el problema de una manera un poco más rústica y a la vez con una solución más efectiva. Nuestro razonamiento fue, si 500 clientes con un servidor OpenVPN no presentaban problemas, simplemente incrementemos la cantidad de servidores OpenVPN y balanceemos las conexiones. Así, levantamos cuatro procesos de OpenVPN (simplemente copiando la configuración del original y cambiándole, el puerto donde escucha el servicio, el puerto de gestión, la red donde entrega IPs a los clientes y los archivos auxiliares donde guarda

la información de estos.

Como los clientes ya instalados tenían configurados un único puerto destino de conexión era necesario balancear las conexiones entre los cuatro procesos, para ellos se automatizó la tarea mediante un parche. Cuando los concentradores VPN iniciaban la conexión, mediante SSH se les aplicaba el parche que modificaba la configuración de conexión. La nueva configuración incluía los nuevos destinos de conexión y la opción *remote-random*, para que el cliente tomara aleatoriamente cada uno de los destinos de conexión.

Como ya dije, la solución usada no es la más ortodoxa, ya que nunca terminamos de definir exactamente cuál era la causa del problema, pero esta solución, comprobada empíricamente, resolvió el problema definitivamente.

Tercer factor de autentificación

Cada usuario que utiliza el sistema primero debe tener el cliente VPN que autentifica el puesto dentro de la sede y esta sede es validada contra el VPN central, adicionalmente cuando el usuario ingresa al sistema de registro usa un segundo factor de autentificación que es su usuario y contraseña. Hemos estudiado la posibilidad de incorporar un tercer factor de autentificación. Para lo cual, se requiere el uso de tokens personales de cada usuario de la aplicación de registro.

Estos tokens serían usados en los puestos de trabajo y tendrían dentro un certificado personal del usuario que permitiría, no sólo cerrar una posible conexión HTTPS contra la aplicación de registro sino además firmar digitalmente los trámites realizados.

Esta propuesta fue elevada e incluso hubo contactos comerciales con algunos proveedores de tokens, pero aún no está aprobada y todo parece indicar que no será implementada. El mayor impedimento lo representa el costo de los tokens y el despliegue necesario para su puesta en funcionamiento.

Tiempo de vida de los certificados y año 2038

Como puede suponerse el tiempo de vida de los certificados de los clientes y de las Autoridades Certificantes (CA) implicará fechas topes para la realización de tareas de mantenimiento que renueven y actualicen los certificados instalados en el servidor VPN central, en los concentradores VPN y en los puestos de trabajo. A simple vista esta tarea representa un gran despliegue y esfuerzo para la gente de soporte, tanto de la organización como de los clientes. Debido a lo anterior debimos analizar detenidamente el escenario para determinar el largo del tiempo de vida de los certificados y la longitud de las claves. Teniendo en cuenta algunas predicciones y proyecciones de la criptografía moderna se puede concluir que si elegimos tiempos de vida largos es recomendable hacer llaves con un parámetro n de RSA más grande.

La fecha límite para la renovación de certificados depende de los tiempos de vida de los certificados de cada túnel SSL, por lo que podríamos tener diferentes tiempos de vida en cada uno y así manejar dos niveles de seguridad. Se debe tener en cuenta que el túnel que une los puestos de trabajo con los concentradores VPNs es el más difícil de gestionar, debido que el cliente OpenVPN del puesto de trabajo corre en un sistema operativo que no administramos. Debido a esto, hemos decidido utilizar certificados con tiempo de vida de 20 años para este último túnel. Mientras que en el túnel que une los concentradores VPN con el VPN central el tiempo de vida es la mitad. En ambos casos se ha usado certificados con claves RSA con un parámetro *n* de tamaño 4096 que, para los tiempos que corren, es un número relativamente alto. Aunque nadie puede predecir qué pasará de acá a 10 años.

La tarea de renovación de certificados de este último túnel ya está planificada. La misma se realizará de manera automatizada el último año de vida de los certificados. Se desarrollará un script que luego de validar el certificado instalado generará uno nuevo firmado por una nueva CA y actualizará la configuración del cliente para que éste se conecte al nuevo

servicio OpenVPN que atenderá con los nuevos certificados también firmados por la nueva CA. Como puede verse esta tarea renovará completamente la Infraestructura de Clave Pública (PKI) del túnel más crítico que permite el acceso desde las sedes remotas.

Independientemente del tiempo de vida de nuestros certificados, debemos tener en mente el problema de las variables time_t de Unix que tendrán un overflow de sus 32bits el 19 de enero de 2038. Este nuevo Y2K ahora Y2038, amenaza a los sistemas Unix/Linux y por consiguiente a todas las comunicaciones en las que intervienen [15]. Particularmente cualquier SSL que sea validado sobre un sistema de 32bits se verá afectado, ya que los certificados tienen fechas de validez que son comparados contra la fecha los de los sistemas operativos.

En nuestro caso forzamos a que los concentradores VPN tengan procesadores 64bits, ya que la última versión de nuestro CD contiene un Linux Debian Jessie preinstalado de arquitectura x86_64. Pero esto no resuelve el problema que existirá con los puestos de trabajo que en la actualidad tienen Windows XP 32bit. Aunque este sistema operativo ya no tiene soporte el mismo es ampliamente usado y según las pruebas realizadas el mismo se verá afectado por Y2038 en el inicio de la conexión OpenVPN. Solucionar este problema no es prioridad hoy en día, sólo se plantea a modo de recordatorio para que se tenga en cuenta en el advenimiento del Y2038.

Conclusión

El presente proyecto fue solamente una parte de la transformación integral del proceso de negocio más importante de la organización. Esta transformación agilizó y aumentó la eficiencia en el desarrollo de las tareas permitiendo incrementar la producción para satisfacer las nuevas necesidades de la organización.

Cabe remarcar que aunque al principio los objetivos del proyecto no estaban del todo claros, la solución implementada demostró la capacidad de escalar para cumplir con la tarea de asegurar las comunicaciones efectuadas por todos los clientes, resguardando la integridad y confidencialidad de los datos que viajan en cada uno de los trámites.

Además de la escalabilidad, se debe destacar la robustez de la solución que mediante la implementación en alta disponibilidad, nos permitió cumplir con estándares de seguridad que den tranquilidad y que demostraron su eficacia brindando un 99,98% de tiempo de servicio en los últimos seis años.

A pesar de lo excéntrico de algunos requerimientos iniciales, mediante desarrollo a medida, se logró satisfacer estos requisitos. Gran parte de este éxito se debe a la gran adaptabilidad que ofrece OpenVPN y su filosofía de software open source, que brinda una flexibilidad que posibilita que sea el usuario quien defina los requerimientos y sea posible adaptar el software para cumplirlos y no al revés. Esto en parte, está relacionado con la ausencia de una relación comercial entre los desarrolladores del producto y los usuarios del mismo. Nobleza obliga, hay que decir que la adaptación corre por cuenta del usuario, lo destacable es que es posible hacerlo y el software open source lo permite.

Más allá de los problemas surgidos con las VPNs en producción, se puede afirmar que la solución alcanzada no requiere prácticamente mantenimiento. Sólo durante la primera y segunda etapa de instalaciones masivas requirió soporte, mantenimiento y modificaciones, pero hace tres años y medio que se encuentra estable, no presenta problemas y el

mantenimiento que exige es mínimo.

Desde lo personal, puedo afirmar que este fue el mayor desafío profesional que tuve en mi carrera laboral. Quienes participamos del proyecto lo hicimos con mucho compromiso y siempre buscando lograr los objetivos e intentando entregar un producto eficiente y seguro que garantice la integridad y confidencialidad de las comunicaciones. También fue parte de nuestro objetivo que la solución sea robusta, estable y que perdure a pesar de posibles cambios en la conducción de la organización. Nuestra predicción al respecto es que esta solución permanecerá en funcionamiento mucho tiempo más.

Anexos

Anexo 1: Estructura de la base de datos y privilegios

Tabla de clientes VPN

```
CREATE TABLE `clivpn` (
  `idvpn` char(9) NOT NULL,
  `startup` int(11) DEFAULT '0',
  `ip_tun` char(15) DEFAULT NULL,
  `cmdclavehw` text NOT NULL,
  `clavehw` text,
  `clavehwmd5` char(32) DEFAULT NULL,
  `shadowmd5` char(32) DEFAULT NULL,
  `clicacrt` text NOT NULL,
  `clicertcrt` text NOT NULL,
  `clicertkey` text NOT NULL,
  `clitakey` text NOT NULL,
  `srvcacrt` text NOT NULL,
  `srvcertcrt` text NOT NULL,
  `srvcertkey` text NOT NULL,
  `srvtakey` text NOT NULL,
  `srvdh` text NOT NULL,
  `subred` varchar(18) NOT NULL,
  `mascara` varchar(15) NOT NULL,
  `preexec` text,
  `postexec` text,
  `fecha` datetime DEFAULT NULL,
  `latitud` double DEFAULT NULL,
  `longitud` double DEFAULT NULL,
  `estado` tinyint(1) DEFAULT '5',
  `ubicacion` varchar(50) DEFAULT NULL,
  `nombre` varchar(50) DEFAULT NULL,
  `direccion` varchar(100) DEFAULT NULL,
  `telefono` varchar(40) DEFAULT NULL,
  `contacto` varchar(30) DEFAULT NULL,
  `version` int(11) DEFAULT '0',
  `ip_local` char(15) DEFAULT NULL,
  `mascara_local` char(15) DEFAULT NULL,
  `gw_local` char(15) DEFAULT NULL,
  `server` char(16) DEFAULT NULL,
  `zona` char(15) DEFAULT NULL,
  `licencia` varchar(14) DEFAULT NULL,
  PRIMARY KEY (`idvpn`),
  KEY `xestado` (`estado`),
  KEY `xnombre` (`nombre`),
  KEY `xzona` (`zona`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Tabla de usuarios

```
CREATE TABLE `usuarios` (
  `usu_id` int(11) NOT NULL AUTO_INCREMENT,
  `usu_nombre` varchar(25) NOT NULL DEFAULT '',
  `usu_pass` char(32) NOT NULL,
  `usu_nombre_real` varchar(50) DEFAULT NULL,
  `usu_apellido_real` varchar(50) DEFAULT NULL,
  `usu_obs` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`usu_id`)
) ENGINE=MyISAM AUTO_INCREMENT=69 DEFAULT CHARSET=utf8;
```

Tabla de auditoría

```
CREATE TABLE `auditoria` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fechahora` datetime DEFAULT NULL,
  `host` char(10) DEFAULT NULL,
  `idvpn` char(9) DEFAULT NULL,
  `accion` varchar(20) DEFAULT NULL,
  `ip_remota` char(16) DEFAULT NULL,
  `ubicacion` varchar(50) DEFAULT NULL,
  `nombre` varchar(50) DEFAULT NULL,
  `duracion` time DEFAULT NULL,
  `bytes` bigint(20) DEFAULT NULL,
  `usuario` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `vpn` (`idvpn`)
) ENGINE=MyISAM AUTO_INCREMENT=13029619 DEFAULT CHARSET=utf8;
```

Vistas

Vista para el proceso de autentificación durante el inicio de conexión, sobre la cual se tiene sólo permiso de *SELECT*:

```
CREATE VIEW `vpns_login_select` AS
select
  `clivpn`.`idvpn` AS `idvpn`,
  `clivpn`.`nombre` AS `nombre`,
  `clivpn`.`ubicacion` AS `ubicacion`,
  `clivpn`.`subred` AS `subred`,
  `clivpn`.`mascara` AS `mascara`,
  `clivpn`.`cmdclavehw` AS `cmdclavehw`,
  `clivpn`.`clavehwmd5` AS `clavehwmd5`,
  `clivpn`.`shadowmd5` AS `shadowmd5`,
  `clivpn`.`version` AS `version`,
  `clivpn`.`startup` AS `startup`,
  `clivpn`.`server` AS `server`
from `clivpn`;
```

Vista para la aplicación web de Gestión de VPNs:

```
CREATE VIEW `vpns_web_admin` AS (
select
  `clivpn`.`idvpn` AS `idvpn`,
  `clivpn`.`ubicacion` AS `ubicacion`,
  `clivpn`.`nombre` AS `nombre`,
  `clivpn`.`direccion` AS `direccion`,
  `clivpn`.`telefono` AS `telefono`,
  `clivpn`.`contacto` AS `contacto`,
  `clivpn`.`subred` AS `subred`,
  `clivpn`.`mascara` AS `mascara`,
  `clivpn`.`estado` AS `estado`,
  `clivpn`.`ip_local` AS `ip_local`,
  `clivpn`.`mascara_local` AS `mascara_local`,
  `clivpn`.`gw_local` AS `gw_local`,
  `clivpn`.`ip_tun` AS `ip_tun`,
  `clivpn`.`latitud` AS `latitud`,
  `clivpn`.`longitud` AS `longitud`,
  `clivpn`.`fecha` AS `fecha`,
  `clivpn`.`cmdclavehw` AS `cmdclavehw`,
  `clivpn`.`clavehw` AS `clavehw`,
  `clivpn`.`clavehwmd5` AS `clavehwmd5`,
  `clivpn`.`startup` AS `startup`,
  `clivpn`.`version` AS `version`,
  `clivpn`.`shadowmd5` AS `shadowmd5`,
  `clivpn`.`zona` AS `zona`,
  `clivpn`.`licencia` AS `licencia`
from `clivpn`);
```

Vista usada por la aplicación de mapa y tablero de estadística:

```
CREATE VIEW `vpns_mapa` AS
select
  `clivpn`.`idvpn` AS `idvpn`,
  `clivpn`.`nombre` AS `nombre`,
  `clivpn`.`ubicacion` AS `ubicacion`,
  `clivpn`.`estado` AS `estado`,
  `clivpn`.`latitud` AS `latitud`,
  `clivpn`.`longitud` AS `longitud`,
  `clivpn`.`ip_tun` AS `ip_tun`,
  `clivpn`.`zona` AS `zona`,
  `clivpn`.`startup` AS `startup`
from `clivpn`;
```

Vista usada exclusivamente durante el proceso de instalación:

```
CREATE VIEW `vpns_setup` AS
select
  `clivpn`.`idvpn` AS `idvpn`,
  `clivpn`.`clavehw` AS `clavehw`,
  `clivpn`.`clavehwmd5` AS `clavehwmd5`,
  `clivpn`.`shadowmd5` AS `shadowmd5`,
  `clivpn`.`fecha` AS `fecha`,
  `clivpn`.`startup` AS `startup`
from `clivpn`;
```

Vista para el proceso de autentificación durante el inicio de conexión, sobre la cual se permite *UPDATE* para registrar los datos del cliente:

```
CREATE VIEW `vpns_login_update` AS
select
  `clivpn`.`idvpn` AS `idvpn`,
  `clivpn`.`estado` AS `estado`,
  `clivpn`.`ip_tun` AS `ip_tun`,
  `clivpn`.`shadowmd5` AS `shadowmd5`,
  `clivpn`.`version` AS `version`,
  `clivpn`.`ip_local` AS `ip_local`,
  `clivpn`.`mascara_local` AS `mascara_local`,
  `clivpn`.`gw_local` AS `gw_local`,
  `clivpn`.`startup` AS `startup`,
  `clivpn`.`server` AS `server`
from `clivpn`;
```

Privilegios de usuarios de base de datos

```
-- Grants for 'gestionvpn'@'gestionvpn'
GRANT USAGE ON *.* TO 'gestionvpn'@'gestionvpn' IDENTIFIED BY
PASSWORD '*6764CFDBD3E8A170065A44A0108EA40F7D16A5B5';
GRANT INSERT, SELECT ON `vpns`.`auditoria` TO
'gestionvpn'@'gestionvpn';
GRANT SELECT, UPDATE ON `vpns`.`usuarios` TO
'gestionvpn'@'gestionvpn';
GRANT SELECT, UPDATE ON `vpns`.`vpns_web_admin` TO
'gestionvpn'@'gestionvpn';

-- Grants for 'mapa'@'gestionvpn'
GRANT USAGE ON *.* TO 'mapa'@'gestionvpn' IDENTIFIED BY PASSWORD
'*C91F74B2041F3C98CB77B87C25505768328107F3';
GRANT SELECT ON `vpns`.`vpns_mapa` TO 'mapa'@'gestionvpn';

-- Grants for 'openvpn'@'vpn'
GRANT USAGE ON *.* TO 'openvpn'@'vpn' IDENTIFIED BY PASSWORD
'*48A469484974E186098A9D85B5AEBB48595B9189';
GRANT INSERT ON `vpns`.`auditoria` TO 'openvpn'@'vpn';
GRANT SELECT ON `vpns`.`vpns_login_select` TO 'openvpn'@'vpn';
GRANT UPDATE ON `vpns`.`vpns_login_update` TO 'openvpn'@'vpn';

-- Grants for 'vpncreator'@'gestionvpn'
GRANT USAGE ON *.* TO 'vpncreator'@'gestionvpn' IDENTIFIED BY
PASSWORD '*245FC4AA19FC6A2A88C0E9FAC13E3BD774ECCFA6';
GRANT INSERT, SELECT, UPDATE ON `vpns`.`clivpn` TO
'vpncreator'@'gestionvpn';

-- Grants for 'vpnsetup'@'localhost'
GRANT USAGE ON *.* TO 'vpnsetup'@'localhost' IDENTIFIED BY PASSWORD
'*9F5C7A9E5CB7C98B8083606AA410EBB2785106BC';
GRANT SELECT ON `vpns`.`clivpn` TO 'vpnsetup'@'localhost';
GRANT SELECT, UPDATE ON `vpns`.`vpns_setup` TO
'vpnsetup'@'localhost';

-- Grants for 'ha'@'localhost'
GRANT USAGE ON *.* TO 'ha'@'localhost' IDENTIFIED BY PASSWORD
```

```
'*5A8D6C1558EFB21D45C819C77324BB480910AAA2';
GRANT SELECT ON `vpns`.`vpns_mapa` TO 'ha'@'localhost';

-- Grants for 'replication'@'vpnapp01'
GRANT REPLICATION SLAVE ON *.* TO 'replication'@'vpnapp01'
IDENTIFIED BY PASSWORD '*3ECFE4DB43096D1AE9F8C8C416D9ACEE59966346';

-- Grants for 'replication'@'vpnapp02'
GRANT REPLICATION SLAVE ON *.* TO 'replication'@'vpnapp02'
IDENTIFIED BY PASSWORD '*3ECFE4DB43096D1AE9F8C8C416D9ACEE59966346';
```

Anexo 2: Scripts de creación de una VPN

Script python para crear nuevo cliente /usr/local/bin/crearVPN.py

Este script se ubica en el servidor de gestión de VPN en /usr/local/bin/crearVPN.py y es llamado desde la aplicación web de gestión con el uso de sudo, por lo que corre como usuario root.

```
1.      #!/usr/bin/python
2.
3.      import sys, os, MySQLdb, MySQLdb.cursors, getopt, re,
  ipaddr, random
4.
5.      debug = 0
6.      DBserver = 'vpnapp'
7.      DBuser = 'vpncreator'
8.      DBpass = 'VPNCreate!Now'
9.      DBname = 'vpns'
10.
11.     def CrearLicencia():
12.         chars = 'ABCDEFGHIJKLMNPQRSTUVWXYZ123456789'
13.         a = ''.join(random.choice(chars) for _ in range(4))
14.         b = ''.join(random.choice(chars) for _ in range(4))
15.         c = ''.join(random.choice(chars) for _ in range(4))
16.         return a+"-"+b+"-"+c
17.
18.     def CodVerificador(vpnname):
19.         if len(vpnname) != 7:
20.             raise "El parametro debe ser una cadena de 8
  caracteres"
21.         CUIT_COEFS = (2, 8, 4, 7, 6, 3, 5)
22.         res = 0
23.         for i in range(0, 7):
24.             if vpnname[i].isdigit():
25.                 res += int(vpnname[i]) * CUIT_COEFS[i]
26.             else:
27.                 res += ord(vpnname[i]) * CUIT_COEFS[i]
28.         res = 11 - res % 11
29.         if res > 8:
30.             res = 0
31.         return res
32.
33.     if __name__ == "__main__":
34.         Empty = re.compile('^[\t ]*(#.*)?$')
35.         IPAddr = re.compile('^((25[0-5]|2[0-4][0-9]|[01]?[0-9]
  [0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$')
36.         Mascaras = ['255.255.0.0', '255.255.128.0',
  '255.255.192.0', '255.255.224.0', '255.255.240.0',
  '255.255.248.0', '255.255.252.0', '255.255.254.0',
  '255.255.255.0', '255.255.255.128', '255.255.255.192',
  '255.255.255.224', '255.255.255.240', '255.255.255.248',
  '255.255.255.252']
37.         zona = dict()
38.         zona['1'] = 'NACIONAL'
```

```
39.         zona['2'] = 'REGIONAL'
40.         zona['3'] = 'INTERNACIONAL'
41.         try:
42.             opciones, argumentos = getopt.getopt(sys.argv[1:],
  'c:s:m:u:n:z:')
43.         except:
44.             print 'Argumentos erroneos'
45.         vpnname = ''
46.         subred = ''
47.         mascara = '255.255.255.0'
48.         for opt, arg in opciones:
49.             if opt == '-u':
50.                 ubicacion = arg
51.             elif opt == '-n':
52.                 nombre = arg
53.             elif opt == '-z':
54.                 zona = arg
55.         db = MySQLdb.connect(host = DBserver, user = DBuser,
  passwd = DBpass, db = DBname, cursorclass =
  MySQLdb.cursors.DictCursor)
56.         cursor = db.cursor()
57.         sql = """select max(idvpn) as idvpn, subred, mascara
  from clivpn where idvpn like 'vpn%'"""
58.         cursor.execute(sql)
59.         DatosDB = cursor.fetchone()
60.         if DatosDB['idvpn'] == None:
61.             nrovpn = 1
62.             subred = '10.64.0.0'
63.             mascara = '255.255.255.0'
64.         else:
65.             nrovpn = int(DatosDB['idvpn'][4:7])+1
66.             subred = DatosDB['subred']
67.             mascara = DatosDB['mascara']
68.         vpnname = 'vpn%04d' % nrovpn
69.         vpnname = vpnname+str(CodVerificador(vpnname))
70.         red = ipaddr.IP(subred+'/'+mascara)
71.         subrednum = red.ip+256
72.         subred = ipaddr.IP(subrednum).ip_ext
73.         poste = ""
74.         cmd = "sudo lshw -quiet -C network |egrep \\'(product|
  serial|*-network)\\'"
75.         licencia = CrearLicencia()
76.         if debug:
77.             cert = "Certificado"
78.             key = "Llave"
79.         else:
80.             os.system('/vpn/ssl/cert.sh '+vpnname+' >/dev/null
  2>&1 ')
81.             clica = open('/vpn/ssl/keys/ca.crt', 'r').read()
82.             clicert = open('/vpn/ssl/keys/'+vpnname+'.crt',
  'r').read()
83.             clikey = open('/vpn/ssl/keys/'+vpnname+'.key',
  'r').read()
84.             clita = open('/vpn/ssl/keys/ta-concentrador.key',
  'r').read()
85.             os.system('/vpn/puestos/crearca.sh '+vpnname+'
  >/dev/null 2>&1 ')
```

```
86.             srvca = open('/vpn/puestos/keys/'+vpnname+'/ca.crt',
  'r').read()
87.             srvdh =
  open('/vpn/puestos/keys/'+vpnname+'/dh1024.pem', 'r').read()
88.             srvcert =
  open('/vpn/puestos/keys/'+vpnname+'/servidor.crt', 'r').read()
89.             srvkey =
  open('/vpn/puestos/keys/'+vpnname+'/servidor.key', 'r').read()
90.             srvta = open('/vpn/puestos/ta-puestos.key',
  'r').read()
91.             shadowcd = '76a191a556018fe3eb8ee56b203f07'
92.         sql = """insert into clivpn (idvpn, nombre, ubicacion,
  cmdclavehw, clicacrt, clicertcrt, clicertkey, clitakey, subred,
  mascara, estado, postexec, srvcacrt, srvcertcrt, srvcertkey,
  srvdh, srvtakey, shadowmd5, zona, licencia) values ('%s', '%s',
  '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', 5,'%s', '%s',
  '%s', '%s', '%s', '%s', '%s', '%s', '%s')""" % (vpnname, nombre,
  ubicacion, cmd, clica, clicert, clikey, clita, subred, mascara,
  poste, srvca, srvcert, srvkey, srvdh, srvta, shadowcd, zona,
  licencia)
93.         if debug:
94.             open("/tmp/log", "a").write(sql)
95.         else:
96.             cursor.execute(sql)
97.         cursor.close()
98.         db.close()
99.         if not debug:
100.            os.system("""ssh root@vpn -q -i
  /var/www/.ssh/vpnssh.key 'echo iroute %s %s > /etc/openvpn/ccd/
  %s'; /usr/local/bin/sincronizarvpns.sh""" %
  (subred,mascara,vpnname))
101.        print vpnname
```

Script bash /vpn/ssl/cert.sh

Este script se ubica en el servidor de gestión de VPN en /vpn/ssl/cert.sh y es llamado desde el script de creación de cliente.

```
1.      #!/bin/bash
2.
3.      cd /vpn/ssl
4.      . vars
5.      ./build-key --batch $1
```

Script bash /vpn/puestos/crearca.sh

Este script se ubica en el servidor de gestión de VPN en /vpn/puestos/crearca.sh y es llamado desde el script de creación de cliente.

```
1.      #!/bin/bash
2.
3.      cd /vpn/puestos
```

```
4.      source ./vars
5.      export KEY_DIR="/vpn/puestos/keys/$1"
6.      export KEY_OU="$1"
7.      /vpn/puestos/clean-all
8.      "$EASY_RSA/pkitool" --initca $*
9.      /vpn/puestos/build-dh
10.     ./build-key-server --batch servidor
```

Script bash /vpn/puestos/generar.sh

Este script se ubica en el servidor de gestión de VPN en /vpn/puestos/generar.sh y es llamado desde el script de creación de cliente y por la aplicación de gestión para la creación de puestos de trabajo de un cliente.

```
1.      #!/bin/bash
2.
3.      PWD2="/vpn"
4.      if [ $# -lt 2 ]; then
5.            echo "USO: generar.sh <Identificador-de-VPN>
  <Cantidad-de-puestos> [<Numerar-desde>]"
6.      else
7.            if [ -n "$3" ]; then
8.                  PRIMERO=$3
9.                  ULTIMO=$(($3+$2-1))
10.           else
11.                 PRIMERO=1
12.                 ULTIMO=$2
13.           fi
14.           cd "$PWD2/puestos"
15.           . vars
16.           export KEY_DIR="/vpn/puestos/keys/$1"
17.           export KEY_OU="$1"
18.           mkdir -p "$PWD2/puestos/instaladores/$1"
19.           mkdir -p /tmp/puestos/$1
20.           rm -rf /tmp/puestos/$1/*
21.           cp $PWD2/puestos/keys/$1/ca.crt
  $PWD2/openvpn/openvpn/config/ca.crt
22.           cp $PWD2/puestos/ta-puestos.key
  $PWD2/openvpn/openvpn/config/ta.key
23.           for i in `seq -f %02g $PRIMERO $ULTIMO`; do
24.                 $PWD2/puestos/pkitool --batch "puesto-$1-$i"
  > /dev/null 2>&1
25.                 cp $KEY_DIR/puesto-$1-$i.crt
  $PWD2/openvpn/openvpn/config/client.crt
26.                 cp $KEY_DIR/puesto-$1-$i.key
  $PWD2/openvpn/openvpn/config/client.key
27.                 sed "s/remote servidor/remote
  $1.vpns.tesis.org.ar 1194/g" $PWD2/openvpn/vpntesis.ovpn >
  $PWD2/openvpn/openvpn/config/vpntesis.ovpn
28.                 makensis $PWD2/openvpn/openvpn-gui.nsi
  >/dev/null
29.                 mv $PWD2/openvpn/openvpn-2.0.9-gui-1.0.3-
```

```
  install.exe $PWD2/puestos/instaladores/$1/openvpn-gui-$1-$i.exe
30.                 cp $PWD2/puestos/instaladores/$1/openvpn-gui-$1-
  $i.exe /tmp/puestos/$1/
31.           done
32.     fi
```

Anexo 3: Scripts de instalación de concentradores VPN

Script python /var/www/cgi-bin/alta

Este script se ubica en los servidores de instalación en /var/www/cgi-bin/alta y es llamado desde el script de instalación vía peticiones HTTP.

```
1.      #!/usr/bin/python
2.
3.      import sys, urllib2, MySQLdb, pickle, MySQLdb.cursors,
  hashlib, datetime, os
4.
5.      DBserver = 'localhost'
6.      DBuser = 'vpnsetup'
7.      DBpass = 'VPNSetup@CGI'
8.      DBname = 'vpns'
9.
10.     def SrvAlta():
11.         Peticion = sys.stdin.read()
12.         Error = 0
13.         Datos = dict()
14.         Datos['Error'] = list()
15.         Datos['files'] = dict()
16.         DatosPeticion =
  pickle.loads(urllib2.base64.binascii.a2b_base64(urllib2.base64.dec
  odestring(Peticion)))
17.         if not DatosPeticion.has_key('IdVPN'):
18.             Datos['Error'].append('Error 001: No se recibio el
  identificador de VPN.')
19.             Error = 1
20.         elif DatosPeticion['IdVPN'] == '':
21.             Datos['Error'].append('Error 002: No se recibio el
  identificador de VPN.')
22.             Error = 1
23.
24.         if not DatosPeticion.has_key('Licencia'):
25.             Datos['Error'].append('Error 003: No se recibio la
  licencia.')
26.             Error = 1
27.         elif DatosPeticion['Licencia'] == '':
28.             Datos['Error'].append('Error 004: No se recibio la
  licencia.')
29.             Error = 1
30.         if DatosPeticion.has_key('ClaveHW'):
31.             SrvAlta2doPaso(DatosPeticion, Datos, Error)
32.         else:
33.             SrvAlta1erPaso(DatosPeticion, Datos, Error)
34.         Respuesta =
  urllib2.base64.encodestring(urllib2.base64.binascii.b2a_base64(pic
  kle.dumps(Datos)))
35.         print """Content-type: application/binary
36.
37.     %s""" % Respuesta
```

```
38.
39.     def SrvAlta1erPaso(DatosPeticion, Datos, Error):
40.         if Error == 0:
41.             Datos['IdVPN'] = str.lower(DatosPeticion['IdVPN'])
42.             Datos['Licencia'] =
  str.upper(DatosPeticion['Licencia'])
43.             try:
44.                 db = MySQLdb.connect(host = DBserver, user =
  DBuser, passwd = DBpass, db = DBname, cursorclass =
  MySQLdb.cursors.DictCursor)
45.             except:
46.                 Datos['Error'].append('Error 101: Hubo un error
  en el servidor para registrar VPNs.')
47.                 Error = 1
48.             if Error == 1:
49.                 return()
50.             cursor = db.cursor()
51.             sql = """select * from clivpn where idvpn = '%s'"""
  % DatosPeticion['IdVPN']
52.             cursor.execute(sql)
53.             if cursor.rowcount == 0:
54.                 Datos['Error'].append('Error 102: El
  identificador de VPN no existe o es incorrecto.')
55.             else:
56.                 DatosDB = cursor.fetchone()
57.                 if DatosDB['licencia'] != None and
  DatosDB['licencia'] != Datos['Licencia']:
58.                     Datos['Error'].append('Error 104: La
  licencia ingresada no corresponde al identificador de VPN.')
59.                 elif DatosDB['clavehw'] != None and
  DatosDB['clavehw'] != '':
60.                     Datos['Error'].append('Error 103: El VPN ya
  se encuentra registrado, contactese con soporte tecnico.')
61.                 else:
62.                     Datos['cmdClaveHW'] = DatosDB['cmdclavehw']
63.                     if DatosDB['preexec'] != None and
  DatosDB['preexec'] != '':
64.                         Datos['preexec'] = DatosDB['preexec']
65.             cursor.close()
66.             db.close()
67.
68.     def SrvAlta2doPaso(DatosPeticion, Datos, Error):
69.         Datos['IdVPN'] = str.lower(DatosPeticion['IdVPN'])
70.         Datos['Licencia'] = str.upper(DatosPeticion['Licencia'])
71.         if DatosPeticion['ClaveHW'] == '':
72.             Datos['Error'].append('Error 201: No se recibio la
  autenticacion del VPN.')
73.             Error = 1
74.         if not DatosPeticion.has_key('cmdClaveHW'):
75.             Datos['Error'].append('Error 202: Hubo un error con
  autenticacion del VPN.')
76.             Error = 1
77.         elif DatosPeticion['cmdClaveHW'] == '':
78.             Datos['Error'].append('Error 203: Hubo un error con
  autenticacion del VPN.')
79.             Error = 1
80.         else:
```

```
81.             Datos['cmdClaveHW'] = DatosPeticion['cmdClaveHW']
82.         if Error == 0:
83.             try:
84.                 db = MySQLdb.connect(host = DBserver, user =
  DBuser, passwd = DBpass, db = DBname, cursorclass =
  MySQLdb.cursors.DictCursor)
85.             except:
86.                 Datos['Error'].append('Error 204: Hubo un error
  en el servidor para registrar VPNs.')
87.                 Error = 1
88.             if Error == 1:
89.                 return()
90.             cursor = db.cursor()
91.             sql = """select * from clivpn where idvpn = '%s'"""
  % DatosPeticion['IdVPN']
92.             cursor.execute(sql)
93.             if cursor.rowcount == 0:
94.                 Datos['Error'].append('Error 205: El
  identificador de VPN no existe o es incorrecto.')
95.             else:
96.                 DatosDB = cursor.fetchone()
97.                 if DatosDB['licencia'] != None and
  DatosDB['licencia'] != Datos['Licencia']:
98.                     Datos['Error'].append('Error 104: La
  licencia ingresada no corresponde al identificador de VPN.')
99.                 elif DatosDB['clavehw'] != None and
  DatosDB['clavehw'] != '':
100.                    Datos['Error'].append('Error 206: El VPN ya
  se encuentra registrado, contactese con soporte tecnico.')
101.                else:
102.                    if DatosDB['cmdclavehw'] !=
  Datos['cmdClaveHW']:
103.                        Datos['Error'].append('Error 207: La
  autenticacion remitida es invalida.')
104.                    else:
105.                        ClienteConf = """client
106.   dev-type tun
107.   dev vpntesis
108.   proto udp
109.   float
110.   resolv-retry infinite
111.   nobind
112.   mute-replay-warnings
113.   ca /etc/openvpn/cliente/ca.crt
114.   cert /etc/openvpn/cliente/"""+DatosDB['idvpn']+""".crt
115.   key /etc/openvpn/cliente/"""+DatosDB['idvpn']+""".key
116.   tls-auth /etc/openvpn/cliente/ta.key 1
117.   remote-cert-tls server
118.   comp-lzo
119.   verb 4
120.   pull
121.   keepalive 10 30
122.   ping-timer-rem
123.   persist-tun
124.   persist-key
125.   log-append /var/log/openvpn-cliente.log
126.   remote-random
```

```
127.    resolv-retry 60
128.    writepid /var/run/openvpn/cliente.pid
129.
130.    remote vpn1.tesis.org.ar 1194
131.    remote vpn1.tesis.org.ar 1195
132.    remote vpn1.tesis.org.ar 1196
133.    remote vpn1.tesis.org.ar 1197
134.    remote vpn2.tesis.org.ar 1194
135.    remote vpn2.tesis.org.ar 1195
136.    remote vpn2.tesis.org.ar 1196
137.    remote vpn2.tesis.org.ar 1197
138.    """
139.                        ServidorConf = """port 1194
140.    proto udp
141.    dev-type tun
142.    dev vpnlan
143.    ca /etc/openvpn/servidor/ca.crt
144.    cert /etc/openvpn/servidor/vpn-server.crt
145.    key /etc/openvpn/servidor/vpn-server.key
146.    dh /etc/openvpn/servidor/dh1024.pem
147.    tls-auth /etc/openvpn/servidor/ta.key 0
148.    server """+DatosDB['subred']+""" """+DatosDB['mascara']+"""
149.    ifconfig-pool-persist /var/lib/openvpn/ip_databases.txt
150.    keepalive 10 30
151.    comp-lzo
152.    status /var/lib/openvpn/openvpn-status.log
153.    log-append  /var/log/openvpn-servidor.log
154.    verb 3
155.    writepid /var/run/openvpn/servidor.pid
156.
157.    push "route 192.168.3.0 255.255.255.0"
158.    push "route 10.1.0.0 255.255.192.0"
159.    """
160.                        Datos['files']
  ['/etc/openvpn/cliente/'+DatosDB['idvpn']+'.crt'] =
  DatosDB['clicertcrt']
161.                        Datos['files']
  ['/etc/openvpn/cliente/'+DatosDB['idvpn']+'.key'] =
  DatosDB['clicertkey']
162.                        Datos['files']
  ['/etc/openvpn/cliente/ca.crt'] = DatosDB['clicacrt']
163.                        Datos['files']
  ['/etc/openvpn/cliente/ta.key'] = DatosDB['clitakey']
164.                        Datos['files']
  ['/etc/openvpn/cliente.conf'] = ClienteConf
165.                        Datos['files']
  ['/etc/openvpn/servidor/vpn-server.crt'] = DatosDB['srvcertcrt']
166.                        Datos['files']
  ['/etc/openvpn/servidor/vpn-server.key'] = DatosDB['srvcertkey']
167.                        Datos['files']
  ['/etc/openvpn/servidor/dh1024.pem'] = DatosDB['srvdh']
168.                        Datos['files']
  ['/etc/openvpn/servidor/ca.crt'] = DatosDB['srvcacrt']
169.                        Datos['files']
  ['/etc/openvpn/servidor/ta.key'] = DatosDB['srvtakey']
170.                        Datos['files']
  ['/etc/openvpn/servidor.conf'] = ServidorConf
```

```
171.                        if DatosDB['postexec'] != None and
  DatosDB['postexec'] != '':
172.                            Datos['postexec'] =
  DatosDB['postexec']
173.                        m = hashlib.md5()
174.                        m.update(DatosPeticion['ClaveHW'])
175.                        Datos['ClaveHWMD5'] = m.hexdigest()
176.                        sql = """update vpns_setup set clavehw =
  '%s', clavehwmd5 = '%s', fecha = NOW(), startup = 0 where idvpn =
  '%s'""" % (DatosPeticion['ClaveHW'], Datos['ClaveHWMD5'],
  DatosPeticion['IdVPN'])
177.                        cursor.execute(sql)
178.                        if DatosPeticion.has_key('ShadowMD5'):
179.                            sql = """update vpns_setup set
  shadowmd5 = '%s' where idvpn = '%s'""" %
  (DatosPeticion['ShadowMD5'], DatosPeticion['IdVPN'])
180.                            cursor.execute(sql)
181.                        ahora =
  datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
182.                        IP = os.environ['REMOTE_ADDR']
183.                        sql = """insert into
  auditoria(fechahora, host, idvpn, accion, ip_remota) values ('%s',
  'setup', '%s', 'INICIA INSTALACION', '%s')""" % (ahora,
  DatosPeticion['IdVPN'], IP)
184.                        cursor.execute(sql)
185.            cursor.close()
186.            db.close()
187.
188.    if __name__ == "__main__":
189.        SrvAlta()
```

Script vpn iniciado al bootear el CD de instalación

Este script se ubica en el CD booteable de instalación del concentrador VPN. Se ubica en /usr/bin/vpn y es llamado automáticamente al bootear.

```
1.      #!/usr/bin/python
2.
3.      import sys
4.      sys.path.append("/tmp")
5.      import libvpn
6.      if __name__ == '__main__':
7.          libvpn.init()
```

Script librería libvpn.py del instalador del CD de instalación

Este script se ubica en el CD booteable de instalación del concentrador VPN. Se ubica en /usr/lib/python2.7/libvpn.pyc y es llamado por /usr/bin/vpn al iniciarse.

```
1.      #!/usr/bin/python
2.
3.      import os, sys, pickle, urllib2, subprocess, re, getopt
4.
5.      SHADOW = '76a191a55426018fe3eb8ee56b203f07'
6.      MainNS = 'setup.tesis.org.ar'
7.      MainIP = '192.168.2.2'
8.      VPN1NS = 'vpn1.tesis.org.ar'
9.      VPN2NS = 'vpn2.tesis.org.ar'
10.     VPNIP = '192.168.2.2'
11.     vm = 1
12.     Estado = [8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
13.     debug=1
14.
15.     def PrimerHost(IP, Mascara):
16.         from ipaddr import IPv4
17.         return IPv4(IPv4(IP+'/'+Mascara).network+1).ip_ext
18.
19.     def command(cmd):
20.         if debug==1:
21.             os.system("""echo "%s" | grep -v dialog >
  /dev/tty11""" % cmd)
22.         ret=os.system(cmd)
23.         return ret
24.
25.     def CodVerificador(vpnname):
26.         if len(vpnname) != 7:
27.             raise "El parametro debe ser una cadena de 7
  caracteres"
28.         CUIT_COEFS = (2, 8, 4, 7, 6, 3, 5)
29.         res = 0
30.         for i in range(0, 7):
31.             if vpnname[i].isdigit():
32.                 res += int(vpnname[i]) * CUIT_COEFS[i]
33.             else:
34.                 res += ord(vpnname[i]) * CUIT_COEFS[i]
35.         res = 11 - res % 11
36.         if res > 8:
37.             res = 0
38.         return res
39.
40.     def CodVerificadorOk(vpnname):
41.         cod = CodVerificador(vpnname[:7])
42.         if str(cod) == vpnname[7]:
43.             return True
44.         else:
45.             return False
46.
47.     def Alta(MountRoot, Conf):
48.         if Estado[0] == "OK":
49.             check=''
50.             Main = MainNS
51.         else:
52.             check='--no-check-certificate'
53.             Main = MainIP
54.         Exit = 0
55.         IdOk = 0
```

```
56.         txtError = ''
57.         Datos = dict()
58.         Datos['IdVPN'] = ''
59.         Datos['Licencia'] = ''
60.         for i in range(2):
61.             while IdOk == 0:
62.                 msg = 'Ingrese identificador del VPN: '
63.                 command("""dialog --no-cancel --ok-label Aceptar
  --backtitle "Instalacion de VPN Tesis - UBA" --inputbox "%s" 8 50
  --stderr 2> /etc/openvpn/vpnname""" % msg)
64.                 Datos['IdVPN'] = open('/etc/openvpn/vpnname',
  'r').read().lower()
65.                 if len(Datos['IdVPN']) == 8:
66.                     if CodVerificadorOk(Datos['IdVPN']):
67.                         IdOk = 1
68.                     else:
69.                         msg = 'Identificador de VPN erroneo.
  Ingrese identificador del VPN: '
70.                 else:
71.                     msg = 'Identificador de VPN erroneo. Ingrese
  identificador del VPN: '
72.             msg = 'Ingrese el numero de licencia
  correspondiente: '
73.             command("""dialog --no-cancel --ok-label Aceptar
  --backtitle "Instalacion de VPN Tesis - UBA" --inputbox "%s" 8 50
  --stderr 2> /etc/openvpn/licencia""" % msg)
74.             Datos['Licencia'] = open('/etc/openvpn/licencia',
  'r').read().lower()
75.             parametros =
  urllib2.base64.encodestring(urllib2.base64.binascii.b2a_base64(pic
  kle.dumps(Datos)))
76.             encabezados = {"Content-type": "application/x-www-
  form-urlencoded", "Accept": "text/plain"}
77.             cmd="""wget %s --ca-certificate=/root/ca.crt
  --certificate=/root/cliente.crt --private-key=/root/cliente.key -O
  /tmp/resp --post-data='%s' https://%s/cgi-bin/alta > /dev/tty11
  2>&1 """ % (check, parametros, Main)
78.             respuesta = command(cmd)
79.             if respuesta == 0:
80.                 Datos =
  pickle.loads(urllib2.base64.binascii.a2b_base64(urllib2.base64.dec
  odestring(open('/tmp/resp', 'r').read())))
81.                 if len(Datos['Error']) == 1:
82.                     if Datos['Error'][0][:9] == 'Error 206' or
  Datos['Error'][0][:9] == 'Error 103' or Datos['Error'][0][:9] ==
  'Error 104':
83.                         command("""dialog --backtitle
  "Instalacion de VPN Tesis - UBA" --msgbox "%s" 9 60""" %
  Datos['Error'][0])
84.                         IdOk = 0
85.                         continue
86.                 try:
87.                     if len(Datos['Error']) != 0:
88.                         for err in Datos['Error']:
89.                             txtError = txtError + err
90.                         Salida(txtError)
91.                 except:
```

```
92.                     Salida('Respuesta inesperada')
93.                 if Datos.has_key('preexec'):
94.                     command(Datos['preexec'])
95.             else:
96.                 Salida('Error 401: Error en la conexion al
  enviar la informacion al servidor central. Proceso abortado, el
  equipo se reiniciara.')
97.             break
98.         command('hostname '+Datos['IdVPN'])
99.         if debug == 1:
100.            os.system("""echo "cmd: %s" > /dev/tty11""" %
  Datos['cmdClaveHW'])
101.        e = subprocess.Popen(Datos['cmdClaveHW'], shell=True,
  stdout=subprocess.PIPE)
102.        Datos['ClaveHW'] = e.stdout.read()
103.        Datos['ShadowMD5'] = SHADOW
104.        e = subprocess.Popen("""ifconfig eth0 |awk '$1=="inet"
  {print substr($2, 6)}'""", shell=True, stdout=subprocess.PIPE)
105.        Datos['IP'] = e.stdout.read().strip()
106.        parametros =
  urllib2.base64.encodestring(urllib2.base64.binascii.b2a_base64(pic
  kle.dumps(Datos)))
107.        encabezados = {"Content-type": "application/x-www-form-
  urlencoded", "Accept": "text/plain"}
108.        for i in range(1):
109.            cmd = """wget %s --ca-certificate=/root/ca.crt
  --certificate=/root/cliente.crt --private-key=/root/cliente.key -O
  /tmp/resp --post-data='%s' https://%s/cgi-bin/alta > /dev/tty11
  2>&1 """ % (check, parametros, Main)
110.            respuesta = command(cmd)
111.            if respuesta == 0:
112.                Datos =
  pickle.loads(urllib2.base64.binascii.a2b_base64(urllib2.base64.dec
  odestring(open('/tmp/resp', 'r').read())))
113.                try:
114.                    if len(Datos['Error']) != 0:
115.                        for err in Datos['Error']:
116.                            txtError = txtError + err
117.                        Salida(txtError)
118.                except:
119.                    Salida('Respuesta inesperada')
120.                if Datos.has_key('files'):
121.                    Conf['files'] = dict()
122.                    for f in Datos['files']:
123.                         Conf['files'][f] = Datos['files'][f]
124.                if Datos.has_key('postexec'):
125.                     Conf['postexec'] = Datos['postexec']
126.                command('rm /tmp/resp')
127.                break
128.            else:
129.                Salida('Error 402: Error en la conexion al
  enviar la informacion al servidor central. Proceso abortado, el
  equipo se reiniciara.')
130.
131.    def ConfConexion(MountRoot):
132.        global Estado
133.        IPAddr = re.compile('^((25[0-5]|2[0-4][0-9]|[01]?[0-9]
```

```
  [0-9]?)\.){3}(25[0-4]|2[0-4][0-9]|[01]?[0-9][0-9]?)$')
134.         Paso = 1
135.         Fin = 0
136.         Conf = dict()
137.         Conf['dhcp'] = 0
138.         Conf['IP'] = ''
139.         Conf['Mascara'] = '255.255.255.0'
140.         Conf['Router'] = ''
141.         Conf['DNS'] = ''
142.         Conf['Configurado'] = 0
143.         while not Fin:
144.             if Paso == 1:
145.                 Respuesta = command("""dialog --yes-label Si
  --no-label No --backtitle "Instalacion de VPN Tesis - UBA" --yesno
  "Desea intentar configurar la conexion automaticamente (DHCP)? " 8
  50""")
146.                 if Respuesta == 0:
147.                     command('dhcpcd eth0 > /dev/tty11 2>&1')
148.                     command('ifconfig eth0  > /dev/tty11 2>&1')
149.                     e = subprocess.Popen("""ifconfig eth0 |awk
  '$1=="inet" {print substr($2, 6)}'""", shell=True,
  stdout=subprocess.PIPE)
150.                     IP = e.stdout.read()
151.                     e = subprocess.Popen("""ip route|grep
  default|awk '{print $3}'""", shell=True, stdout=subprocess.PIPE)
152.                     Router = e.stdout.read()
153.                     if IPAddr.match(IP) and
  IPAddr.match(Router):
154.                         if IP[:3] != '169':
155.                             Fin = 1
156.                             Conf['Configurado'] = 1
157.                             Conf['dhcp'] = 1
158.                         else:
159.                             command("""dialog --backtitle
  "Instalacion de VPN Tesis - UBA" --msgbox "No se pudo obtener una
  configuracion mediante DHCP" 9 60""")
160.                             Paso = 2
161.                     else:
162.                         command("""dialog --backtitle
  "Instalacion de VPN Tesis - UBA" --msgbox "No se pudo obtener una
  configuracion mediante DHCP" 9 60""")
163.                         Paso = 2
164.                 else:
165.                     Paso = 2
166.             elif Paso == 2:
167.                 ok = 2
168.                 while ok != 0:
169.                     if ok == 2:
170.                         msg = 'Ingrese la IP:'
171.                         ok = 1
172.                     else:
173.                         msg = 'IP erronea, ingrese nuevamente la
  IP:'
174.                     ret = command("""dialog --nocancel --ok-
  label "Siguiente" --extra-button --extra-label "Retroceder"
  --backtitle "Instalacion de VPN Tesis - UBA" --inputbox "%s" 9 60
  "%s" --stderr 2> /tmp/dialog.ans""" % (msg, Conf['IP']))
```

```
175.                     Conf['IP'] = open('/tmp/dialog.ans',
  'r').read()
176.                     if  IPAddr.match(Conf['IP']):
177.                         ok = 0
178.                     else:
179.                         if ret != 0:
180.                             ok = 0
181.                 if ret != 0:
182.                     Paso = 1
183.                 else:
184.                     Paso = 3
185.             elif Paso == 3:
186.                 ok = 2
187.                 while ok != 0:
188.                     if ok == 2:
189.                         msg = 'Ingrese la mascara de subred:'
190.                         ok = 1
191.                     else:
192.                         msg = 'Mascara erronea, ingrese
  nuevamente la mascara de subred:'
193.                     ret = command("""dialog --nocancel --ok-
  label "Siguiente" --extra-button --extra-label "Retroceder"
  --backtitle "Instalacion de VPN Tesis - UBA" --inputbox "%s" 9 60
  "%s" --stderr 2> /tmp/dialog.ans""" % (msg, Conf['Mascara']))
194.                     Conf['Mascara'] = open('/tmp/dialog.ans',
  'r').read()
195.                     if IPAddr.match(Conf['Mascara']):
196.                         ok = 0
197.                     else:
198.                         if ret != 0:
199.                             ok = 0
200.                 if ret != 0:
201.                     Paso = 2
202.                 else:
203.                     Paso = 4
204.             elif Paso == 4:
205.                 ok = 2
206.                 while ok != 0:
207.                     if Conf['Router'] == '':
208.                         Conf['Router'] = PrimerHost(Conf['IP'],
  Conf['Mascara'])
209.                     if ok == 2:
210.                         msg = 'Ingrese la puerta de enlace:'
211.                         ok = 1
212.                     else:
213.                         msg = 'Gateway erroneo, ingrese
  nuevamente la puerta de enlace:'
214.                     ret = command("""dialog --nocancel --ok-
  label "Siguiente" --extra-button --extra-label "Retroceder"
  --backtitle "Instalacion de VPN Tesis - UBA" --inputbox "%s" 9 60
  "%s" --stderr 2> /tmp/dialog.ans""" % (msg, Conf['Router']))
215.                     Conf['Router'] = open('/tmp/dialog.ans',
  'r').read()
216.                     if IPAddr.match(Conf['Router']):
217.                         if Conf['DNS'] == '':
218.                             Conf['DNS'] = Conf['Router']
219.                         ok = 0
```

```
220.                     else:
221.                         if ret != 0:
222.                             ok = 0
223.                 if ret != 0:
224.                     Conf['Router'] = ''
225.                     Conf['DNS'] = ''
226.                     Paso = 3
227.                 else:
228.                     Paso = 5
229.             elif Paso == 5:
230.                 ok = 2
231.                 while ok != 0:
232.                     if ok == 2:
233.                         msg = 'Ingrese el DNS primario:'
234.                         ok = 1
235.                     else:
236.                         msg = 'DNS erroneo, ingrese nuevamente
  el DNS primario:'
237.                         ret = command("""dialog --nocancel --ok-
  label "Siguiente" --extra-button --extra-label "Retroceder"
  --backtitle "Instalacion de VPN Tesis - UBA" --inputbox "%s" 9 60
  "%s" --stderr 2> /tmp/dialog.ans""" % (msg, Conf['DNS']))
238.                     Conf['DNS'] = open('/tmp/dialog.ans',
  'r').read()
239.                     if IPAddr.match(Conf['DNS']):
240.                         ok = 0
241.                     else:
242.                         if ret != 0:
243.                             ok = 0
244.                 if ret != 0:
245.                      Paso = 4
246.                 else:
247.                     cmdret = command("""echo 'nameserver %s'
  > /etc/resolv.conf""" % Conf['DNS'])
248.                     if cmdret != 0:
249.                         command("""dialog --backtitle
  "Instalacion de VPN Tesis - UBA" --msgbox "Ocurrio un problema al
  setear el servidor DNS. Esto puede causar inconvenientes en la
  conectividad" 9 60""")
250.                     cmdret = command("""ifconfig eth0 %s netmask
  %s """ % (Conf['IP'], Conf['Mascara']))
251.                     if cmdret != 0:
252.                         command("""dialog --backtitle
  "Instalacion de VPN Tesis - UBA" --msgbox "Ocurrio un problema al
  setear la IP y Mascara." 9 60""")
253.                     cmdret = command("""ip route add default via
  %s""" % Conf['Router'])
254.                     if cmdret != 0:
255.                         command("""dialog --backtitle
  "Instalacion de VPN Tesis - UBA" --msgbox "Ocurrio un problema al
  setear el servidor Gateway. Esto puede causar inconvenientes en la
  conectividad" 9 60""")
256.                     Estado=[8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
257.                     if TestearConexion(MountRoot):
258.                         Fin = 1
259.                         Conf['Configurado'] = 1
260.                     else:
```

```
261.                         Paso = 6
262.             elif Paso == 6:
263.                 Respuesta = command("""dialog --yes-label Si
  --no-label No --backtitle "Instalacion de VPN Tesis - UBA" --yesno
  "La configuracion realizada no es satisfactoria, desea revisarla?
  " 8 50""")
264.                 if Respuesta == 0:
265.                     Paso = 1
266.                     command('ifconfig eth0 0.0.0.0')
267.                     command('ip route del default')
268.                 else:
269.                     Salida('Proceso abortado. Se reiniciara el
  equipo, por favor retire el disco.')
270.                     Paso = 8
271.             if Paso == 8:
272.                 Fin = 1
273.         return(Conf)
274.
275.     def ShowTest(Estado, Porcentaje):
276.         cmd="""dialog --backtitle "Instalacion de VPN Tesis -
  UBA" \
277.     --mixedgauge "Testeando la conexion, por favor aguarde." 0 0
  %s \
278.     "Testeando el SRV de inst. %s" "%s" \
279.     "Testeando el SRV de inst. por su IP fija %s" "%s" \
280.     "Testeando el VPN %s puerto 1194" "%s" \
281.     "Testeando el VPN %s puerto 1195" "%s" \
282.     "Testeando el VPN %s puerto 1196" "%s" \
283.     "Testeando el VPN %s puerto 1197" "%s" \
284.     "Testeando el VPN %s puerto 1194" "%s" \
285.     "Testeando el VPN %s puerto 1195" "%s" \
286.     "Testeando el VPN %s puerto 1196" "%s" \
287.     "Testeando el VPN %s puerto 1197" "%s" \
288.     """ % (Porcentaje, MainNS, Estado[0], MainIP, Estado[1],
  VPN1NS, Estado[2], VPN1NS, Estado[3], VPN1NS, Estado[4], VPN1NS,
  Estado[5], VPN2NS, Estado[6], VPN2NS, Estado[7], VPN2NS,
  Estado[8], VPN2NS, Estado[9])
289.         command(cmd)
290.         command("sleep 1")
291.
292.     def TestearInstalador(Srv,check):
293.         ret = command("""wget %s --tries=1 --timeout=10 --ca-
  certificate=/root/ca.crt --certificate=/root/cliente.crt
  --private-key=/root/cliente.key -O /tmp/resp -o /tmp/log https://
  %s/ > /dev/tty11 2>&1""" % (check, Srv))
294.         if ret == 0:
295.             txt = open('/tmp/resp', 'r').read()
296.             if txt[0:32] == '5388f60d7695cb57b87c799ee62d20b2':
297.                 return(0)
298.             else:
299.                 return(1)
300.         else:
301.             return(1)
302.
303.     def TestearConectividadUDP(Srv, Port):
304.         ret = command("""nmap -sU "%s" -p %s |grep "udp open"
  > /dev/tty11 2>&1""" % (Srv, Port))
```

```
305.        return(ret)
306.
307.    def TestearConexion(MountRoot):
308.        global Estado
309.        Estado[0] = "En progreso"
310.        ShowTest(Estado, "0")
311.        command("sleep 1")
312.        ret=command("""host -W1 %s > /dev/tty11 2>&1""" %
  MainNS)
313.        if ret == 0:
314.            check = ''
315.            ret = TestearInstalador(MainNS,check)
316.            if ret == 0:
317.                Estado[0] = "OK"
318.            else:
319.                Estado[0] = "FAIL"
320.        else:
321.            Estado[0] = "FAIL"
322.        Estado[1] = "En progreso"
323.        ShowTest(Estado, "10")
324.        check='--no-check-certificate'
325.        ret = TestearInstalador(MainIP,check)
326.        if ret == 0:
327.            Estado[1] = "OK"
328.        else:
329.            Estado[1] = "FAIL"
330.        if Estado[0] == "OK" or Estado[1] == "OK":
331.            Estado[2] = "En progreso"
332.            ShowTest(Estado, "20")
333.            ret = command("""host -W1 %s > /dev/tty11 2>&1""" %
  VPN1NS)
334.            if ret == 0:
335.                ret = TestearConectividadUDP(VPN1NS, 1194)
336.                if ret == 0:
337.                    Estado[2] = "OK"
338.                else:
339.                    Estado[2] = "FAIL"
340.                Estado[3] = "En progreso"
341.                ShowTest(Estado, "30")
342.                ret = TestearConectividadUDP(VPN1NS, 1195)
343.                if ret == 0:
344.                    Estado[3] = "OK"
345.                else:
346.                    Estado[3] = "FAIL"
347.                Estado[4] = "En progreso"
348.                ShowTest(Estado, "40")
349.                ret = TestearConectividadUDP(VPN1NS, 1196)
350.                if ret == 0:
351.                    Estado[4] = "OK"
352.                else:
353.                    Estado[4] = "FAIL"
354.                Estado[5] = "En progreso"
355.                ShowTest(Estado, "50")
356.                ret = TestearConectividadUDP(VPN1NS, 1197)
357.                if ret == 0:
358.                    Estado[5] = "OK"
359.                else:
```

```
360.                    Estado[5] = "FAIL"
361.                ShowTest(Estado, "60")
362.            else:
363.                Estado[2] = "FAIL DNS"
364.                Estado[3] = "FAIL DNS"
365.                Estado[4] = "FAIL DNS"
366.                Estado[5] = "FAIL DNS"
367.                ShowTest(Estado, "60")
368.                command("sleep 3")
369.            ret = command("""host -W1 %s > /dev/tty11 2>&1""" %
  VPN2NS)
370.            if ret == 0:
371.                ret = TestearConectividadUDP(VPN2NS, 1194)
372.                if ret == 0:
373.                    Estado[6] = "OK"
374.                else:
375.                    Estado[6] = "FAIL"
376.                Estado[7] = "En progreso"
377.                ShowTest(Estado, "70")
378.                ret = TestearConectividadUDP(VPN2NS, 1195)
379.                if ret == 0:
380.                    Estado[7] = "OK"
381.                else:
382.                    Estado[7] = "FAIL"
383.                Estado[8] = "En progreso"
384.                ShowTest(Estado, "80")
385.                ret = TestearConectividadUDP(VPN2NS, 1196)
386.                if ret == 0:
387.                    Estado[8] = "OK"
388.                else:
389.                    Estado[8] = "FAIL"
390.                Estado[9] = "En progreso"
391.                ShowTest(Estado, "90")
392.                ret = TestearConectividadUDP(VPN2NS, 1197)
393.                if ret == 0:
394.                    Estado[9] = "OK"
395.                else:
396.                    Estado[9] = "FAIL"
397.                ShowTest(Estado, "100")
398.            else:
399.                Estado[6] = "FAIL DNS"
400.                Estado[7] = "FAIL DNS"
401.                Estado[8] = "FAIL DNS"
402.                Estado[9] = "FAIL DNS"
403.                ShowTest(Estado, "100")
404.                command("sleep 2")
405.            command("sleep 1")
406.            if (Estado[0] == "OK" or Estado[1] == "OK") and
  (Estado[2] == "OK" or Estado[3] == "OK" or Estado[4] == "OK" or
  Estado[5] == "OK" or Estado[6] == "OK" or Estado[7] == "OK" or
  Estado[8] == "OK" or Estado[9] == "OK"):
407.                return(1)
408.            else:
409.                return(0)
410.        else:
411.            if Estado[0] == "OK" or Estado[1] == "OK":
412.                return(1)
```

```
413.            else:
414.                return(0)
415.
416.    def GuardarConfiguracion(MountRoot, Conf):
417.        Conf['IdVPN'] = open('/etc/openvpn/vpnname', 'r').read()
418.        command("cp /etc/openvpn/vpnname
  /mnt/custom/etc/openvpn/")
419.        dhcp = 0
420.        if Conf.has_key('dhcp'):
421.            if Conf['dhcp'] == 1:
422.                open(MountRoot+'/etc/network/interfaces',
  'w').write("""# This file describes the network interfaces
  available on your system
423.    # and how to activate them. For more information, see
  interfaces(5).
424.
425.    # The loopback network interface
426.    auto lo
427.    iface lo inet loopback
428.
429.    # The primary network interface
430.    auto eth0
431.    iface eth0 inet dhcp
432.    """)
433.                dhcp=1
434.        if dhcp==0:
435.            open(MountRoot+'/etc/network/interfaces',
  'w').write("""# /etc/network/interfaces -- configuration file for
  ifup(8), ifdown(8)
436.
437.    # The loopback interface
438.    auto lo
439.    iface lo inet loopback
440.
441.    # The first network card - this entry was created during the
  Debian installation
442.    # (network, broadcast and gateway are optional)
443.
444.    auto eth0
445.    iface eth0 inet static
446.            address %s
447.            netmask %s
448.            gateway %s
449.    """ % (Conf['IP'], Conf['Mascara'], Conf['Router']))
450.            open(MountRoot+'/etc/resolv.conf',
  'w').write('nameserver %s\n' % Conf['DNS'])
451.        if Conf.has_key('files'):
452.            for f in Conf['files']:
453.                open(MountRoot+f, 'w').write(Conf['files'][f])
454.        open(MountRoot+'/etc/hostname',
  'w').write(Conf['IdVPN'])
455.        StrHosts = open(MountRoot+'/etc/hosts', 'r').read()
456.        StrHosts = StrHosts.replace('vpn00000', Conf['IdVPN'])
457.        open(MountRoot+'/etc/hosts', 'w').write(StrHosts)
458.        StrHosts = open(MountRoot+'/etc/postfix/main.cf',
  'r').read()
459.        StrHosts =
```

```
  StrHosts.replace('vpn00000.vpns.tesis.org.ar', Conf['IdVPN']
  +'.vpns.tesis.org.ar')
460.        open(MountRoot+'/etc/postfix/main.cf',
  'w').write(StrHosts)
461.        command('chmod 0600
  /mnt/custom/etc/openvpn/cliente/*.key')
462.        command('umount /mnt/custom')
463.
464.    def InstalarSistema():
465.        cmd = """dialog --backtitle "Instalacion de VPN Tesis -
  UBA" --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0
  0 "Particionado del disco" "En Progreso" "Formateado del disco" 8
  "Montado del disco" 8 "Instalacion del sistema" 8 "Instalacion del
  Grub" 8 "Creacion de particion swap" 8 "Activacion de particion
  swap" 8 ;
466.    echo '1,,83,*' | sfdisk /dev/sda > /dev/tty11 2>&1 ;
467.    sleep 4;
468.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
  --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 8
  "Particionado del disco" "Completo" "Formateado del disco" "En
  Progreso" "Montado del disco" 8 "Instalacion del sistema" 8
  "Instalacion del Grub" 8 "Creacion de particion swap" 8
  "Activacion de particion swap" 8 ;
469.    mkfs.ext4 -L ROOT /dev/sda1 > /dev/tty11 2>&1;
470.    sleep 1;
471.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
  --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 28
  "Particionado del disco" "Completo" "Formateado del disco"
  "Completo" "Montado del disco" "En Progreso" "Instalacion del
  sistema" 8 "Instalacion del Grub" 8 "Creacion de particion swap" 8
  "Activacion de particion swap" 8 ;
472.    mount /dev/sda1 /mnt/custom > /dev/tty11 2>&1;
473.    mount /dev/cdrom /mnt/cdrom -o ro > /dev/tty11 2>&1;
474.    ls /mnt/cdrom > /dev/tty11 2>&1;
475.    sleep 1;
476.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
  --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 33
  "Particionado del disco" "Completo" "Formateado del disco"
  "Completo" "Montado del disco" "Completo" "Instalacion del
  sistema" "En Progreso" "Instalacion del Grub" 8 "Creacion de
  particion swap" 8 "Activacion de particion swap" 8 ;
477.    tar zxf /mnt/cdrom/FS-VPN.tgz -C /mnt/custom > /dev/tty11
  2>&1;
478.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
  --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 57
  "Particionado del disco" "Completo" "Formateado del disco"
  "Completo" "Montado del disco" "Completo" "Instalacion del
  sistema" "Completo" "Instalacion del Grub" "En Progreso" "Creacion
  de particion swap" 8 "Activacion de particion swap" 8 ;
479.    umount /mnt/cdrom > /dev/tty11 2>&1;
480.    sleep 1;
481.    /usr/bin/update-grub.sh > /dev/tty11 2>&1;
482.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
  --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 66
  "Particionado del disco" "Completo" "Formateado del disco"
  "Completo" "Montado del disco" "Completo" "Instalacion del
  sistema" "Completo" "Instalacion del Grub" "Completo" "Creacion de
```

```
 particion swap" "En Progreso" "Activacion de particion swap" 8;
483.    dd if=/dev/zero of=/mnt/custom/swap bs=1M count=512 >
 /dev/tty11 2>&1;
484.    chmod 0600 /mnt/custom/swap
485.    sleep 1;
486.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
 --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 90
 "Particionado del disco" "Completo" "Formateado del disco"
 "Completo" "Montado del disco" "Completo" "Instalacion del
 sistema" "Completo" "Instalacion del Grub" "Completo" "Creacion de
 particion swap" "Completo" "Activacion de particion swap" "En
 Progreso" ;
487.    mkswap /mnt/custom/swap > /dev/tty11 2>&1;
488.    dialog --backtitle "Instalacion de VPN Tesis - UBA"
 --mixedgauge "Instalacion en proceso, por favor aguarde." 0 0 100
 "Particionado del disco" "Completo" "Formateado del disco"
 "Completo" "Montado del disco" "Completo" "Instalacion del
 sistema" "Completo" "Instalacion del Grub" "Completo" "Creacion de
 particion swap" "Completo" "Activacion de particion swap"
 "Completo";
489.    sleep 1;
490.    """
491.        Respuesta = command("""dialog --yes-label Si --no-label
 No --backtitle "Instalacion de VPN Tesis - UBA" --yesno "El
 siguiente paso ELIMINARA TODA LA INFORMACION que pueda contener
 esta PC. Esta seguro que desea continuar?" 9 50""")
492.        if Respuesta == 0:
493.            command(cmd)
494.        else:
495.            Salida('Proceso abortado. Se reiniciara el equipo,
 por favor retire el disco.')
496.
497.    def Salida(msg):
498.        command("""dialog --backtitle "Instalacion de VPN Tesis
 - UBA" --msgbox "%s" 9 60""" % msg)
499.        command('/etc/init.d/halt.sh reboot')
500.        command('umount /mnt/custom')
501.        command('echo reisub > /proc/sysrq-trigger')
502.        while 1:
503.            command("sleep 10")
504.
505.    def init():
506.        opciones, argumentos = getopt.getopt(sys.argv[1:],
 'aict')
507.        salir = 0
508.        command("""echo "Bienvenido a la instalacion del sistema
 VPN..." > /dev/tty11 """)
509.        open('/tmp/c.c', 'w').write("""int main() { unsigned
 char m[2+4], r[] = "\\x0f\\x01\\x0d\\x00\\x00\\x00\\x00\\xc3";
 *((unsigned*)&r[3]) =
510.     (unsigned)m; ((void(*)())&r)(); return (m[5]>0xd0) ? 1 : 0;
 }
511.    """)
512.        command('cc -o /tmp/c /tmp/c.c >/tmp/log.cc')
513.        if command('/tmp/c') == 0:
514.            ok = 1
515.        else:
```

```
516.            ok = 0
517.        command("""rm /tmp/c*""")
518.        e = subprocess.Popen('uname -r', shell=True,
  stdout=subprocess.PIPE)
519.        kernel = e.stdout.read().strip()
520.        if kernel == '2.6.38-std231-i586':
521.            MountRoot = '/mnt/custom'
522.        else:
523.            MountRoot = ''
524.        Configuracion = dict()
525.        if  ('-a', '') in opciones:
526.            Alta(MountRoot, Configuracion)
527.            GuardarConfiguracion(MountRoot, Configuracion)
528.        elif ('-t', '') in opciones:
529.            TestearConexion(MountRoot)
530.        elif ('-i', '') in opciones:
531.            InstalarSistema()
532.        elif ('-c', '') in opciones:
533.            Configuracion = ConfConexion(MountRoot)
534.            GuardarConfiguracion(MountRoot, Configuracion)
535.        else:
536.            os.system("""echo "Preconfiguracion" >
  /dev/tty11""")
537.            Conectado = 0
538.            Configuracion = ConfConexion(MountRoot)
539.            if Configuracion['Configurado'] == 0:
540.                command("""dialog --backtitle "Instalacion de
  VPN Tesis - UBA" --msgbox "Error 301: No se pudo configurar la
  conexion, por favor contactese con soporte tecnico." 9 60""")
541.                sys.exit(0)
542.            else:
543.                Conectado = 1
544.                os.system("""echo "Configuracion=1" >
  /dev/tty11""")
545.            if Conectado:
546.                Alta(MountRoot, Configuracion)
547.                os.system("""echo "Alta=1" > /dev/tty11""")
548.                InstalarSistema()
549.                os.system("""echo "Instalacion=1" >
  /dev/tty11""")
550.                if vm:
551.                   ok = 1
552.                if ok == 1:
553.                    GuardarConfiguracion(MountRoot,
  Configuracion)
554.                    os.system("""echo "GuardaConfig=1" >
  /dev/tty11""")
555.                    if Configuracion.has_key('postexec'):
556.                        command(Configuracion['postexec'])
557.                    Salida('La instalacion se ha realizado
  satisfactoriamente! La maquina se reiniciara a continuacion.')
558.                    salir = 1
559.                else:
560.                    Salida('Error 303: Ocurrio un error en la
  instalacion, por favor contactese con soporte tecnico. Proceso
  abortado, el equipo se reiniciara')
561.            else:
```

```
562.                command("""dialog --backtitle "Instalacion de
  VPN Tesis - UBA" --msgbox "Error 302: No se pudo configurar la
  conexion, por favor contactese con soporte tecnico." 9 60""")
563.                sys.exit(0)
```

Anexo 4: Configuraciones del servidor VPN central

Configuración del servidor OpenVPN

El siguiente archivo de configuración se encuentra en /etc/openvpn/vpn01.conf.

```
port 1194
proto udp
dev-type tun
dev vpn01
ca /etc/openvpn/certs/ca.crt
cert /etc/openvpn/certs/tesis.crt
key /etc/openvpn/certs/tesis.key
dh /etc/openvpn/certs/dh1024.pem
tls-auth /etc/openvpn/certs/ta.key 0
server 10.1.0.0 255.255.240.0
client-config-dir /etc/openvpn/ccd
ping 10
ping-restart 60
comp-lzo
ifconfig-pool-persist /var/lib/openvpn/ip_databases-vpn1.txt
status /var/lib/openvpn/status-vpn1.log
verb 3
script-security 2

management localhost 7501

client-connect /etc/openvpn/connect.py
client-disconnect /etc/openvpn/disconnect.py

crl-verify /etc/openvpn/crl.pem

push "route 192.168.3.0 255.255.255.0"
push "route 10.1.0.0 255.255.0.0"
```

El siguiente archivo de configuración se encuentra en /etc/openvpn/vpn02.conf.

```
port 1195
proto udp
dev-type tun
dev vpn02
ca /etc/openvpn/certs/ca.crt
cert /etc/openvpn/certs/tesis.crt
key /etc/openvpn/certs/tesis.key
dh /etc/openvpn/certs/dh1024.pem
tls-auth /etc/openvpn/certs/ta.key 0
server 10.1.16.0 255.255.240.0
client-config-dir /etc/openvpn/ccd
ping 10
ping-restart 60
```

```
42.     comp-lzo
43.     ifconfig-pool-persist /var/lib/openvpn/ip_databases-vpn2.txt
44.     status /var/lib/openvpn/status-vpn2.log
45.     verb 3
46.     script-security 2
47.
48.     management localhost 7502
49.
50.     client-connect /etc/openvpn/connect.py
51.     client-disconnect /etc/openvpn/disconnect.py
52.
53.     crl-verify /etc/openvpn/crl.pem
54.
55.     push "route 192.168.3.0 255.255.255.0"
56.     push "route 10.1.0.0 255.255.0.0"
```

El siguiente archivo de configuración se encuentra en /etc/openvpn/vpn03.conf.

```
1.      port 1196
2.      proto udp
3.      dev-type tun
4.      dev vpn03
5.      ca /etc/openvpn/certs/ca.crt
6.      cert /etc/openvpn/certs/tesis.crt
7.      key /etc/openvpn/certs/tesis.key
8.      dh /etc/openvpn/certs/dh1024.pem
9.      tls-auth /etc/openvpn/certs/ta.key 0
10.     server 10.1.32.0 255.255.240.0
11.     client-config-dir /etc/openvpn/ccd
12.     ping 10
13.     ping-restart 60
14.     comp-lzo
15.     ifconfig-pool-persist /var/lib/openvpn/ip_databases-vpn3.txt
16.     status /var/lib/openvpn/status-vpn3log
17.     verb 3
18.     script-security 2
19.
20.     management localhost 7503
21.
22.     client-connect /etc/openvpn/connect.py
23.     client-disconnect /etc/openvpn/disconnect.py
24.
25.     crl-verify /etc/openvpn/crl.pem
26.
27.     push "route 192.168.3.0 255.255.255.0"
28.     push "route 10.1.0.0 255.255.0.0"
```

El siguiente archivo de configuración se encuentra en /etc/openvpn/vpn04.conf.

```
1.      port 1197
2.      proto udp
3.      dev-type tun
```

```
4.      dev vpn04
5.      ca /etc/openvpn/certs/ca.crt
6.      cert /etc/openvpn/certs/tesis.crt
7.      key /etc/openvpn/certs/tesis.key
8.      dh /etc/openvpn/certs/dh1024.pem
9.      tls-auth /etc/openvpn/certs/ta.key 0
10.     server 10.1.48.0 255.255.240.0
11.     client-config-dir /etc/openvpn/ccd
12.     ping 10
13.     ping-restart 60
14.     comp-lzo
15.     ifconfig-pool-persist /var/lib/openvpn/ip_databases-vpn4.txt
16.     status /var/lib/openvpn/status-vpn4.log
17.     verb 3
18.     script-security 2
19.
20.     management localhost 7504
21.
22.     client-connect /etc/openvpn/connect.py
23.     client-disconnect /etc/openvpn/disconnect.py
24.
25.     crl-verify /etc/openvpn/crl.pem
26.
27.     push "route 192.168.3.0 255.255.255.0"
28.     push "route 10.1.0.0 255.255.0.0"
```

Script de client-connect, connect.py

Este script se ubica en el servidor de VPN en /etc/openvpn/connect.py y es llamado por el OpenVPN cuando un cliente se conecta. El script recibe todos los parámetros a través de variables de entorno.

```
1.      #!/usr/bin/python
2.
3.      import MySQLdb, MySQLdb.cursors, os, time, sys
4.      from datetime import datetime
5.      import subprocess
6.
7.      DBserver = 'vpnapp'
8.      DBuser = 'openvpn'
9.      DBpass = 'OpenVPN@LoginCheck'
10.     DBname = 'vpns'
11.     DebugFlag = 1
12.
13.     def Cmd(Str):
14.        if (DebugFlag):
15.            Log.write(Str+"\n")
16.        os.system(Str)
17.
18.     if (DebugFlag):
19.         Log = file('/var/log/vpn-log', 'a')
20.         sys.stdout = open('/var/log/vpn-stdout', 'a')
21.         sys.stderr = open('/var/log/vpn-stdout', 'a')
22.     IPTun = os.environ.get('ifconfig_pool_remote_ip')
```

```
23.     config = os.environ.get('config')
24.     try:
25.         config = config.split("/")[3].split(".")[0]
26.     except:
27.         config = ''
28.     CN = os.environ.get('common_name')
29.     IPPub = os.environ.get('untrusted_ip')
30.     db = MySQLdb.connect(host = DBserver, user = DBuser, passwd
  = DBpass, db = DBname, cursorclass = MySQLdb.cursors.DictCursor)
31.     cursor = db.cursor()
32.     sql = """select * from vpns_login_select where idvpn =
  '%s'""" % CN
33.     cursor.execute(sql)
34.     if cursor.rowcount == 0:
35.         import sys
36.         sys.exit()
37.     else:
38.         DatosDB = cursor.fetchone()
39.     now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
40.     host = os.uname()[1]
41.     os.system("logger -p local0.info -t %s CONNECT %s IP: %s
  Provincia: %s Nombre: %s" % (CN, config, IPPub,
  DatosDB['ubicacion'], DatosDB['nombre']))
42.     sql = """insert into auditoria (fechahora, host, idvpn,
  accion, ip_remota) values('%s', '%s', '%s', 'CONNECT %s', '%s')
  """ % (now, host, CN, config, IPPub)
43.     cursor.execute(sql)
44.     Cmd("route del -net %s netmask %s " % (DatosDB['subred'],
  DatosDB['mascara']))
45.     Cmd("route add -net %s netmask %s dev %s" %
  (DatosDB['subred'], DatosDB['mascara'], os.environ.get('dev')))
46.     sql = """update vpns_login_update set estado = 7, ip_tun =
  '%s', server = '%s' where idvpn = '%s'""" % (IPTun, config, CN)
47.     cursor.execute(sql)
48.     subprocess.Popen( ('/etc/openvpn/post-connect.py', ),
  close_fds = True).wait()
49.     if (DebugFlag):
50.         Log.close()
```

Script post-connect.py

Este script se ubica en el servidor de VPN en /etc/openvpn/post-connect.py y es llamado por el script connect.py cuando un cliente se conecta. El script recibe todos los parámetros que connect.py recibió de OpenVPN a través de variables de entorno.

```
1.      #!/usr/bin/python
2.
3.      import MySQLdb, MySQLdb.cursors, os, unicodedata, sys
4.      from subprocess import Popen, PIPE
5.      import signal
6.      from datetime import datetime
7.
```

```
8.      DBserver = 'vpnapp'
9.      DBuser = 'openvpn'
10.     DBpass = 'OpenVPN@LoginCheck'
11.     DBname = 'vpns'
12.     rsa_key = '/etc/openvpn/vpnssh.key'
13.     DebugFlag = 1
14.
15.     def Cmd(Str):
16.         if (DebugFlag):
17.             Log.write(Str+"\n")
18.         os.system(Str)
19.
20.     if (DebugFlag):
21.         Log = file('/var/log/vpn-log', 'a')
22.     config = os.environ.get('config')
23.     try:
24.         config = config.split("/")[3].split(".")[0]
25.     except:
26.         config = ''
27.     Cmd("ls")
28.
29.     def Esperar():
30.         count = 0
31.         ret = 256
32.         while ret != 0:
33.             ret = os.system("""ping -c1 -w1 %s >/dev/null
  2>&1""" % os.environ.get('ifconfig_pool_remote_ip'))
34.             count += 1
35.             os.system("""bash -c "sleep 1" """)
36.             if count > 45:
37.                 print "CN: %s - salio por repeticion" %
  os.environ.get('common_name')
38.                 sys.stdout.flush()
39.                 sys.exit(1)
40.
41.     def SSHExec(CmdRemoto):
42.         def TimeOutHandler(signum, frame):
43.             print 'Kill ssh (Timeout)'
44.             sys.stdout.flush()
45.             ssh.kill()
46.             return
47.         ssh = Popen(["ssh", "-q", "-o", "StrictHostKeyChecking =
  no", "-o", "ConnectTimeout = 20", "-i", rsa_key,
  "vpnadmin@"+os.environ.get('ifconfig_pool_remote_ip'), CmdRemoto],
  stdout = PIPE)
48.         signal.signal(signal.SIGALRM, TimeOutHandler)
49.         signal.alarm(300)
50.         if ssh.wait() == 0:
51.             salida = ssh.stdout.readlines()
52.             return salida
53.         else:
54.             print "CN: %s - Error al ejecutar comando remoto" %
  (os.environ.get('common_name'))
55.             sys.exit(1)
56.
57.     def Cmd(Str):
58.         os.system(Str)
```

```
59.
60.     def ActualizarDNS(CN, nombredns, ubicaciondns, IPTun,
  IPLoc):
61.         Cmd("""bash -c "echo -e 'server ns.tesis.org.ar\nupdate
  delete %s.vpns.tesis.org.ar\nupdate add %s.vpns.tesis.org.ar 600
  in a %s\n\n'|nsupdate -k /etc/openvpn/Ktesis.org.ar.key; sleep 1"
  """ % (CN, CN, IPLoc))
62.
63.     def PrimerConexion(CN, IPLoc, IPTun, nombredns,
  ubicaciondns, nombremail, ubicacionmail, cursor):
64.         CmdRemoto = """sudo ifconfig vpntesis|grep 'inet addr'|
  head -1|tr ':' ' '|awk '{print $3}';
65.     sudo ifconfig eth0|grep 'inet addr'|head -1|tr ':' ' '|awk
  '{print $7}';
66.     sudo ip route|grep default|awk '{print $3}';
67.     sudo echo '%s.%s.vpns.tesis.org.ar' > /tmp/hostname;
68.     sudo mv /tmp/hostname /etc/hostname;
69.     sudo sed '2 a\\127.0.1.1\t%s.%s.vpns.tesis.org.ar\t%s'
  /etc/hosts > /tmp/hosts;
70.     sudo mv /tmp/hosts /etc/hosts;
71.     sudo hostname -F /etc/hostname;
72.     """ % (nombredns, ubicaciondns, nombredns, ubicaciondns,
  nombredns)
73.         salida = SSHExec(CmdRemoto)
74.         IPTun = salida[0].strip()
75.         MascLoc = salida[1].strip()
76.         GWLoc = salida[2].strip()
77.         now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
78.         host = os.uname()[1]
79.         IPPub = os.environ.get('untrusted_ip')
80.         sql = """insert into auditoria (fechahora, host, idvpn,
  accion, ip_remota) values('%s', '%s', '%s', 'FIN INSTALACION',
  '%s') """ % (now, host, CN, IPPub)
81.         cursor.execute(sql)
82.         sql = """update vpns_login_update set ip_tun = '%s',
  ip_local = '%s', mascara_local = '%s', gw_local = '%s', startup =
  1, version = 0 where idvpn = '%s'""" % (IPTun, IPLoc, MascLoc,
  GWLoc, CN)
83.         cursor.execute(sql)
84.
85.     def Main():
86.         CN = os.environ.get('common_name')
87.         Esperar()
88.         db = MySQLdb.connect(host = DBserver, user = DBuser,
  passwd = DBpass, db = DBname, cursorclass =
  MySQLdb.cursors.DictCursor)
89.         cursor = db.cursor()
90.         sql = """select * from vpns_login_select where idvpn =
  '%s'""" % CN
91.         cursor.execute(sql)
92.         if cursor.rowcount == 0:
93.             import sys
94.             sys.exit()
95.         else:
96.             DatosDB = cursor.fetchone()
97.         CmdRemoto = """%s | md5sum;
98.         (sudo iptables -L; sudo iptables -t nat -L) | sudo
```

```
  md5sum /etc/passwd /etc/shadow - | md5sum;
99.         sudo ifconfig eth0|grep 'inet addr'|head -1|tr ':' ' '|
  awk '{print $3}';
100.        """ % DatosDB['cmdclavehw']
101.        salida = SSHExec(CmdRemoto)
102.        ClaveHWMD5 = salida[0][:32]
103.        ShadowMD5 = salida[1][:32]
104.        IPLoc = salida[2].strip()
105.        IPTun = os.environ.get('ifconfig_pool_remote_ip')
106.        Version = DatosDB['version']
107.        if ClaveHWMD5 == DatosDB['clavehwmd5']:
108.            nombremail = unicodedata.normalize('NFKD',
  unicode(DatosDB['nombre'], 'iso-8859-1')).encode('ASCII',
  'ignore')
109.            nombredns = nombremail.replace(' ', '').replace('.',
  '').lower()
110.            ubicacionmail = unicodedata.normalize('NFKD',
  unicode(DatosDB['ubicacion'], 'iso-8859-1')).encode('ASCII',
  'ignore')
111.            ubicaciondns = ubicacionmail.replace(' ',
  '').replace('.', '').lower()
112.            if DatosDB['startup'] == 0:
113.                PrimerConexion(CN, IPLoc, IPTun, nombredns,
  ubicaciondns, nombremail, ubicacionmail, cursor)
114.                Version = 0
115.            ActualizarDNS(CN, nombredns, ubicaciondns, IPTun,
  IPLoc)
116.            if ShadowMD5 == DatosDB['shadowmd5']:
117.                Cmd("iptables -D vpn_allows -s %s/%s -j
  from_vpns > /dev/null 2>&1 " % (DatosDB['subred'],
  DatosDB['mascara']))
118.                Cmd("iptables -I vpn_allows -s %s/%s -j
  from_vpns" % (DatosDB['subred'], DatosDB['mascara']))
119.                sql = """update vpns_login_update set estado =
  3, ip_tun = '%s', server = '%s' where idvpn = '%s'""" % (IPTun,
  config, CN)
120.                cursor.execute(sql)
121.            else:
122.                sql = """update vpns_login_update set estado =
  2, ip_tun = '%s', server = '%s' where idvpn = '%s'""" % (IPTun,
  config, CN)
123.                cursor.execute(sql)
124.                os.system("""echo "Shadow Fail en """+CN+""" -
  "`date `|mail -s "Shadow Fail en """+CN+""" "
  adminvpn@tesis.org.ar""")
125.        else:
126.            sql = """update vpns_login_update set estado = 1,
  ip_tun = '%s', server = '%s' where idvpn = '%s'""" % (IPTun,
  config, CN)
127.            cursor.execute(sql)
128.            print "CN: %s --%s--!=--%s--" % (CN, ClaveHWMD5,
  DatosDB['clavehwmd5'])
129.            os.system("""echo "Hard Fail en """+CN+"""
  ("""+DatosDB['ubicacion']+""" - """+DatosDB['nombre']+""")- "`date
  `|mail -s "Hard Fail en """+CN+""" " adminvpns@tesis.org.ar""")
130.
131.    def CreateDaemon():
```

```
132.        """Detach a process from the controlling terminal and
  run it in the
133.        background as a daemon.
134.        """
135.        try:
136.            pid = os.fork()
137.        except OSError, e:
138.            raise Exception, "%s [%d]" % (e.strerror, e.errno)
139.        if (pid == 0):
140.            os.setsid()
141.            try:
142.                pid = os.fork()
143.            except OSError, e:
144.                raise Exception, "%s [%d]" % (e.strerror,
  e.errno)
145.            if (pid == 0):
146.                os.chdir("/")
147.                os.umask(0)
148.            else:
149.                os._exit(0)
150.        else:
151.            os._exit(0)
152.        import resource
153.        maxfd = resource.getrlimit(resource.RLIMIT_NOFILE)[1]
154.        if (maxfd == resource.RLIM_INFINITY):
155.            maxfd = MAXFD
156.        for fd in range(0, maxfd):
157.            try:
158.                os.close(fd)
159.            except OSError:
160.                pass
161.        Loggin = "/var/log/post-connect.log"
162.        os.open(Loggin, os.O_RDWR | os.O_CREAT |os.O_APPEND)
163.        os.lseek(0, 0, os.SEEK_END)
164.        os.dup2(0, 1)
165.        os.dup2(0, 2)
166.        return(0)
167.
168.    if (__name__ == '__main__'):
169.        CreateDaemon()
170.        print '%s: CN: %s - Validacion' %
  (datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
  os.environ.get('common_name'))
171.        Main()
172.       print '%s: CN: %s - Fin Validacion' %
  (datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
  os.environ.get('common_name'))
```

Script de client-disconnect, disconnect.py

Este script se ubica en el servidor de VPN en /etc/openvpn/disconnect.py y es llamado por el OpenVPN cuando un cliente se desconecta. El script recibe todos los parámetros a través de variables de

entorno.

```
1.      #!/usr/bin/python
2.
3.      import MySQLdb, MySQLdb.cursors, os, time, sys
4.      from datetime import datetime
5.
6.      DBserver = 'vpnapp'
7.      DBuser = 'openvpn'
8.      DBpass = 'OpenVPN@LoginCheck'
9.      DBname = 'vpns'
10.     DebugFlag = 1
11.
12.     def Cmd(Str):
13.         if (DebugFlag):
14.             Log.write(Str+"\n")
15.         os.system(Str)
16.
17.     if (DebugFlag):
18.         Log = file('/var/log/vpn-log', 'a')
19.         sys.stdout = open('/var/log/vpn-stdout', 'a')
20.         sys.stderr = open('/var/log/vpn-stdout', 'a')
21.     CN = os.environ.get('common_name')
22.     IPPub = os.environ.get('trusted_ip')
23.     LogSegundos = int(os.environ.get('time_duration'))
24.     LogBytes = os.environ.get('bytes_received')
25.     horas = LogSegundos/3600
26.     minutos = (LogSegundos-(horas*3600))/60
27.     segundos = LogSegundos-(horas*3600)-(minutos*60)
28.     config = os.environ.get('config')
29.     try:
30.         config = config.split("/")[3].split(".")[0]
31.     except:
32.         config = ''
33.     db = MySQLdb.connect(host = DBserver, user = DBuser, passwd
  = DBpass, db = DBname, cursorclass = MySQLdb.cursors.DictCursor)
34.     cursor = db.cursor()
35.     sql = """select * from vpns_login_select where idvpn =
  '%s'""" % CN
36.     cursor.execute(sql)
37.     if cursor.rowcount == 0:
38.         Log.write("Disconnect: VPN: "+CN+" no esta en la BD.")
39.         import sys
40.         sys.exit()
41.     else:
42.         DatosDB = cursor.fetchone()
43.     now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
44.     host = os.uname()[1]
45.     os.system("logger -p local0.info -t %s DISCONNECT %s
  Duracion: %02d:%02d:%02d Bytes recibidos: %s IP: %s Provincia: %s
  Nombre: %s" % (CN, config, horas, minutos, segundos, LogBytes,
  IPPub, DatosDB['ubicacion'], DatosDB['nombre']))
46.     sql = """insert into auditoria (fechahora, host, idvpn,
  accion, ip_remota, duracion, bytes) values('%s', '%s', '%s',
  'DISCONNECT %s', '%s', '%02d:%02d:%02d', %s) """ % (now, host, CN,
  config, IPPub, horas, minutos, segundos, LogBytes)
47.     cursor.execute(sql)
```

```
48.     if (DebugFlag):
49.         Log.write('Disconnect CN:'+CN+"\n")
50.     if config == DatosDB['server']:
51.         Cmd("route del -net %s netmask %s " %
  (DatosDB['subred'], DatosDB['mascara']))
52.         Cmd("iptables -D vpn_allows -s %s/%s -j from_vpns" %
  (DatosDB['subred'], DatosDB['mascara']))
53.         sql = """update vpns_login_update set estado = 0 where
  idvpn = '%s'""" % CN
54.         cursor.execute(sql)
55.     if (DebugFlag):
56.         Log.close()
```

Anexo 5: Herramientas de soporte

Script para validar hardware, updateHard.sh

Este script se ubica en el servidor de gestión de VPN en /usr/local/bin/updateHard.sh y es llamado por la aplicación web de gestión de VPN.

```
1.      #!/bin/bash
2.
3.      if [ -z $1 ]; then
4.          echo "No se especifico el VPN."
5.          exit 1
6.      else
7.          IP=`mysql -h vpnapp -u gestionvpn -pGestionVPN@web2015
  -sN -e "select ip_tun from vpns_web_admin where idvpn='$1'" vpns`
8.          if [ "$IP" == "NULL" ] || [ "$IP" == "" ]; then
9.              echo "No se pudo resolver la ip del VPN $1."
10.             exit 1
11.         fi
12.         if [ -z $2 ]; then
13.             user="consola"
14.         else
15.             user=$2
16.         fi
17.         rand=$((RANDOM%1000))
18.         cmd="sudo lshw -quiet -C network |egrep '(product|
  serial|*-network)'"
19.         /usr/bin/ssh vpnadmin@$IP -i /var/www/.ssh/vpnssh.key
  "$cmd" >/tmp/clave-$rand
20.         if [ ! $? -eq 0 ]; then
21.             echo "No se pudieron obtener datos del VPN $1"
22.             rm -f /tmp/clave-$rand
23.             exit 1
24.         fi
25.         md5=`cat /tmp/clave-$rand|md5sum|awk '{print $1}'`
26.         cmd="sudo lshw -quiet  -C network |egrep \\'(product|
  serial|*-network)\\'"
27.         clave=`cat /tmp/clave-$rand`
28.         /usr/bin/mysql -h vpnapp -u gestionvpn
  -pGestionVPN@web2015 -e "update vpns_web_admin set
  clavehw='$clave', clavehwmd5='$md5', cmdclavehw='$cmd' where
  idvpn='$1' " vpns 2>&1
29.         if [ ! $? -eq 0 ]; then
30.             echo "No se pudieron almacenar los nuevos datos para
  el VPN $1"
31.             rm -f /tmp/clave-$rand
32.             exit 1
33.         else
34.             ssh root@vpn -q -i /var/www/.ssh/vpnssh.key
  "/usr/local/bin/killvpn.sh $1"
35.             rm -f /tmp/clave-$rand
36.             if [ $? -eq 0 ]; then
37.                 echo "Se actualizó el hardware de $1"
```

```
38.             else
39.                 echo "Se actualizó el hardware de $1. Pero no se
  pudo reiniciar su conexión."
40.             fi
41.             logger -p local0.info -t $user AdminVPNs: Se
  actualizaron los datos de hardware del VPN: $1
42.         fi
43.     fi
```

Script para validar las credenciales de usuario, updateShadow.sh

Este script se ubica en el servidor de gestión de VPN en /usr/local/bin/updateShadow.sh y es llamado por la aplicación web de gestión de VPN.

```
1.      #!/bin/bash
2.
3.      if [ -z $1 ]; then
4.          echo "No se especifico el VPN."
5.          exit 1
6.      else
7.          if [ -z $2 ]; then
8.             user="consola"
9.          else
10.             user=$2
11.         fi
12.         cmd="(sudo iptables -L; sudo iptables -t nat -L) | sudo
  md5sum /etc/passwd /etc/shadow - | md5sum"
13.         ip=`/usr/bin/mysql -h vpnapp -sN -u gestionvpn
  -pGestionVPN@web2015 -e "select ip_tun from vpns_web_admin where
  idvpn='$1' " vpns`
14.         md5=`/usr/bin/ssh vpnadmin@$ip -i
  /var/www/.ssh/vpnssh.key "$cmd"|awk '{print $1}'`
15.         if [ ! $? -eq 0 ]; then
16.             echo "No se pudieron obtener datos del VPN $1"
17.             exit 1
18.         fi
19.         /usr/bin/mysql -h vpnapp -u gestionvpn
  -pGestionVPN@web2015 -e "update vpns_web_admin set
  shadowmd5='$md5' where idvpn='$1' " vpns 2>&1
20.         logger -p local0.info -t $user AdminVPNs: Se
  actualizaron los datos de shadow del VPN: $1
21.         if [ ! $? -eq 0 ]; then
22.             echo "No se pudieron almacenar los nuevos datos para
  el VPN $1"
23.             exit 1
24.         else
25.             ssh root@vpn -q -i /var/www/.ssh/vpnssh.key
  "/usr/local/bin/killvpn.sh $1"
26.             if [ $? -eq 0 ]; then
27.                 echo "Se actualizó los datos de usuarios y
  passwords de $1"
28.             else
29.                 echo "Se actualizó los datos de usuarios y
```

```
  passwords de $1. Pero no se pudo reiniciar su conexión."
30.             fi
31.         fi
32.     fi
```

Script para regenerar credenciales, /usr/local/bin/recrearVPN.py

Este script se ubica en el servidor de gestión de VPN en /usr/local/bin/recrearVPN.py y es llamado desde la aplicación web de gestión con el uso de sudo, por lo que corre como usuario root.

```
1.      #!/usr/bin/python
2.
3.      import sys, os, MySQLdb, MySQLdb.cursors, getopt, re,
  ipaddr, random
4.
5.      debug = 0
6.      DBserver = 'vpnapp'
7.      DBuser = 'vpncreator'
8.      DBpass = 'VPNCreate!Now'
9.      DBname = 'vpns'
10.     CAFile = '/vpn/ssl/keys/ca.crt'
11.
12.     def CrearLicencia():
13.         chars = 'ABCDEFGHIJKLMNPQRSTUVWXYZ123456789'
14.         a = ''.join(random.choice(chars) for _ in range(4))
15.         b = ''.join(random.choice(chars) for _ in range(4))
16.         c = ''.join(random.choice(chars) for _ in range(4))
17.         return a+"-"+b+"-"+c
18.
19.     if __name__ == "__main__":
20.         try:
21.             opciones, argumentos = getopt.getopt(sys.argv[1:],
  'c:')
22.         except:
23.             print 'Argumentos erroneos'
24.         vpnname = ""
25.         for opt, arg in opciones:
26.             if opt == '-c':
27.                 vpnname = arg
28.         if vpnname == "":
29.             print "Falta identificador de VPN."
30.             sys.exit(1)
31.         db = MySQLdb.connect(host = DBserver, user = DBuser,
  passwd = DBpass, db = DBname, cursorclass =
  MySQLdb.cursors.DictCursor)
32.         cursor = db.cursor()
33.         os.system('/vpn/ssl/revocar-cert.sh '+vpnname+'
  >/dev/null 2>&1 ')
34.         os.system('mv /vpn/ssl/keys/'+vpnname+'.*
  /vpn/ssl/revocados/ >/dev/null 2>&1 ')
35.         os.system('/vpn/ssl/cert.sh '+vpnname+' >/dev/null 2>&1
  ')
36.         cert = open('/vpn/ssl/keys/'+vpnname+'.crt', 'r').read()
```

```
37.         key = open('/vpn/ssl/keys/'+vpnname+'.key', 'r').read()
38.         licencia = CrearLicencia()
39.         sql = """update clivpn set clicertcrt = '%s', clicertkey
  = '%s', clavehw = NULL, clavehwmd5 = NULL, startup = 0, version =
  0, ip_tun = NULL, cmdclavehw = 'sudo lshw -quiet -C network |egrep
  \\'(product|serial|*-network)\\'', licencia='%s' where idvpn =
  '%s'""" % (cert, key, licencia, vpnname)
40.         cursor.execute(sql)
41.         os.system("""scp -q -i /var/www/.ssh/vpnssh.key
  /vpn/ssl/keys/crl.pem root@vpn:/etc/openvpn/ >/dev/null 2>&1 &
  """)
42.         os.system("""ssh -q -i /var/www/.ssh/vpnssh.key root@vpn
  "/usr/local/bin/sincronizar.sh; /usr/local/bin/killvpn.sh %s" >
  /dev/null 2>&1""" % vpnname)
43.         print vpnname
```

Script para revocar un certificado, /vpn/ssl/revocar-cert.sh

Este script se ubica en el servidor de gestión de VPN en /vpn/ssl/revocar-cert.sh y es llamado por recrearVPN.py para revocar un certificado.

```
1.      #!/bin/bash
2.
3.      cd /vpn/ssl/
4.      . vars
5.      ./revoke-full $1
```

Script para sincronizar los servidores VPN

Este script se ubica en el servidor de VPN en /usr/local/bin/sincronizar.sh y es llamado por el script crearVPN.py y por el script recrearVPN.py para sincronizar la carpeta /etc/openvpn/ccd y la lista de revocacion (CRL) /etc/openvpn/crl.pem.

```
1.      #!/bin/bash
2.
3.      rsync -a -e "ssh -q" /etc/openvpn/ccd/
  vpn01:/etc/openvpn/ccd/
4.      rsync -a -e "ssh -q" /etc/openvpn/ccd/
  vpn02:/etc/openvpn/ccd/
5.      rsync -a -e "ssh -q" /etc/openvpn/crl.pem
  vpn01:/etc/openvpn/
6.      rsync -a -e "ssh -q" /etc/openvpn/crl.pem
  vpn02:/etc/openvpn/
```

Script para sincronizar matar la conexión de un cliente

Este script se ubica en el servidor de VPN en /usr/local/bin/killvpn.sh y es llamado por el script updateHard.sh, updateShadow.sh para que reinicie su conexión y por recrearVPN.py en caso de revocación para que no siga conectado con el viejo certificado El mismo se debe ejecutar en el servidor VPN porque los puertos de gestión de OpenVPN sólo atienden en localhost.

```
#!/bin/bash

(echo -e "kill $1 \nquit\n" | tee >(nc localhost 7501) | tee >(nc localhost 7502) | tee >(nc localhost 7503) | tee >(nc localhost 7504) ) > /dev/null
```

Script de usuario de diagnostico del Concentrador VPN

Este script se ubica en los concentradores de VPN en /usr/local/bin/testVPN.py y es llamado al producirse el ingreso al sistema por consola del usuario “test” con contraseña “test”.

```
#!/usr/bin/python

import os, sys, socket, pwd, time

class bcolors:
    HEADER = chr(27)+'[35m'
    OKBLUE = chr(27)+'[34m'
    OKGREEN = chr(27)+'[32m'
    WARNING = chr(27)+'[33m'
    FAIL = chr(27)+'[31m'
    ENDC = chr(27)+'[0m'
    def disable(self):
        self.HEADER = ''
        self.OKBLUE = ''
        self.OKGREEN = ''
        self.WARNING = ''
        self.FAIL = ''
        self.ENDC = ''

def IsInternetUp(IP,PORT):
    testConn = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    try:
        testConn.connect((IP, PORT))
        CONEXION=1
        testConn.close()
    except:
        CONEXION=0
        testConn.close()
    return CONEXION
```

```
30.
31.     try:
32.         uid=pwd.getpwnam(os.environ.get('USERNAME')).pw_uid
33.     except:
34.         uid=pwd.getpwnam(os.environ.get('USER')).pw_uid
35.
36.     if uid != 0 :
37.         os.system('sudo '+sys.argv[0])
38.     else:
39.         os.system('clear')
40.         archivo='/var/log/diagnostico/diag-'+time.strftime("%Y%m
  %d-%H%M%S")
41.         Log=file(archivo, 'a')
42.         os.system('logger Se realizo un diagnostico. Se guardo
  en '+archivo)
43.
44.         def logprint(mensaje,fd=Log):
45.             print mensaje
46.             fd.write(mensaje+'\n')
47.
48.         ahora=time.strftime("%Y:%m-%d %H:%M:%S")
49.         Log.write('Date time: '+ahora+'\n\n')
50.         HOSTNAME=os.popen('cat /etc/openvpn/vpnname 2>
  /dev/null').read().strip()
51.         logprint(bcolors.HEADER+' '*((80-22-
  len(HOSTNAME))/2)+"IDENTIFICADOR DE VPN:
  "+HOSTNAME+bcolors.ENDC+"\n")
52.         IP=os.popen("""ifconfig eth0|grep "inet addr"|tr ':' '
  '|awk '{print $3}'""").read().strip()
53.         MASK=os.popen("""ifconfig eth0|grep Mask|tr ':' ' '|awk
  '{print $7}'""").read().strip()
54.         GATEWAY=os.popen("""ip route|grep default|awk '{print
  $3}'""").read().strip()
55.         DNS=os.popen("""cat /etc/resolv.conf 2> /dev/null |grep
  nameserver|head -n1|awk '{print $2}'""").read().strip()
56.         PROXY=os.popen('cat /etc/openvpn/cliente.conf
  2>/dev/null|grep "http-proxy "').read().strip()
57.         logprint("IP: "+bcolors.OKBLUE+IP+bcolors.ENDC+" "*(40-
  4-len(IP))+"MASCARA: "+bcolors.OKBLUE+MASK+bcolors.ENDC)
58.         logprint("GATEWAY:
  "+bcolors.OKBLUE+GATEWAY+bcolors.ENDC+" "*(40-9-len(GATEWAY))
  +"DNS: "+bcolors.OKBLUE+DNS+bcolors.ENDC)
59.         if PROXY!='':
60.             logprint("PROXY: "+bcolors.OKBLUE+PROXY.split(' ')
  [1]+":"+PROXY.split(' ')[2]+bcolors.ENDC)
61.         logprint(" ")
62.         if GATEWAY!='':
63.             os.system('ping -w 3 '+GATEWAY+' > /dev/null 2>&1')
64.             GWON=os.popen('arp -n|grep '+GATEWAY+'|grep -v
  incomplete')
65.             if GWON!='':
66.                 logprint("Testeando que el Gateway este
  prendido...                  [      " + bcolors.OKGREEN + "OK" +
  bcolors.ENDC +"      ]")
67.             else:
68.                 logprint("Testeando que el Gateway este
  prendido...                  [     " + bcolors.FAIL + "FAIL" +
```

```
  bcolors.ENDC +"     ]")
69.         else:
70.             logprint("Leyendo el Gateway de la configuracion...
  [     " + bcolors.FAIL + "FAIL" + bcolors.ENDC +"     ]")
71.         HOSTRET=os.system('host %s.vpns.tesis.org.ar %s >
  /dev/null 2>&1' % (HOSTNAME,DNS))
72.         if HOSTRET==0:
73.             logprint("Intentando resolver nombres en el DNS
  configurado...       [      " + bcolors.OKGREEN + "OK" +
  bcolors.ENDC +"      ]")
74.         else:
75.             logprint("Intentando resolver nombres en el DNS
  configurado...       [    " + bcolors.WARNING + "WARNING" +
  bcolors.ENDC +"   ]")
76.         PIDCLI=os.popen('cat /var/run/openvpn/cliente.pid 2>
  /dev/null').read().strip()
77.         if PIDCLI!='':
78.             logprint("Comprobando que el cliente VPN este
  corriendo...           [      " + bcolors.OKGREEN + "OK" +
  bcolors.ENDC +"      ]")
79.         else:
80.             logprint("Comprobando que el cliente VPN este
  corriendo...           [     " + bcolors.FAIL + "FAIL" +
  bcolors.ENDC +"     ]")
81.         TUNCLIIP=os.popen("""ifconfig vpntesis 2> /dev/null|grep
  "inet addr:"|tr ':' ' '|awk '{print $3}'""").read().strip()
82.         if TUNCLIIP!='':
83.             esp=int((14-len(TUNCLIIP))/2)
84.             if esp*2+len(TUNCLIIP)==14:
85.                 msg="
  "*esp+bcolors.OKGREEN+TUNCLIIP+bcolors.ENDC+" "*esp
86.             else:
87.                 msg="
  "*(esp+1)+bcolors.OKGREEN+TUNCLIIP+bcolors.ENDC+" "*esp
88.             logprint("Leyendo la IP asignada al tunel cliente
  VPN...             ["+msg+"]")
89.         else:
90.             logprint("Leyendo la IP asignada al tunel cliente
  VPN...             [     " + bcolors.FAIL + "FAIL" + bcolors.ENDC
  +"     ]")
91.         RUTASVPN=os.popen('ip route|grep -e "dev
  vpntesis"').read().strip()
92.         if RUTASVPN!='':
93.             logprint("Comprobando las rutas del cliente VPN...
  [      " + bcolors.OKGREEN + "OK" + bcolors.ENDC +"      ]")
94.         else:
95.             logprint("Comprobando las rutas del cliente VPN...
  [     " + bcolors.FAIL + "FAIL" + bcolors.ENDC +"     ]")
96.         RUTASSRVTRA=os.popen('ip route|grep -e
  "^192.168.3."').read().strip()
97.         if RUTASSRVTRA!='':
98.             logprint("Comprobando las rutas al servidor de
  tramites...           [      " + bcolors.OKGREEN + "OK" +
  bcolors.ENDC +"      ]")
99.         else:
100.            logprint("Comprobando las rutas al servidor de
  tramites...           [     " + bcolors.FAIL + "FAIL" +
```

```
  bcolors.ENDC +"     ]")
101.
102.        PIDSRV=os.popen('cat /var/run/openvpn/servidor.pid 2>
  /dev/null').read().strip()
103.        if PIDSRV!='':
104.            logprint("Comprobando que el servidor VPN este
  corriendo...          [      " + bcolors.OKGREEN + "OK" +
  bcolors.ENDC +"      ]")
105.        else:
106.            logprint("Comprobando que el servidor VPN este
  corriendo...          [     " + bcolors.FAIL + "FAIL" +
  bcolors.ENDC +"     ]")
107.        TUNSRVIP=os.popen("""ifconfig vpnlan 2> /dev/null|grep
  "inet addr:"|tr ':' ' '|awk '{print $3}'""").read().strip()
108.        if TUNSRVIP!='':
109.            esp=int((14-len(TUNSRVIP))/2)
110.            if esp*2+len(TUNSRVIP)==14:
111.                msg="
  "*esp+bcolors.OKGREEN+TUNSRVIP+bcolors.ENDC+" "*esp
112.            else:
113.                msg="
  "*(esp+1)+bcolors.OKGREEN+TUNSRVIP+bcolors.ENDC+" "*esp
114.            logprint("Leyendo la IP asignada al tunel servidor
  VPN...            ["+msg+"]")
115.        else:
116.            logprint("Leyendo la IP asignada al tunel servidor
  VPN...            [     " + bcolors.FAIL + "FAIL" + bcolors.ENDC
  +"     ]")
117.        RUTASCLI=os.popen('ip route|grep vpnlan').read().strip()
118.        if RUTASCLI!='':
119.            logprint("Comprobando las rutas del servidor VPN...
  [      " + bcolors.OKGREEN + "OK" + bcolors.ENDC +"      ]")
120.        else:
121.            logprint("Comprobando las rutas del servidor VPN...
  [     " + bcolors.FAIL + "FAIL" + bcolors.ENDC +"     ]")
122.        CLIENTES=os.popen("""cat /var/lib/openvpn/openvpn-
  status.log 2> /dev/null |grep -e "^puesto"|sort|tr ':' ' '|awk
  '{print $1}'|sed 's/,/ - IP: /g'""").read().strip()
123.        logprint("\n\nLos clientes conectados son los
  siguientes:")
124.        logprint(CLIENTES)
125.        logprint("\n\n"+"-"*80+"\nSi necesita desplazarse hacia
  arriba puede hacerlo con [SHIFT]+[PGUP]\ny luego hacia abajo
  [SHIFT]+[PGDOWN]")
126.        Log.close()
127.        os.system('bash -c read')
```

Script /usr/local/bin/subirPuestos.sh

Este script se ubica en el servidor de gestión de VPN en /usr/local/bin/subirPuestos.sh y es llamado desde la aplicación web de gestión.

```
1.      #!/bin/bash
2.
3.      IDVPN=$1
4.      PASS=$2
5.      shift 2
6.      LISTA=$*
7.      DIRLOCAL=/vpn/puestospub/$IDVPN
8.      DIRWEB=/var/www/tesis.org.ar/puestos/$IDVPN
9.
10.     rm -rf $DIRLOCAL
11.     mkdir -p $DIRLOCAL
12.     /usr/bin/htpasswd -c -b $DIRLOCAL/htpasswd $IDVPN $PASS
13.     echo $IDVPN $PASS > /tmp/$IDVPN-puestos
14.     echo -e "AuthUserFile $DIRWEB/htpasswd\nAuthType
  Basic\nAuthName \"Puestos VPN\"\nRequire valid-user" >
  $DIRLOCAL/.htaccess
15.     /usr/bin/rar a $DIRLOCAL/$IDVPN.rar -hp$PASS $LISTA
```

Script /usr/local/bin/subirISO.sh

Este script se ubica en el servidor de gestión de VPN en /usr/local/bin/subirISO.sh y es llamado desde la aplicación web de gestión.

```
1.      #!/bin/sh
2.
3.      IDVPN=$1
4.      PASS=$2
5.      DIRLOCAL=/vpn/isospub/$IDVPN
6.      DIRWEB=/var/www/tesis.org.ar/vpns/$IDVPN
7.
8.      rm -rf $DIRLOCAL
9.      mkdir -p $DIRLOCAL
10.     /usr/bin/htpasswd -c -b $DIRLOCAL/htpasswd $IDVPN $PASS
11.     echo "AuthUserFile $DIRWEB/htpasswd\nAuthType
  Basic\nAuthName \"ISO VPN\"\nRequire valid-user" >
  $DIRLOCAL/.htaccess
12.     /usr/bin/genisoimage -o $DIRLOCAL/.$IDVPN.iso -b
  isolinux/isolinux.bin -c boot.cat -no-emul-boot -boot-load-size 4
  -boot-info-table -input-charset iso-8859-1 -l -R -r
  /vpn/geniso/tmp/work-$IDVPN/ 2>/dev/null
13.     mv $DIRLOCAL/.$IDVPN.iso $DIRLOCAL/$IDVPN.iso
14.     rm -rf /vpn/geniso/tmp/work-$IDVPN
```

Anexo 6: Configuración de alta disponibilidad

Configuración crm cib de vpn01 y vpn02

```
1.      node vpn01
2.      node vpn02
3.      primitive failover-ip-dmz ocf:heartbeat:IPaddr2 \
4.            params ip="192.168.2.2" nic="eth0" cidr_netmask="24"
  iflabel="0" \
5.            op monitor interval="10s"
6.      primitive mail ocf:heartbeat:MailTo \
7.            params email="admin@tesis.org.ar" subject="Servicio
  VPN"
8.      primitive openvpn-srv lsb:openvpn \
9.            op monitor interval="15s" timeout="15s" \
10.           meta migration-threshold="3"
11.     primitive pingd_dmz ocf:pacemaker:ping \
12.           params host_list="192.168.1.1" multiplier="150"
  name="pingd_dmz" \
13.           op monitor interval="25s" timeout="60s"
14.     clone pingd_dmz_clone pingd_dmz \
15.           meta globally-unique="false"
16.     location local_dmz_loc failover-ip-dmz \
17.           rule $id="local_location-rule" -inf: not_defined
  pingd_dmz or pingd_dmz lte 0
18.     colocation ip_openvpn inf: failover-ip-dmz openvpn-srv
19.     colocation mailinfo inf: mail failover-ip-dmz
20.     property $id="cib-bootstrap-options" \
21.           cluster-infrastructure="openais" \
22.           expected-quorum-votes="2" \
23.           stonith-enabled="false" \
24.           no-quorum-policy="ignore" \
25.           default-action-timeout="120"
```

Configuración crm cib de vpnapp01 y vpnapp02

```
1.      node vpnapp01
2.      node vpnapp02
3.      primitive failover-ip-dmz ocf:heartbeat:IPaddr2 \
4.            params ip="192.168.2.5" nic="eth0" cidr_netmask="24"
  iflabel="1" \
5.            op monitor interval="10s"
6.      primitive mail ocf:heartbeat:MailTo \
7.            params email="admin@tesis.org.ar" subject="Servicio
  App VPN"
8.      primitive mysql-db ocf:heartbeat:mysql \
9.            params config="/etc/mysql/my.cnf"
  binary="/usr/bin/mysqld_safe" pid="/var/run/mysqld/mysqld.pid"
  socket="/var/run/mysqld/mysqld.sock" test_user="ha"
  test_passwd="saraza" test_table="vpns.vpns_mapa" \
10.           op monitor interval="15s" timeout="5s" \
11.           meta migration-threshold="3"
12.     primitive pingd_dmz ocf:pacemaker:ping \
```

```
13.           params host_list="192.168.1.1" multiplier="150"
  name="pingd_dmz" \
14.           op monitor interval="25s" timeout="60s"
15.     clone mysql-db-clone mysql-db \
16.           meta globally-unique="false"
17.     clone pingd_dmz_clone pingd_dmz \
18.           meta globally-unique="false"
19.     location local_dmz_loc failover-ip-dmz \
20.           rule $id="local_location-rule" -inf: not_defined
  pingd_dmz or pingd_dmz lte 0
21.     colocation ip_mysql inf: failover-ip-dmz mysql-db-clone
22.     colocation mailinfo inf: mail failover-ip-dmz
23.     property $id="cib-bootstrap-options" \
24.           cluster-infrastructure="openais" \
25.           expected-quorum-votes="2" \
26.           stonith-enabled="false" \
27.           default-action-timeout="120" \
28.           no-quorum-policy="ignore"
29.     rsc_defaults $id="rsc-options" \
30.           resource-stickiness="1000"
```

Bibliografía Específica

[1] Eric Rescorla, SSL and TLS: Designing and Building Secure Systems, Addison Wesley Pub Co Inc, 2000
[2] Charlie Hosner, Whitepaper: OpenVPN and the SSL VPN Revolution (2004), http://www.sans.org/reading-room/whitepapers/vpns/openvpn-ssl-vpn-revolution-1459 (consultado el 15/05/2015)
[3] Markus Feilner & Norbert Graf, Beginning OpenVPN 2.0.9, Packt Publishing Ltd, 2006
[4] OpenVPN Security Overview, http://openvpn.net/index.php/documentation/security-overview.html (consultado el 15/05/2015)
[5] EVP Symmetric Encryption and Decryption, https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption (consultado el 12/07/2015)
[6] OpenSSL and EVP, https://www.nlnetlabs.nl/downloads/publications/hsm/hsm_node17.html (consultado el 12/07/2015)
[7] Hardening OpenVPN, https://community.openvpn.net/openvpn/wiki/Hardening (consultado el 12/07/2015)
[8] Simple Stateful Load Balancer with iptables and NAT , https://www.webair.com/community/simple-stateful-load-balancer-with-iptables-and-nat/ (consultado el 12/07/2015)
[9] openvpn(8) - Linux man page, http://linux.die.net/man/8/openvpn (consultado el 25/07/2015)
[10] Andrew Beekhof, Pacemaker 1.1 Clusters from Scratch Creating Active/Passive and Active/Active Clusters on Fedora Edition 8 (2015), http://clusterlabs.org/doc/en-US/Pacemaker/1.1-pcs/pdf/Clusters_from_Scratch/Pacemaker-1.1-Clusters_from_Scratch-en-US.pdf (consultado el 12/07/2015)
[11] Easy RSA - OpenVPN, https://openvpn.net/easyrsa.html (consultado el 15/07/2015)
[12] NSIS Users Manual, http://nsis.sourceforge.net/Docs/ (consultado el 15/07/2015)
[13] Apache HTTP Server Tutorial: .htaccess files, http://httpd.apache.org/docs/2.4/howto/htaccess.html (consultado el 18/07/2015)
[14] Jan Just Keijser, OpenVPN 2 Cookbook, Packt Publishing Ltd, 2011
[15] Y2038 Frequently Asked Questions (FAQ), http://y2038.com/ (consultado el 25/07/2015)

Bibliografía General

- Debian Wiki OpenVPN, https://wiki.debian.org/OpenVPN (consultada el 10/05/2015)
- Jon C. Snader, VPNs Illustrated: Tunnels, VPNs, and Ipsec, Part 2, Section 8.5, Addison Wesley Professional, 2005
- OpenVPN Community Software, https://openvpn.net/index.php/open-source.html (consultada el 10/05/2015)