

# ASIR

# Virtual Private Network

---



## **ÍNDICE**

### **Índice.**

#### **1. Estudio del problema y análisis del sistema.**

- 1.1. Introducción.
- 1.2. Objetivos.
- 1.3. Virtual Private Network.
- 1.4. Funciones y requisitos.
- 1.5. Introducción, evaluación de diversas soluciones y justificación de la solución elegida.
- 1.6. Modelado de la solución, recursos hardware y recursos software.

#### **2. Manual de instalación y fases de pruebas.**

- 2.1. OpenLDAP.
- 2.2. FreeRADIUS.
- 2.3. OpenLDAP + TLS.
- 2.4. FreeRADIUS + OpenLDAP + TLS.
- 2.5. OpenVPN.
- 2.6. Estableciendo el CN del certificado del usuario como nombre de usuario para el login.
- 2.7. Lista de paquetes instalados.

#### **3. Manual de administración.**

- 3.1. Crear un fichero con los ajustes de la conexión a la VPN que los usuarios deberán cargar.
- 3.2. Añadir o eliminar a los usuarios del grupo de LDAP “grupoVPN”.
- 3.3. Creación y revocación de certificados para los usuarios.

#### **4. Manual de usuario.**

#### **5. Funcionamiento de la infraestructura creada.**

#### **6. Planificación del proyecto.**

#### **7. Conclusiones finales.**

- 7.1. Grado de cumplimiento con los objetivos fijados.
- 7.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado.

#### **8. Bibliografía.**

#### **ANEXO-LDAP**

#### **ANEXO-FREERADIUS**

#### **ANEXO-OPENVPN**

## 1. Estudio del problema y análisis del sistema

### **1.1. Introducción**

Actualmente en el ciclo formativo de grado superior de ASIR se usa una red local que ofrece unos servicios determinados a los distintos usuarios, alumnos y profesores. Al tratarse de una red local, los usuarios han de estar físicamente en el centro para poder hacer uso de estos servicios.

Pero supongamos que por cualquier razón se hace necesario el poder hacer uso de estos servicios sin estar físicamente en el centro, es decir, desde otra red como por ejemplo la de casa. En este caso la solución posible es pasar a través de internet.

Esto nos lleva a otro punto a tener en cuenta, la seguridad. Dentro de la red local mantener la seguridad es una tarea relativamente sencilla ya que el administrador posee control total sobre el tráfico que va a circular sobre dicha red. En cambio, al tener que hacer uso de internet para conectar ambas redes, no tendremos el control sobre el tráfico una vez que los datos salgan al exterior desde un extremo y antes de que lleguen al otro extremo.

Por suerte existe una solución para tal situación, crear una **VPN (Virtual Private Network o Red Privada Virtual)**. Esto nos permitirá conectar dos o más puntos de manera segura.

Además, también será necesario controlar el acceso de usuarios, es decir, quienes se van a poder conectar a la VPN y quien no. Para esto vamos a necesitar hacer uso de algún protocolo de tipo AAA (Autenticación, Autorización y Auditoría) como puede ser **RADIUS**.

### **1.2. Objetivo**

El objetivo de la Red Privada Virtual (VPN) de ASIR es permitir a los usuarios acceder de forma segura a los recursos informáticos en red que ofrece el departamento de informática de nuestro instituto, desde cualquier red exterior como por ejemplo la de casa.

Además, se pretende hacer uso de soluciones que implementen la posibilidad de cifrado asimétrico con **Transport Layer Security (TLS)**, que se trata de un protocolo de seguridad que proporciona comunicaciones seguras por una red, comúnmente Internet. Es una de las mejores tecnologías de cifrado para asegurar la identidad de los integrantes de la VPN.

### **1.3. Virtual Private Network**

Una VPN (*Virtual Private Network*), es una red privada que usa una red pública (normalmente Internet) para conectar dos o más puntos de manera segura. Un gran ejemplo de esto sería la posibilidad de conectar dos sucursales sin importar la zona geográfica, utilizando como único medio Internet.

#### **Topologías de una VPN:**

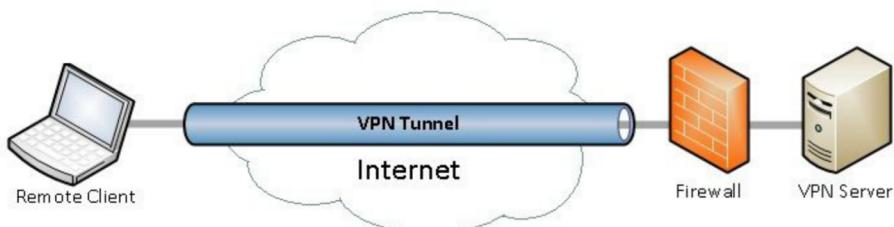
- **VPN de Acceso Remoto:** Es quizás el modelo más usado actualmente y consiste en varios usuarios que se conectan a un servidor remoto utilizando Internet como vínculo de acceso. El cliente se autentica al servidor de acceso remoto, y el servidor se autentica ante el cliente. Una vez autenticados, los clientes tienen un nivel de acceso muy similar al que tendrían si estuvieran conectados a la propia red local donde se encuentra el servidor VPN.

- **VPN Punto-a-Punto:** Este tipo de VPN consiste en proporcionar un “túnel” para la conexión de puntos remotos entre sí. Permite interconectar dos redes LAN geográficamente distantes a través de Internet de manera segura. Este tipo de configuración es ideal para interconectar sucursales de una compañía separadas geográficamente.

Una vez entendidas las diferencias entre las distintas topologías, la que nos interesa para nuestro proyecto es: **VPN de Acceso Remoto**.

### Funcionamiento de una VPN

Una red privada virtual se basa en un protocolo denominado **protocolo de túnel**, es decir, un protocolo encargado de encapsular los paquetes cifrados que se transmiten desde un lado de la VPN hacia el otro.



La palabra "túnel" se usa para simbolizar el hecho que los datos estén cifrados desde el momento que entran a la VPN hasta que salen de ella y, por lo tanto, son incomprendibles para cualquiera que no se encuentre en uno de los extremos de la VPN, es decir, como si los datos viajarán a través de un túnel. Esto quiere decir que además de protocolos de tunelizado, una VPN implementa mecanismos de seguridad para el cifrado de datos como puede ser por ejemplo TLS/SSL.

### Ventajas y desventajas de usar una VPN

Dentro de las ventajas más significativas podemos mencionar:

- Posibilidad de conectar redes físicamente separadas sin necesidad de usar una red dedicada, si no a través de Internet.
- Acceso a los recursos de la red a la cual se conecta el cliente de forma remota.
- La integridad, confidencialidad y seguridad de los datos. Los datos viajan seguros de un extremo a otro de la VPN.

Algunas desventajas pueden ser:

- La velocidad de acceso es generalmente menor a la de una conexión tradicional.
- Es necesario ciertos conocimientos para implementar una conexión VPN.
- Una brecha de seguridad del equipo cliente puede poner en riesgo los recursos de la red a la que se conecta.

### 1.4. Funciones y requisitos

Los servicios que ofrecerá el sistema una vez implementado serán los siguientes:

1. Será posible acceder a la red de ASIR desde una red externa y hacer uso de sus servicios.
2. La conexión entre ambas redes se realizará a través de una VPN.
3. La conexión VPN se realizará de forma segura mediante cifrado, durante toda la sesión de trabajo entre el equipo remoto del usuario y el servidor VPN implantado en la red de ASIR.
4. Los usuarios que podrán conectarse deberán existir en un servicio de directorio LDAP y además pertenecer a un grupo específico en LDAP, al que se le permitirá el acceso a la VPN.

5. Se hará uso de una autenticación de doble factor para realizar la conexión con la VPN:
  - a. Algo que el usuario posee: certificado digital y clave privada.
  - b. Algo que el usuario sabe: usuario/contraseña.
6. Un servidor VPN se encargará de validar los certificados.
7. Un servidor RADIUS será el encargado de la validación de las credenciales usuario/contraseña.

### **1.5. Introducción, evaluación de diversas soluciones y justificación de la solución elegida**

El planteamiento de este proyecto va depender de 3 piedras angulares:

- **Un servidor VPN.**
- **Un servidor RADIUS.**
- **Un servidor LDAP.**

#### **Soluciones para la implementación de un servidor VPN**

Como norma habitual, una VPN se encarga de enlazar dos subredes remotas (implementación punto a punto) o una subred remota y uno o varios usuarios remotos (implementación cliente-servidor), creando para ello un túnel virtual a través del cual serán encapsulados los paquetes antes de ser injectados. Este encapsulado consistirá en una nueva trama de datos, pero esta vez cifrada, la cual podrá albergar protocolos de red de las capas superiores.

En la actualidad existen varias tecnologías que permiten implementar una VPN, haciendo uso de mecanismos diferentes para crear los túneles y la seguridad que ofrecerá.

Algunas tecnologías no están abiertas a los desarrolladores porque son propietarias, como pueden ser muchas de las soluciones VPN que Cisco incorpora en sus dispositivos.

Como tecnologías VPN disponibles gratuitamente y más utilizadas en la actualidad, se compararán PPTP, L2TP/IPSec y OpenVPN.

#### **Cuadro comparativo entre distintas soluciones VPN**

	PPTP	L2TP/IPsec	OpenVPN
<b>Breve descripción</b>	Protocolo VPN propietario de Microsoft, muy básico basado en PPP.	Un avanzado protocolo estandarizado formalmente y recomendado para reemplazar PPTP en plataformas donde se requieran cifrado seguro de datos.	Una avanzada solución VPN de código abierto y que es ahora el estándar de facto en el campo de redes de código abierto.
<b>Protocolos de túnel / cifrado</b>	PPTP / MPPE	L2TP / IPsec	OpenVPN / OpenSSL
<b>Algoritmos de cifrado</b>	RSA RC4 de 128 bits.	3DES ó AES de 256 bits.	RSA RC5, Blowfish, 3DES, AES de 256 bits.
<b>Seguridad e integridad de datos</b>	- Cifrado básico. La seguridad de PPTP está totalmente rota.	- Se considera altamente seguro cuando utiliza el algoritmo de cifrado AES de	- Se considera altamente seguro cuando utiliza el algoritmo de cifrado AES

	<ul style="list-style-type: none"> <li>- No comprueba integridad de datos.</li> </ul>	<p>256 bits.</p> <ul style="list-style-type: none"> <li>- Encapsula los datos dos veces.</li> <li>- Si comprueba la integridad de los datos</li> </ul>	<p>de 256 bits.</p> <ul style="list-style-type: none"> <li>- Uso de certificados digitales con SSL/TLS.</li> <li>- Si comprueba la integridad de los datos.</li> </ul>
<b>Velocidad</b>	Es el que mayor velocidad tiene debido a la menor seguridad que implementa.	Tiene una sobrecarga ligeramente mayor a sus rivales debido al doble encapsulamiento que realiza.	Ofrece un alto rendimiento. Es extremadamente rápido y fiable incluso en conexiones de alta latencia y grandes distancias.
<b>Uso de puertos</b>	PPTP utiliza el puerto TCP 1723 y GRE (protocolo 47). PPTP puede ser fácilmente bloqueado por la restricción del protocolo GRE.	L2TP/IPsec utiliza UDP 500 para el intercambio de claves inicial, el protocolo 50 para los datos cifrados IPsec, UDP 1701 para la configuración inicial L2TP y UDP 4500 para NAT. L2TP/IPsec es más fácil de bloquear que OpenVPN debido a su dependencia de protocolos y puertos fijos.	OpenVPN puede ser fácilmente configurado para ejecutarse en cualquier puerto utilizando UDP o TCP. Para eludir los cortafuegos restrictivos, OpenVPN puede ser configurado para utilizar TCP en el puerto 443.
<b>Estabilidad</b>	PPTP no es tan fiable ni puede recuperarse tan rápidamente como OpenVPN sobre conexiones de red inestables. Problemas de compatibilidad con el protocolo GRE y algunos routers.	Una vez realizada una correcta configuración, será fiable su funcionamiento entre los dispositivos. En la práctica L2TP/IPsec ha demostrado ser tan estable como OpenVPN.	Muy estable y rápido sobre redes inalámbricas no fiables donde es común la pérdida de paquetes y la congestión. OpenVPN tiene un modo TCP para conexiones altamente fiables pero este modo sacrifica algo de velocidad debido a la ineficiencia del encapsulado TCP dentro de TCP.
<b>Compatibilidad</b>	Nativo en la mayoría de los sistemas operativos de dispositivos de sobremesa, portátiles y tablets.	Nativo en la mayoría de los sistemas operativos de dispositivos de sobremesa, portátiles y tablets.	Compatible con la mayoría de los sistemas operativos de dispositivos de sobremesa, portátiles y tablets. Es necesaria una aplicación de terceros para realizar la conexión.
<b>Plataformas</b>	Multiplataforma.	Multiplataforma.	Multiplataforma.

La opción elegida entre las tres analizadas para implantar un servidor VPN es **OpenVPN**.

OpenVPN es la mejor opción para todas las plataformas. Es extremadamente seguro, rápido, fiable y altamente configurable. Utiliza tecnologías de código abierto como la biblioteca de cifrado OpenSSL. Dispone de un algoritmo de cifrado seguro y puede ser configurado para usar cualquier puerto, lo que hace que sea difícil de bloquear por los cortafuegos.

Descartando PPTP completamente por su débil seguridad, entre L2TP/IPsec y OpenVPN, nos quedamos con OpenVPN ya que L2TP/IPsec es más lento debido a su doble encapsulamiento y además puede ser bloqueado con más facilidad debido a la dependencia de uso de puertos y protocolos fijos.

Además, OpenVPN es compatible con NAT, es decir, ambos extremos pueden ser redes NATeadas sin ningún problema.

El único inconveniente de OpenVPN es la necesidad de instalar el software para el cliente, pero en la mayoría de las plataformas esto solo toma unos minutos.

### **Solución para la implementación de un servidor RADIUS**

RADIUS (acrónimo en inglés de Remote Authentication Dial-In User Server). Es un protocolo de autenticación, autorización y auditoría (AAA) para aplicaciones de acceso a la red o movilidad IP. Utiliza el puerto 1812 UDP para establecer sus conexiones.

La autenticación gestionada por este protocolo se realiza a través del ingreso de un nombre de usuario y una clave de acceso. Esta información es procesada por un dispositivo NAS (Network Access Server) siendo posteriormente validada por un servidor RADIUS a través del protocolo correspondiente valiéndose de diversos mecanismos de autenticación y permitiendo el acceso al sistema.

La opción elegida para implantar un servidor RADIUS es **FreeRADIUS** por tratarse de uno de los servidores RADIUS más populares y de libre distribución. Es simple, bastante flexible ya que puede interactuar con distintos almacenes de datos y ofrece soporte para diversos mecanismos de autenticación. Es decir, FreeRADIUS permite entre otras cosas, distintas formas de autenticación y entre ellas está **realizar la autenticación contra un servidor LDAP**. Además en un futuro lo podremos usar, aparte de para la VPN, para controlar el acceso Wifi en el centro.

Cuenta con una gran comunidad que lo sustenta. Desde sus inicios, el proyecto ha crecido hasta incluir soporte para más tipos de autenticación que cualquier otro servidor de código abierto. Se utiliza a diario por 100 millones de personas. Hay más de 50.000 sitios que utilizan FreeRADIUS, que van en tamaño de 10 usuarios a más de 10 millones de usuarios.

### **Solución para la implementación de un servidor LDAP**

LDAP son las siglas de Lightweight Directory Access Protocol (en español Protocolo Ligero/Simplificado de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también se considera una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Está basado en el estándar X.500 para compartir directorios, pero es menos complejo e intensivo en el uso de recursos. Como X.500, LDAP organiza la información en un modo jerárquico usando directorios. Estos directorios pueden almacenar una gran variedad de información y se pueden incluso usar de forma similar al Servicio de Información de Red (NIS), permitiendo que cualquiera pueda acceder a su cuenta desde cualquier máquina en la red acreditada con LDAP.

La mayor ventaja de LDAP es que se puede consolidar información para toda una organización dentro de un repositorio central. Por ejemplo, en vez de administrar listas de usuarios para cada grupo dentro de una organización, puede usar LDAP como directorio central, accesible desde cualquier parte de la red. Puesto que LDAP soporta la Capa de conexión segura (SSL) y la Seguridad de la capa de transporte (TLS), los datos confidenciales se pueden proteger de los curiosos.

La opción elegida es **OpenLDAP**. Se trata de una implementación libre y de código abierto del protocolo Lightweight Directory Access Protocol (LDAP) que está respaldada por una gran comunidad. OpenLDAP incluye un número de características importantes:

- Soporte LDAPv3 — OpenLDAP soporta la Capa de autenticación y seguridad (SASL), la Seguridad de la capa de transporte (TLS) y la Capa de conexión segura (SSL), entre otras mejoras. Muchos de los cambios en el protocolo desde LDAPv2 han sido diseñados para hacer LDAP más seguro.
- Soporte IPv6 — OpenLDAP soporta la próxima generación del protocolo de Internet versión 6.
- LDAP sobre IPC — OpenLDAP se puede comunicar dentro de un sistema usando comunicación interproceso (IPC). Esto mejora la seguridad al eliminar la necesidad de comunicarse a través de la red.
- API de C actualizada — Mejora la forma en que los programadores se conectan para usar servidores de directorio LDAP.
- Soporte LDIFv1 — Provee compatibilidad completa con el formato de intercambio de datos, Data Interchange Format (LDIF) versión 1.
- Servidor Stand-Alone mejorado — Incluye un sistema de control de acceso actualizado, conjunto de hilos, herramientas mejoradas y mucho más.

### 1.6. Modelado de la solución, recursos hardware y recursos software

La infraestructura que se va a implantar para dotar de una VPN a la red de ASIR depende principalmente de las siguientes soluciones software:

- **OpenVPN**: encargado de crear una red privada que use una red pública (Internet) para conectar dos o más puntos de manera segura, además, validará el primer factor necesario para la autenticación: certificados y claves privadas.
- **FreeRADIUS**: encargado de validar el segundo factor de la autenticación para poder permitir el acceso de usuarios a la VPN: usuario/contraseña contra LDAP
- **OpenLDAP**: encargado de almacenar y gestionar una base de datos centralizada que guardará, entre otras, cosas las credenciales de los usuarios que deberán ser validadas y podrán conectarse a la VPN.

Tomando como referencia la manera que está estructurada la red de ASIR, vamos a simular los servidores necesarios con máquinas virtuales de VirtualBox.

Para la virtualización de los servidores necesitaremos un equipo que deberá cumplir los requisitos mínimos del sistema operativo que tenga instalado y que hará de anfitrión, sumados a los requisitos mínimos del sistema operativo huésped que vayamos a virtualizar con VirtualBox. En mi caso, utilizaré mi equipo portátil que consta de:

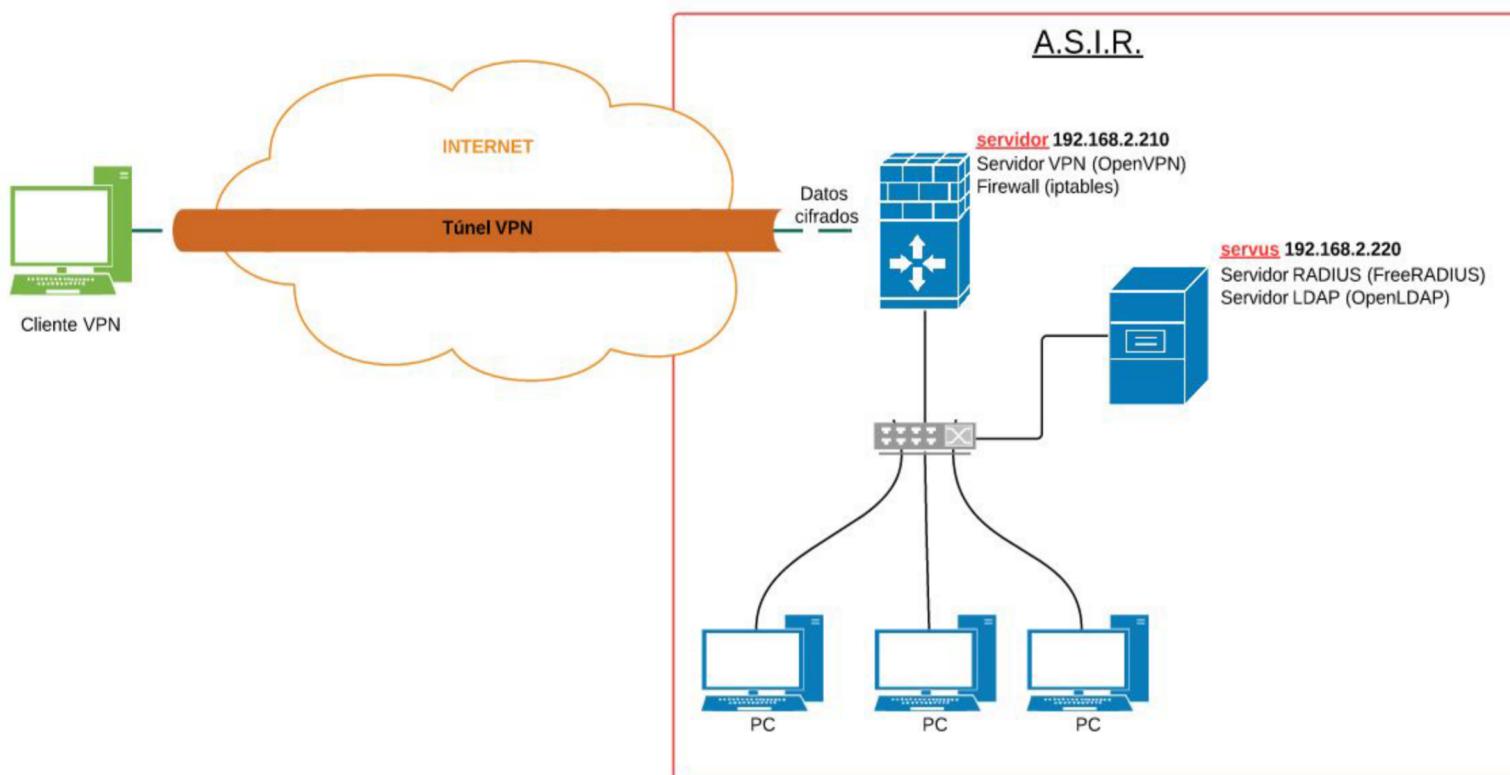
- **Procesador**: Intel Core i3-3217u @ 1,8GHz.
- **Memoria RAM**: 6GB DDR3

- **Tarjeta gráfica:** Nvidia Geforce 820M 1GB VRAM dedicados.
- **Disco duro:** SATA 500GB
- **Sistema operativo:** Xubuntu 14.04 Desktop

Necesitaremos 2 máquinas virtuales de VirtualBox con las siguientes características:

- **servidor:** en él instalaremos **OpenVPN** y constará de:
  - Memoria RAM: 2 GB.
  - Disco duro: 10 GB.
  - Interfaz de red: Adaptador puente con el equipo anfitrión.
  - Sistema operativo: Ubuntu Server 14.04.
  - Le daremos la IP 192.168.2.210.
- **servus:** en él instalaremos **OpenLDAP** y **FreeRADIUS**, constará de:
  - Memoria RAM: 2 GB.
  - Disco duro: 10 GB.
  - Interfaz de red: Adaptador puente con el equipo anfitrión.
  - Sistema operativo: Ubuntu Server 14.04.
  - Le daremos la IP 192.168.2.220.

Para verlo todo más claro, podemos observar el siguiente esquema que describe gráficamente cómo quedaría la infraestructura del proyecto que tenemos entre manos.



## 2. Manual de instalación

El orden que se ha seguido para la instalación de todos los componentes necesarios para la total implementación de la infraestructura en la que se basa proyecto ha sido la siguiente:

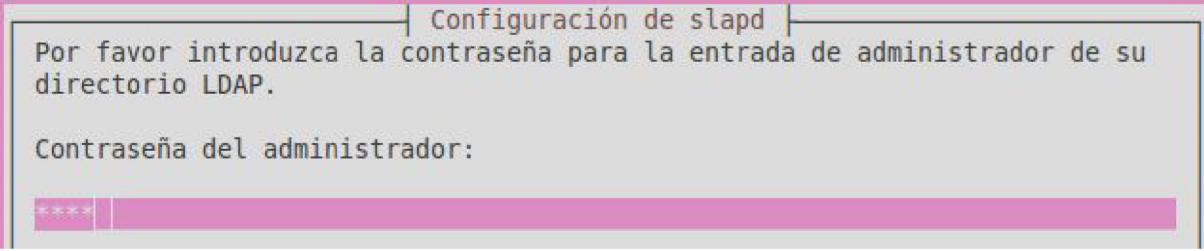
1. OpenLDAP.
2. FreeRADIUS.
3. OpenVPN.

### **2.1. OpenLDAP**

1º Comenzamos con la instalación de los paquetes necesarios que se encuentran en los repositorios de Ubuntu. `sudo apt-get install slapd ldap-utils`

```
juanancp@servus:~$ sudo apt-get install slapd ldap-utils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
```

2º Introducimos la contraseña para el usuario administrador de OpenLDAP.



```
Configuración de slapd
Por favor introduzca la contraseña para la entrada de administrador de su
directorio LDAP.

Contraseña del administrador:
```

3º Generamos los ficheros .ldif encargados de crear, modificar, etc. la base de datos que usaremos para almacenar la información de los usuarios. En nuestro caso serán los siguientes. (ver el contenido de los ficheros en anexo-ldap)

```
juanancp@servus:~$ ls
backend.ldap.inf.ldif  config.ldif  frontend.ldap.inf.ldif
juanancp@servus:~$
```

4º Cargamos el primero, **config.ldif**, con la siguiente instrucción: `sudo ldapadd -Y EXTERNAL -H ldapi:/// -f config.ldif`

```
juanancp@servus:~$ sudo ldapadd -Y EXTERNAL -H ldapi:/// -f config.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=config"
```

5º Cargamos el segundo, **backend.ldap.inf.ldif**, con la siguiente instrucción: `sudo ldapadd -x -D cn=admin,cn=config -W -H ldapi:/// -f backend.ldap.inf`

```
juanancp@servus:~$ sudo ldapadd -x -D cn=admin,cn=config -W -H ldapi:/// -f backend.ldap.i
nf.ldif
Enter LDAP Password:
adding new entry "cn=module,cn=config"

adding new entry "olcDatabase=hdb,cn=config"
juanancp@servus:~$
```

6º Cargamos el tercero y último, **frontend.Idap.inf.Idif**, con la siguiente instrucción: `sudo ldapadd -x -D cn=admin,dc=ldap,dc=inf -W -H ldapi:/// -f frontend.ldap.inf.Idif`

```
juananc@servus:~$ sudo ldapadd -x -D cn=admin,dc=ldap,dc=inf -W -H ldapi:/// -f frontend.ldap.inf.Idif
Enter LDAP Password:
adding new entry "dc=ldap,dc=inf"
adding new entry "cn=admin,dc=ldap,dc=inf"
```

7º Instalamos los paquetes necesarios para que los usuarios existentes en la base de datos de OpenLDAP puedan autenticarse en el sistema. `sudo apt-get install libnss-ldap libpam-ldap`

```
juananc@servus:~$ sudo apt-get install libnss-ldap libpam-ldap
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  auth-client-config ldap-auth-client ldap-auth-config
Paquetes sugeridos:
```

La instalación de estos paquetes nos mostrará un asistente en el que debemos de introducir ciertos parámetros.

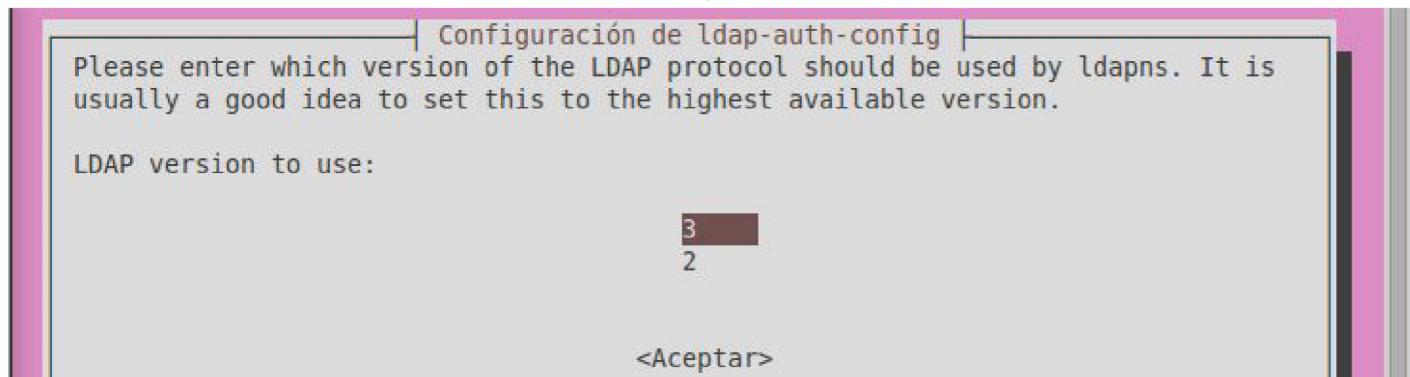
En la primera pantalla, deberemos especificar dónde se encuentra instalado el servidor LDAP. Lo dejamos por defecto en **ldapi://** ya que esto quiere decir que se encuentra instalado en “localhost”.

Configuración de `ldap-auth-config`  
Please enter the URI of the LDAP server to use. This is a string in the form of `ldap://<hostname or IP>:<port>/`. `ldaps://` or `ldapi://` can also be used. The port number is optional.  
Note: It is usually a good idea to use an IP address because it reduces risks of failure in the event name service problems.  
LDAP server Uniform Resource Identifier:  
`ldapi://`

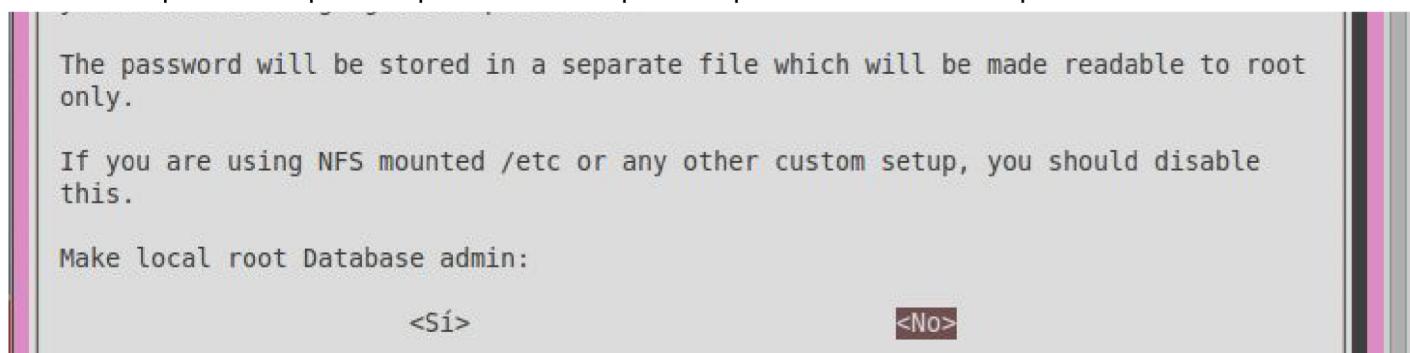
La segunda pantalla, nos pide que introduzcamos el base DN de nuestra base de datos que en nuestro caso es **dc=ldap,dc=inf**.

Configuración de `ldap-auth-config`  
Please enter the distinguished name of the LDAP search base. Many sites use the components of their domain names for this purpose. For example, the domain "example.net" would use "dc=example,dc=net" as the distinguished name of the search base.  
Distinguished name of the search base:  
`dc=ldap,dc=inf`

En la tercera pantalla, tendremos que elegir la versión del protocolo LDAP que queremos usar. Salvo que dispongamos en nuestra red de clientes muy antiguos, lo normal será elegir el valor más alto, ya que es recomendable usar la versión más actual. Así que elegimos la 3.



En la cuarta pantalla, indicaremos si las utilidades que utilicen PAM deberán comportarse del mismo modo que cuando cambiamos contraseñas locales. Esto hará que las contraseñas se guarden en un archivo independiente que sólo podrá ser leído por el superusuario. Decimos que **No**.



En la quinta y última pantalla, el sistema nos pregunta si queremos que sea necesario identificarse para realizar consultas en la base de datos de LDAP. Elegimos **No**.



8º A continuación configuraremos la autenticación de los usuarios en el sistema. Para ello utilizaremos **auth-client-config**, un script que nos ayuda a modificar los archivos de configuración de PAM y NSS. Para conseguirlo, ejecutamos el siguiente comando en la terminal: `sudo auth-client-config -t nss -p lac_ldap`

```
juanancp@servus:~$ sudo auth-client-config -t nss -p lac_ldap  
juanancp@servus:~$
```

Como puede verse, en nuestro caso hemos utilizado dos atributos:

- **-t nss**, con el que le indicamos que los ficheros que vamos a modificar son los correspondientes a NSS
- **-p lac\_ldap**, con el que indicamos que los datos para la configuración debe tomarlos del fichero lac\_ldap. Este fichero se habrá generado durante la instalación.

## FASE DE PRUEBAS

Comprobamos una autenticación con el usuario **profesor1** y visualizamos su **id** para comprobar que todo sea correcto. Como vemos pertenece al grupo que más nos interesa para nuestro proyecto y que solo los usuarios que pertenezcan a él tendrán privilegios para realizar una conexión VPN, el grupo **grupoVPN** con gid 50003.

```
juanapc@servus:~$ su profesor1
Contraseña:
profesor1@servus:/home/juanapc$ id
uid=6000(profesor1) gid=50001(grupoProfesores) grupos=50001(grupoProfesores),50002(grupoUsuarios),50003(grupoVPN)
```

Realizamos la misma comprobación que antes pero con el usuario **alumno1**. En este caso podremos ver que no pertenece al grupo **grupoVPN** por lo que no podrá realizar una conexión VPN.

```
Terminal - juanapc@servus:~
Archivo Editar Ver Terminal Pestañas Ayuda
juanapc@servus:~$ su alumno1
Contraseña:
alumno1@servus:/home/juanapc$ id
uid=5000(alumno1) gid=50000(grupoAlumnos) grupos=50000(grupoAlumnos),50002(grupoUsuarios)
alumno1@servus:/home/juanapc$
```

Con esto completamos la instalación necesaria, de momento, de OpenLDAP para nuestro proyecto.

## 2.2. FreeRADIUS

1º Comenzamos instalando el paquete incluido en los repositorios de Ubuntu, el cual nos va a proporcionar la infraestructura para armar nuestro servidor RADIUS: `sudo apt-get install freeradius`

```
juanapc@servus:~$ sudo apt-get install freeradius
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

2º A continuación instalamos el paquete que nos va a proporcionar el módulo con el que nuestro servidor RADIUS será capaz de comunicarse con nuestro servidor LDAP: `sudo apt-get install freeradius-ldap`

```
juanapc@servus:~$ sudo apt-get install freeradius-ldap
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

3º Ahora editaremos el fichero de configuración para el módulo de LDAP. Éste se encuentra en **/etc/freeradius/modules/ldap**. Lo que haremos básicamente es indicarle los parámetros necesarios para que FreeRADIUS sepa dónde se encuentra el servidor LDAP y pueda comunicarse con él. (ver el contenido final del fichero completo en anexo-radius)

```
GNU nano 2.2.6          Archivo: /etc/freeradius/modules/ldap          Modificado

ldap {
    #
    # Note that this needs to match the name in the LDAP
    # server certificate, if you're using ldaps.
    server = "servus"
    identity = "cn=admin,dc=ldap,dc=inf"
    password = 0688
    basedn = "cn=grupoVPN,ou=Grupos,dc=ldap,dc=inf"
    filter = "(uid=%{Stripped-User-Name}:-%{User-Name})"
```

4º Editaremos el fichero **/etc/freeradius/radiusd.conf**. Lo que haremos en este paso es activar los mensajes de autenticación para el log y desactivar la opción de configuración del proxy. (ver el contenido final del fichero completo en anexo-radius)

```
auth = yes
auth_badpass = yes
auth_goodpass = yes

proxy_requests = no
#$INCLUDE proxy.conf
```

5º Ahora editamos el fichero **/etc/freeradius/clients.conf**. Este fichero contendrá los parámetros necesarios para que los clientes puedan comunicarse con el servidor RADIUS. Aquí es donde se establece el secreto compartido necesario para las conexiones. (ver el contenido final del fichero completo en anexo-radius)

```
GNU nano 2.2.6          Archivo: /etc/freeradius/clients.conf          Modificado

client ldap {
    ipaddr = 192.168.2.220
    secret = Periheli0/
    nastype = other
}

client OpenVPN {
    ipaddr = 192.168.2.210
    secret = Periheli0/
    nastype = other
}
```

6º Como pasa con Apache, en FreeRADIUS podemos crear “servidores virtuales” para cada caso en concreto que queramos. Vamos a eliminar los “servidores virtuales” que vienen activos por defecto y crear uno nuevo para nuestra conexión con LDAP. Realizamos las siguientes operaciones:

Primero eliminamos el contenido del directorio **/etc/freeradius/sites-enabled**

```
root@servus:~# ls -l /etc/freeradius/sites-enabled/
total 0
lrwxrwxrwx 1 root freerad 26 jun 13 12:50 default -> ../sites-available/default
lrwxrwxrwx 1 root freerad 31 jun 13 12:50 inner-tunnel -> ../sites-available/inner-tunnel
root@servus:~# rm /etc/freeradius/sites-enabled/*
```

7º Luego hacemos una copia del fichero **/etc/freeradius/sites-available/default** y la nombramos como **ldap**. Este será el fichero para el “servidor virtual” encargado de nuestro servidor LDAP. Así que una vez lo hayamos generado, creamos un enlace simbólico hacia él en el directorio **/etc/freeradius/sites-enabled**.

```

root@servus:~# cp /etc/freeradius/sites-available/default /etc/freeradius/sites-available/
ldap
root@servus:~# ln -s /etc/freeradius/sites-available/ldap /etc/freeradius/sites-enabled/
root@servus:~# ls -l /etc/freeradius/sites-enabled/
total 0
lrwxrwxrwx 1 root freerad 36 jun 13 12:58 ldap -> /etc/freeradius/sites-available/ldap

```

8º Por último editaremos el fichero **/etc/freeradius/sites-available/ldap** para que el servidor RADIUS sólo establezca conexiones a través del protocolo ldap. Esto lo conseguimos eliminando del fichero todas las referencias a otros mecanismos de autenticación, como por ejemplo eap, pap, chap, etc. y dejando solo **ldap**. (ver el contenido final del fichero completo en anexo-radius)

```

GNU nano 2.2.6          Archivo: /etc/freeradius/sites-available/ldap          Modificado
authorize {
    preprocess
    ldap
    expiration
    logintime
}

authenticate {
    Auth-Type LDAP {
        ldap

```

## FASE DE PRUEBAS

Para verificar que todo ha ido bien vamos a realizar una prueba de conexión. Podemos arrancar nuestro servidor RADIUS en modo debug para ver que es lo que ocurre en el momento de la conexión y controlar que todo funcione correctamente. Para ello ejecutamos el siguiente comando: **sudo freeradius -X**

Desde otra ventana del terminal podemos realizar el test de conexión con el siguiente comando:  
**radtest profesor1 profesor1 servus 1812 Periheli0/**

```
++[ldap] returns ok
++[expiration] returns noop
++[logintime] returns noop
Found Auth-Type = LDAP
# Executing group from file /etc/freeradius/sites-enabled/ldap
+- entering group LDAP {...}
[ldap] login attempt by "profesor1" with password "profesor1"
[ldap] user DN: uid=profesor1,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
[ldap] (re)connect to servus:389, authentication 1
[ldap] bind as uid=profesor1,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf/profesor1 to servus:389
[ldap] waiting for bind result ...
[ldap] Bind was successful
[ldap] user profesor1 authenticated successfully
++[ldap] returns ok
# Executing section post-auth from file /etc/freeradius/sites-enabled/ldap
+- entering group post-auth {...}
++[ldap] returns noop
Sending Access-Accept of id 58 to 127.0.0.1 port 36571
Finished request 0.
Going to the next request
Waking up in 4.9 seconds.
Cleaning up request 0 ID 58 with timestamp +40
Ready to process requests.
```

```
Terminal - juananpc@servus: ~
Archivo Editar Ver Terminal Pestañas Ayuda
juanancp@servus:~$ radtest profesor1 profesor1 servus 1812 Periheli0/
Sending Access-Request of id 58 to 127.0.1.1 port 1812
    User-Name = "profesor1"
    User-Password = "profesor1"
    NAS-IP-Address = 127.0.1.1
    NAS-Port = 1812
    Message-Authenticator = 0x00000000000000000000000000000000
rad recv: Access-Accept packet from host 127.0.1.1 port 1812, id=58, length=20
```

Como vemos, nos devuelve un **Access-Accept**, que quiere decir que profesor1 es válido para autenticarse en el sistema, recordemos que esto se debe entre otras cosas a que pertenece al grupo de LDAP **grupoVPN**.

Ahora probamos otro test de conexión, igual que antes, pero con el usuario **alumno1**.

```
[ldap] performing search in ou=Profesores,ou=Usuarios,dc=ldap,dc=inf, with filter (uid=alumno1)
[ldap] object not found
[ldap] search failed
[ldap] ldap_release_conn: Release Id: 0
++[ldap] returns notfound
++[expiration] returns noop
++[logintime] returns noop
ERROR: No authenticate method (Auth-Type) found for the request: Rejecting the user
Failed to authenticate the user.
Using Post-Auth-Type Reject
# Executing group from file /etc/freeradius/sites-enabled/ldap
+- entering group REJECT {...}
[attr filter.access_reject]      expand: %{User-Name} -> alumno1
attr filter: Matched entry DEFAULT at line 11
++[attr filter.access_reject] returns updated
Delaying reject of request 1 for 1 seconds
Going to the next request
Waking up in 0.9 seconds.
Sending delayed reject for request 1
Sending Access-Reject of id 60 to 127.0.0.1 port 36882
Waking up in 4.9 seconds.
Cleaning up request 1 ID 60 with timestamp +134
Ready to process requests.
[]
```

```
Terminal - juananpc@servus: ~
Archivo Editar Ver Terminal Pestañas Ayuda
juanancp@servus:~$ radtest alumno1 alumno1 servus 1812 Periheli0/
Sending Access-Request of id 60 to 127.0.1.1 port 1812
    User-Name = "alumno1"
    User-Password = "alumno1"
    NAS-IP-Address = 127.0.1.1
    NAS-Port = 1812
    Message-Authenticator = 0x00000000000000000000000000000000
rad recv: Access-Reject packet from host 127.0.1.1 port 1812, id=60, length=20
juanancp@servus:~$ []
```

Como vemos nos devuelve un Access-Reject, que quiere decir que no es un usuario válido para autenticarse en el sistema. Debido a que no pertenece al grupo **grupoVPN**.

Y con esto ya tenemos listo, de momento, nuestro servidor RADIUS, que es capaz de permitir o denegar autenticaciones de usuarios en el sistema, basándose en nuestro servidor LDAP.

### 2.3. OpenLDAP + TLS

Ahora que tenemos nuestro servidor RADIUS y nuestro servidor LDAP funcionando correctamente y comunicándose entre sí es momento de aumentar la seguridad del sistema, ya que la conexión entre ambos ahora mismo se realiza en plano, es decir, los datos viajan sin cifrar y aunque los dos estén en corriendo en la misma máquina siempre es bueno aportar seguridad al sistema.

Para hacer uso de TLS en nuestro servidor LDAP, necesitaremos un certificado público, una llave privada y una autoridad certificadora (CA) la cual va a firmar el certificado que nuestro servidor LDAP usará en sus conexiones.

1º Tanto para crear nuestra CA, como para crear y firmar el par llave/certificado para nuestro servidor LDAP usaremos la herramienta gnutls-bin disponible en los repositorios de ubuntu. Así que lo que haremos en primer lugar será instalarla `sudo apt-get install gnutls-bin`

```
juanapc@servus:~$ sudo apt-get install gnutls-bin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  gnutls-bin
0 actualizados, 1 se instalarán, 0 para eliminar y 0 no actualizados
```

2º Una vez instalado el paquete, procedemos a crear la llave privada que usará nuestra CA. Para generarla lanzamos la siguiente instrucción:

```
sudo sh -c "certtool --generate-privkey > /etc/ssl/private/cakey.pem"
```

```
juanapc@servus:~$ sudo sh -c "certtool --generate-privkey > /etc/ssl/private/cakey.pem"
Generating a 2432 bit RSA private key...
juanapc@servus:~$ 
```

3º Ahora es turno para el certificado público de nuestra CA. Antes que nada, crearemos una plantilla que contendrá los datos de nuestra CA. Generamos el fichero con el editor nano `sudo nano /etc/ssl/ca.info` y le añadimos las líneas que vemos en la imagen.

```
GNU nano 2.2.6                               Archivo: /etc/ssl/ca.info                               Modificado
cn = ASIR
ca
cert_signing_key
```

4º A continuación generamos el certificado CA auto-firmado haciendo uso de la plantilla que hemos creado anteriormente. Para ello lanzamos la siguiente instrucción `sudo certtool --generate-self-signed \ --load-privkey /etc/ssl/private/cakey.pem \ --template /etc/ssl/ca.info \ --outfile /etc/ssl/certs/cacert.pem`

```
juanapc@servus:~$ sudo certtool --generate-self-signed \
> --load-privkey /etc/ssl/private/cakey.pem \
> --template /etc/ssl/ca.info \
> --outfile /etc/ssl/certs/cacert.pem
```

Automáticamente se nos mostrará la información de nuestro certificado CA una vez creado.

```
Generating a self signed certificate...
X.509 Certificate Information:
  Version: 3
  Serial Number (hex): 557de9b4
  Validity:
    Not Before: Sun Jun 14 20:53:08 UTC 2015
    Not After: Mon Jun 13 20:53:08 UTC 2016
  Subject: CN=ASIR
  Subject Public Key Algorithm: RSA
  Certificate Security Level: Normal
    Modulus (bits 2432):
      00:e3:d2:77:e8:99:be:c8:ab:fb:8e:83:4a:24:72:f0
      ab:c9:6a:eb:49:33:4c:12:04:b1:27:fa:90:32:61:39
      2a:e3:a3:90:f4:31:40:a1:26:51:a3:6a:ef:42:90:6b
      5b:1a:5c:b3:cc:d1:99:9e:66:d3:b8:ee:71:e0:fd:03
      85:a7:ff:5e:87:9a:5f:12:6b:c1:72:bf:5e:bd:bd:4f
      df:fa:5a:b9:3c:53:6d:d1:65:50:7f:3e:8b:9e:ec:b3
```

5º El siguiente paso será crear la llave privada para nuestro servidor. Lanzamos la siguiente instrucción:  
sudo certtool --generate-privkey \ --bits 2048 \ --outfile /etc/ssl/private/servus\_slapd\_key.pem

```
juanapc@servus:~$ sudo certtool --generate-privkey \
> --bits 2048 \
> --outfile /etc/ssl/private/servus_slapd_key.pem
** Note: Please use the --sec-param instead of --bits
Generating a 2048 bit RSA private key...
```

6º Ahora es turno de crear y firmar el certificado para el servidor, pero antes crearemos la plantilla que tomará como referencia: sudo nano /etc/ssl/servus.info

GNU nano 2.2.6	Archivo: /etc/ssl/servus.info	Modificado
organization = ASIR cn = servus tls www_server encryption_key signing_key expiration_days = 3650		

7º A continuación generamos y firmamos el certificado para nuestro servidor LDAP. Para ello lanzamos la siguiente instrucción: sudo certtool --generate-certificate \ --load-privkey

```
/etc/ssl/private/servus_slapd_key.pem \ --load-ca-certificate /etc/ssl/certs/cacert.pem \ --load-ca-privkey
/etc/ssl/private/cakey.pem \ --template /etc/ssl/servus.info \ --outfile /etc/ssl/certs/servus_slapd_cert.pem
```

```
juanapc@servus:~$ sudo certtool --generate-certificate \
> --load-privkey /etc/ssl/private/servus_slapd_key.pem \
> --load-ca-certificate /etc/ssl/certs/cacert.pem \
> --load-ca-privkey /etc/ssl/private/cakey.pem \
> --template /etc/ssl/servus.info \
> --outfile /etc/ssl/certs/servus_slapd_cert.pem
```

Automáticamente se nos mostrará la información de nuestro certificado una vez creado.

```
Generating a signed certificate...
X.509 Certificate Information:
  Version: 3
  Serial Number (hex): 557ded02
  Validity:
    Not Before: Sun Jun 14 21:07:14 UTC 2015
    Not After: Wed Jun 11 21:07:14 UTC 2025
  Subject: O=ASIR,CN=servus
  Subject Public Key Algorithm: RSA
  Certificate Security Level: Low
  Modulus (bits 2048):
    00:c6:d8:ef:0e:bc:e5:8c:2c:6d:8e:c8:8c:1e:8f:1a
    34:d7:21:0f:4f:68:72:02:45:84:22:6c:9e:7a:e5:da
    f3:f2:14:a3:e3:ba:d3:60:25:77:79:3b:93:13:e8:96
    f4:04:c3:03:63:93:d0:92:b9:2a:ea:d5:cd:3d:11:49
    ed:71:ef:e7:05:70:41:70:b5:b0:24:d3:2e:d3:4e:ca
    3b:52:41:fa:fb:e4:29:27:1f:82:09:1e:87:5c:17:c1
    ab:2d:4c:b9:da:19:32:4a:fb:45:ef:17:15:e8:99:fe
    4f:46:92:55:b9:0c:c8:0b:fc:97:ad:3b:c2:c3:85:2c
    23:70:bd:52:44:4e:69:cf:68:03:b0:fb:82:ee:ce:bf
    11:13:1f:3a:78:04:e3:48:25:ae:e1:ec:1b:81:73:d4
    9b:4c:bc:58:c7:30:ad:81:4e:f8:84:11:72:e1:4c:0d
```

8º Ahora procederemos a configurar nuestro servidor LDAP para que haga uso del protocolo TLS. Para ello recurrimos a la creación de un fichero llamado certinfo.ldif `sudo nano /etc/ssl/certinfo.ldif` e introduciremos las líneas que vemos en la imagen de a continuación. Estas se encargarán de añadir los nuevos atributos necesarios en el árbol **cn=config** de nuestro servidor LDAP.

```
GNU nano 2.2.6          Archivo: /etc/ssl/certinfo.ldif          Modificado

dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/cacert.pem
-
add: olcTLS CertificateFile
olcTLS CertificateFile: /etc/ssl/certs/servus_slapd_cert.pem
-
add: olcTLS CertificateKeyFile
olcTLS CertificateKeyFile: /etc/ssl/private/servus_slapd_key.pem
```

9º Una vez hayamos creado el fichero certinfo.ldif lo cargaremos en nuestro servidor LDAP con la siguiente instrucción: `sudo ldapmodify -D cn=admin,cn=config -H ldapi:/// -f /etc/ssl/certinfo.ldif -W -v`

```
juanapc@servus:~$ sudo ldapmodify -D cn=admin,cn=config -H ldapi:/// -f /etc/ssl/certinfo.ldif -W -v
ldap_initialize( ldapi:///?base )
Enter LDAP Password:
add olcTLSCACertificateFile:
    /etc/ssl/certs/cacert.pem
add olcTLS CertificateFile:
    /etc/ssl/certs/servus_slapd_cert.pem
add olcTLS CertificateKeyFile:
    /etc/ssl/private/servus_slapd_key.pem
modifying entry "cn=config"
modify complete
```

10º Brindaremos los permisos de acceso adecuados los certificados/llaves para evitar errores en un futuro. Para ello hacemos uso de las siguientes instrucciones:

```
sudo adduser openldap ssl-cert
sudo chgrp ssl-cert /etc/ssl/private/servus_slapd_key.pem
sudo chmod g+r /etc/ssl/private/servus_slapd_key.pem
sudo chmod o-r /etc/ssl/private/servus_slapd_key.pem
```

```
juanapc@servus:~$ sudo adduser openldap ssl-cert
Añadiendo al usuario `openldap' al grupo `ssl-cert' ...
Adding user openldap to group ssl-cert
Hecho.
juanapc@servus:~$ sudo chgrp ssl-cert /etc/ssl/private/servus_slapd_key.pem
juanapc@servus:~$ sudo chmod g+r /etc/ssl/private/servus_slapd_key.pem
juanapc@servus:~$ sudo chmod o-r /etc/ssl/private/servus_slapd_key.pem
juanapc@servus:~$ 
```

11º Por último, editaremos el fichero **/etc/ldap/ldap.conf** donde especificaremos la ruta del certificado CA y añadimos un parámetro, **TLS\_REQCERT demand** para que sea obligatorio el uso de TLS.

```
# TLS certificates (needed for GnuTLS)
TLS_CACERT      /etc/ssl/certs/cacert.pem
TLS_REQCERT     demand
```

**Nota:** LDAP sobre TLS/SSL (ldaps://) está desfasado en favor de **StartTLS**. StartTLS se refiere a una sesión LDAP existente (escuchando en el puerto TCP 389) siendo protegida por TLS/SSL. Mientras que LDAPS, como HTTPS, es un claro cifrado del protocolo de inicio que opera a través del puerto 636.

## FASE DE PRUEBAS

Antes de reiniciar nuestro servidor LDAP, realizamos una búsqueda en el servidor LDAP con la herramienta **ldapsearch** a la que añadimos la opción **-ZZ** que inicia una negociación TLS sobre el puerto 389, forzando una respuesta correcta.

```
juananpc@servus:~$ ldapsearch -D "cn=admin,dc=ldap,dc=inf" -w 0688 -b "dc=ldap,dc=inf" -H ldapi:/// "(objectClass=inetOrgPerson)" uid -ZZ
ldap_start_tls: Protocol error (2)
        additional info: unsupported extended operation
juananpc@servus:~$
```

Como vemos en la imagen nos lanza un error del protocolo StartTLS ya que la opción **-ZZ** de la búsqueda está forzando a usar TLS y este aun no se encuentra disponible porque no hemos reiniciado el servicio **slapd**.

Procedemos a reiniciar el servicio con la instrucción **sudo service slapd restart**

```
juananpc@servus:~$ sudo service slapd restart
 * Stopping OpenLDAP slapd                                         [ OK ]
 * Starting OpenLDAP slapd                                         [ OK ]
juananpc@servus:~$
```

Volvemos a realizar la misma búsqueda.

```
juananpc@servus:~$ ldapsearch -D "cn=admin,dc=ldap,dc=inf" -w 0688 -b "dc=ldap,dc=inf" -H ldapi:/// "(objectClass=inetOrgPerson)" uid -ZZ
# extended LDIF
#
# LDAPv3
# base <dc=ldap,dc=inf> with scope subtree
# filter: (objectClass=inetOrgPerson)
# requesting: uid
#
# alumno1, Alumnos, Usuarios, ldap.inf
dn: uid=alumno1,ou=Alumnos,ou=Usuarios,dc=ldap,dc=inf
uid: alumno1

# alumno2, Alumnos, Usuarios, ldap.inf
dn: uid=alumno2,ou=Alumnos,ou=Usuarios,dc=ldap,dc=inf
uid: alumno2

# profesor1, Profesores, Usuarios, ldap.inf
dn: uid=profesor1,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
uid: profesor1

# profesor2, Profesores, Usuarios, ldap.inf
dn: uid=profesor2,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
uid: profesor2

# search result
search: 3
result: 0 Success

# numResponses: 5
# numEntries: 4
```

Esta vez, como ya se han aplicado los cambios en la configuración del servidor LDAP, la búsqueda se ejecuta correctamente haciendo uso de TLS (-ZZ) y podemos ver cómo nos devuelve los resultados.

## 2.4. FreeRADIUS + OpenLDAP + TLS

Ya tenemos nuestro servidor LDAP capacitado para hacer uso de TLS. Ahora vamos a configurar FreeRADIUS para que la comunicación entre FreeRADIUS y OpenLDAP sea segura gracias a TLS.

Lo único que tendremos que hacer es editar el fichero `/etc/freeradius/modules/ldap` donde le indicaremos la ruta del CA, del certificado del servidor y de la llave privada del servidor. Además estableceremos el parámetro `start_tls` en `yes`.

```
GNU nano 2.2.6                               Archivo: modules/ldap                         Modificado

# server. It contains all of the "tls_*" configuration
# entries used in older versions of FreeRADIUS. Those
# configuration entries can still be used, but we recommend
# using these.
tls {
    # Set this to 'yes' to use TLS encrypted connections
    # to the LDAP database by using the StartTLS extended
    # operation.
    #
    # The StartTLS operation is supposed to be
    # used with normal ldap connections instead of
    # using ldaps (port 689) connections
    start tls = yes
    cacertfile      = /etc/ssl/certs/cacert.pem
    cacertdir       = /etc/ssl/certs/
    certfile        = /etc/ssl/certs/servus_slapd_cert.pem
    keyfile         = /etc/ssl/private/servus_slapd_key.pem
    # randfile       = /path/to/rnd

    # Certificate Verification requirements. Can be:
```

## FASE DE PRUEBAS

Una vez hayamos modificado dicho fichero, podremos realizar una test de conexión con FreeRADIUS para comprobar que realmente se está usando TLS para establecer la comunicación con OpenLDAP.

Para ello abriremos dos terminales, en una lanzaremos FreeRADIUS en modo debug con el comando `sudo freeradius -X` y desde la otra usaremos la herramienta **radtest** para realizar la prueba de conexión: `radtest profesor1 profesor1 servus 1812 Periheli0/`

```
root@servus:/etc/freeradius# radtest profesor1 profesor1 servus 1812 Periheli0/
Sending Access-Request of id 22 to 127.0.1.1 port 1812
    User-Name = "profesor1"
    User-Password = "profesor1"
    NAS-IP-Address = 127.0.1.1
    NAS-Port = 1812
    Message-Authenticator = 0x000000000000000000000000000000000000000000000000000000000000000
rad_recv: Access-Accept packet from host 127.0.1.1 port 1812, id=22, length=20
root@servus:/etc/freeradius# 
```

```
Terminal - juananpc@servus:~ - + ×
Archivo Editar Ver Terminal Pestañas Ayuda
[ldap] user DN: uid=profesor1,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
[ldap] (re)connect to servus:389, authentication 1
[ldap] setting TLS CACert File to /etc/ssl/certs/cacert.pem
[ldap] setting TLS CACert Directory to /etc/ssl/certs/
[ldap] setting TLS Require Cert to demand
[ldap] setting TLS Cert File to /etc/ssl/certs/servus_slapd_cert.pem
[ldap] setting TLS Key File to /etc/ssl/private/servus_slapd_key.pem
[ldap] starting TLS
[ldap] bind as uid=profesor1,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf/profesor1 to servu
s:389
[ldap] waiting for bind result ...
[ldap] Bind was successful
[ldap] user profesor1 authenticated successfully
++[ldap] returns ok
Login OK: [profesor1/profesor1] (from client localhost port 1812)
# Executing section post-auth from file /etc/freeradius/sites-enabled/ldap
+- entering group post-auth {...}
++[ldap] returns noop
Sending Access-Accept of id 22 to 127.0.0.1 port 36065
Finished request 0.
Going to the next request
Waking up in 4.9 seconds.
Cleaning up request 0 ID 22 with timestamp +14
Ready to process requests.

```

Como vemos en la imagen, la autenticación es permitida por el usuario profesor1 y además hace uso de TLS para la comunicación con el servidor LDAP.

Para asegurarnos de que todo va bien, vamos a provocar un error. Volvamos a editar el fichero **/etc/freeradius/modules/ldap** y añadiremos la palabra ERROR en el nombre del certificado CA para así hacerlo inválido.

```
# using tuaps (port 669) connections
start tls = yes
cacertfile      = /etc/ssl/certs/cacertERROR.pem
cacertdir       = /etc/ssl/certs/
```

Una vez realizada la operación, volvamos a realizar el mismo test de conexión que antes: **radtest profesor1 profesor1 servus 1812 Periheli0/**

```
root@servus:/etc/freeradius# radtest profesor1 profesor1 servus 1812 Periheli0/
Sending Access-Request of id 228 to 127.0.1.1 port 1812
  User-Name = "profesor1"
  User-Password = "profesor1"
  NAS-IP-Address = 127.0.1.1
  NAS-Port = 1812
  Message-Authenticator = 0x00000000000000000000000000000000
rad_recv: Access-Reject packet from host 127.0.1.1 port 1812, id=228, length=20
root@servus:/etc/freeradius# 
```

```
Terminal - juananpc@servus:~ - + ×
Archivo Editar Ver Terminal Pestañas Ayuda
[ldap] ldap_get_conn: Got Id: 0
[ldap] attempting LDAP reconnection
[ldap] (re)connect to servus:389, authentication 0
[ldap] setting TLS CACert File to /etc/ssl/certs/cacertERROR.pem
[ldap] setting TLS CACert Directory to /etc/ssl/certs/
[ldap] setting TLS Require Cert to demand
[ldap] setting TLS Cert File to /etc/ssl/certs/servus_slapd_cert.pem
[ldap] setting TLS Key File to /etc/ssl/private/servus_slapd_key.pem
[ldap] starting TLS
[ldap] ldap_start_tls_s()
[ldap] could not start TLS Connect error
[ldap] (re)connection attempt failed
[ldap] search failed
[ldap] ldap_release_conn: Release Id: 0
++[ldap] returns fail
Invalid user: [profesor1/profesor1] (from client localhost port 1812)
Using Post-Auth-Type Reject
# Executing group from file /etc/freeradius/sites-enabled/ldap
+- entering group REJECT {...}
[attr filter.access reject]      expand: %{User-Name} -> profesor1
[attr filter] Matched entry DEFAULT at line 11
```

Efectivamente, vemos en la imagen como se rechaza la autenticación del usuario profesor1 y además muestra un error de conexión TLS al haber cambiado el nombre al certificado CA.

Ahora si, ya tenemos nuestro servidor RADIUS y nuestro servidor LDAP listos para su funcionamiento y haciendo uso de una comunicación segura entre ellos gracias a TLS.

## 2.5. OpenVPN

Procedemos a instalar la última piedra angular de nuestro proyecto, un servidor VPN. El servidor VPN estará alojado en **servidor**, otro equipo diferente de donde se encuentran instalados los servidores LDAP y RADIUS.

Cubriremos el modelo de implementación necesario para nuestro caso: “**Cliente-servidor**”, es decir, un único servidor con múltiples clientes remotos capaz de encaminar el tráfico IP.

1º Lo primero que haremos es instalar los paquetes necesarios incluidos en el repositorio de Ubuntu:  
`sudo apt-get install openvpn openvpn-auth-radius easy-rsa`

- `openvpn`: software encargado de proporcionar al sistema un servidor VPN.
- `openvpn-auth-radius`: plugin para `openvpn` encargado de la comunicación con el servidor RADIUS.
- `easy-rsa`: herramienta basada en scripts para el manejo de claves y certificados.

```
juanapc@servidor:~$ sudo apt-get install openvpn openvpn-auth-radius easy-rsa
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  libccid liblzo2-2 libpkcs11-helper1 opensc pcscd
Paquetes sugeridos:
  pcmciautils systemd
Se instalarán los siguientes paquetes NUEVOS:
  easy-rsa libccid liblzo2-2 libpkcs11-helper1 opensc openvpn
  openvpn-auth-radius pcscd
0 actualizados, 8 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 1.511 kB de archivos.
Se utilizarán 4.941 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] 
```

2º Seguidamente vamos a copiar al directorio de OpenVPN los ficheros que nos harán falta para su posterior configuración y la creación de los certificados. Para ello copiamos el directorio de `easy-rsa` con el comando `cp -r /usr/share/easy-rsa/ /etc/openvpn/` y el fichero de configuración del plugin para RADIUS con el comando `cp /usr/share/doc/openvpn-auth-radius/examples/radiusplugin.cnf /etc/openvpn/`. Una vez realizado ésto, comprobamos que efectivamente los ficheros se encuentren en el directorio de OpenVPN.

```
juanapc@servidor:~$ sudo -s
root@servidor:~# cp -r /usr/share/easy-rsa/ /etc/openvpn/
root@servidor:~# cp /usr/share/doc/openvpn-auth-radius/examples/radiusplugin.cnf /etc/openvpn/
root@servidor:~# cd /etc/openvpn/
root@servidor:/etc/openvpn# ls
easy-rsa  radiusplugin.cnf  update-resolv-conf
root@servidor:/etc/openvpn# 
```

Antes de configurar la VPN, primero tendremos que crear la infraestructura de clave pública (PKI). La PKI se compone de la autoridad de certificación (CA), las claves privadas y los certificados (claves públicas), tanto en el servidor como para los clientes. También tendremos que generar un fichero Diffie-Hellman para garantizar la confidencialidad en el intercambio de claves. Para la creación de la PKI, haremos uso de los scripts `easy-rsa` suministrados por la distribución de OpenVPN en sí.

3º Para ello, lo primero que haremos será editar el fichero **/etc/openvpn/easy-rsa/vars** donde cambiaremos los valores por defecto, sobre la información de los certificados, por los de nuestro gusto.

```
# which will be placed in the certificate.
# Don't leave any of these fields blank.
export KEY_COUNTRY="ES"
export KEY_PROVINCE="CA"
export KEY_CITY="Cadiz"
export KEY_ORG="ASIR"
export KEY_EMAIL="admin@mailasir.inf"
export KEY_OU="Informatica"
```

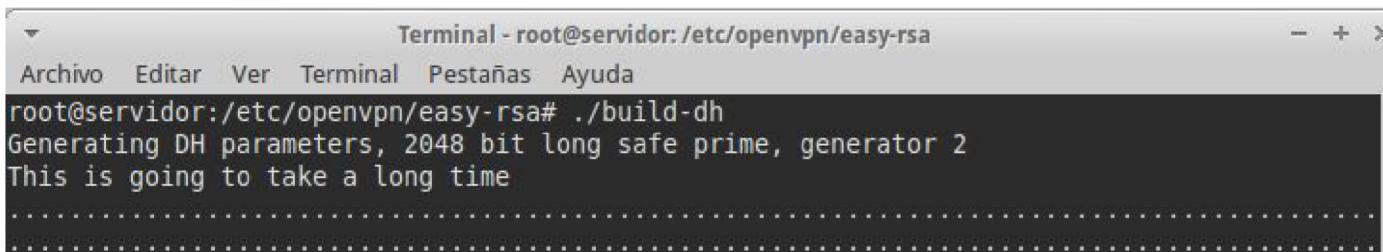
4º Nos posicionamos en el directorio **/etc/openvpn/easy-rsa** y cargamos las variables del fichero que hemos modificado con el comando **source vars** luego ejecutamos el script **./clean-all** para eliminar, si hubiera, certificados existentes. Para finalizar procedemos a crear el certificado de autoridad ejecutando el script **./pktool --initca** Con ésto, ya tendremos nuestro CA listo para ser usado.

```
root@servidor:/etc/openvpn# cd easy-rsa/
root@servidor:/etc/openvpn/easy-rsa# source vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
root@servidor:/etc/openvpn/easy-rsa# ./clean-all
root@servidor:/etc/openvpn/easy-rsa# ./pktool --initca
Using CA Common Name: ASIR CA
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'ca.key'
-----
root@servidor:/etc/openvpn/easy-rsa# 
```

5º Ahora vamos a generar un certificado y su clave privada para el servidor. Para ello, ejecutamos el siguiente script **./pktool --server servidor** De esta forma, se creará automáticamente el certificado y su clave.

```
root@servidor:/etc/openvpn/easy-rsa# ./pktool --server servidor
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'servidor.key'
-----
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'ES'
stateOrProvinceName :PRINTABLE:'CA'
localityName         :PRINTABLE:'Cadiz'
organizationName     :PRINTABLE:'ASIR'
```

6º Por último, para nuestro servidor necesitaremos también una llave Diffie-Hellman para el intercambio de claves. La generamos ejecutando el siguiente script: **./build-dh**



The screenshot shows a terminal window titled "Terminal - root@servidor: /etc/openvpn/easy-rsa". The menu bar includes "Archivo", "Editar", "Ver", "Terminal", "Pestañas", and "Ayuda". The command **./build-dh** is being run, and the output indicates that it is generating DH parameters with a 2048-bit safe prime and generator 2. It also states that the process will take a long time.

```
Terminal - root@servidor: /etc/openvpn/easy-rsa
Archivo Editar Ver Terminal Pestañas Ayuda
root@servidor:/etc/openvpn/easy-rsa# ./build-dh
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....
```

7º Una vez tenemos lo todo lo necesario, copiamos al directorio **/etc/openvpn** el certificado de autoridad; el certificado y la clave del servidor, y la llave Diffie-Hellman que hemos generado previamente. Para ello lanzamos la instrucción `cp ca.crt servidor.crt servidor.key dh2048.pem /etc/openvpn/`

```
root@servidor:/etc/openvpn/easy-rsa# cd keys/
root@servidor:/etc/openvpn/easy-rsa/keys# ls
01.pem  ca.key      index.txt      index.txt.old  serial.old    servidor.csr
ca.crt  dh2048.pem  index.txt.attr  serial        servidor.crt  servidor.key
root@servidor:/etc/openvpn/easy-rsa/keys# cp ca.crt servidor.crt servidor.key dh2048.pem /etc/openvpn/
```

8º Ahora vamos a generar un certificado y su clave privada para el usuario **profesor1**. Antes que nada editaremos el fichero **/etc/easy-rsa/vars** para cambiar el tiempo de expiración de los certificados, ya que para el servidor estaba en 3650 días (10 años) y para los usuarios lo reduciremos a 365 días (1 año).

```
# In how many days should certificates expire?
export KEY_EXPIRE=365
```

9º Una vez hayamos modificado el fichero, lo cargamos con la instrucción `source vars` y después ejecutaremos el script `./pktool profesor1`. De esta forma se generará automáticamente el certificado y la clave privada para el usuario **profesor1**.

```
root@servidor:/etc/openvpn/easy-rsa# source vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
root@servidor:/etc/openvpn/easy-rsa# ./pktool profesor1
Generating a 2048 bit RSA private key
.....+ ++
.....+ ++
writing new private key to 'profesor1.key'

Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'ES'
stateOrProvinceName :PRINTABLE:'CA'
localityName         :PRINTABLE:'Cadiz'
organizationName     :PRINTABLE:'ASIR'
organizationalUnitName:PRINTABLE:'Informatica'
commonName           :PRINTABLE:'profesor1'
name                 :PRINTABLE:'EasyRSA'
```

10º Lo que haremos a continuación será copiar los certificados del usuario junto con el certificado de autoridad del servidor a la máquina cliente, que se conectará más tarde a la VPN, a través de un método seguro como lo es **scp**.

```
root@servidor:/etc/openvpn/easy-rsa# cd keys/
root@servidor:/etc/openvpn/easy-rsa/keys# ls
01.pem  ca.key      index.txt.attr  profesor1.crt  serial        servidor.csr
02.pem  dh2048.pem  index.txt.old   profesor1.csr  serial.old    servidor.key
ca.crt  index.txt    index.txt.old  profesor1.key  servidor.crt
root@servidor:/etc/openvpn/easy-rsa/keys# scp profesor1.crt profesor1.key ca.crt juananpc@192.168.2.1:~
juananpc@192.168.2.1's password:
profesor1.crt                                         100% 5415      5.3KB/s  00:00
profesor1.key                                         100% 1704      1.7KB/s  00:00
ca.crt                                                 100% 1688      1.7KB/s  00:00
```

11º Ahora pasamos a la configuración del servidor VPN. Primero copiamos el fichero **server.conf**, donde se encuentran todas las opciones relacionadas con la configuración del servidor, en el directorio **/etc/openvpn**. Para ello lanzamos la siguiente instrucción:

```
cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf /etc/openvpn/
```

```
root@servidor:/etc/openvpn# cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz .
root@servidor:/etc/openvpn# gzip -d server.conf.gz
root@servidor:/etc/openvpn# ls
ca.crt      easy-rsa          server.conf    servidor.key
dh2048.pem   radiusplugin.cnf  servidor.crt  update-resolv-conf
```

12º Luego pasaremos a editar el fichero que ya se encuentra en **/etc/openvpn/server.conf**. Básicamente lo que haremos será indicar cuales serán el certificado de autoridad, el certificado del servidor, la clave privada del servidor y la llave Diffie-Hellman que el servidor VPN va a tomar, además descomentaremos el parámetro `crl-verify crt.pem` para que tenga en cuenta la lista de los certificados de usuarios revocados. Los demás parámetros los podemos dejar por defecto. (ver fichero completo en anexo-vpn)

```
GNU nano 2.2.6          Archivo: /etc/openvpn/server.conf

port 1194
proto udp
dev tun
server 10.8.0.0 255.255.255.0

push "route 192.168.2.0 255.255.255.0"
topology subnet

ca ca.crt
cert servidor.crt
key servidor.key
dh dh2048.pem
```

13º A continuación habilitaremos el ip forwarding para que el sistema se encargue de la retransmisión de los paquetes. Para ello editamos el fichero **/etc/sysctl.conf** y descomentamos la línea en cuestión.

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Una vez hayamos editado el fichero, debemos confirmar los cambios con el siguiente comando: `sysctl -p /etc/sysctl.conf`

```
root@servidor:/etc/openvpn# sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 1
root@servidor:/etc/openvpn#
```

14º Es hora de editar el fichero de configuración del plugin para openvpn encargado de establecer la comunicación con nuestro servidor RADIUS. Lo que haremos será abrir con el editor el fichero **/etc/openvpn/radiusplugin.cnf** y en el apartado “**server**”, debemos indicar el nombre o la ip de nuestro servidor RADIUS y el secreto compartido que establecimos en la configuración de FreeRADIUS. (ver fichero completo en anexo-vpn)

```

GNU nano 2.2.6          Archivo: /etc/openvpn/radiusplugin.cnf

server
{
    # The UDP port for radius accounting.
    acctport=1813
    # The UDP port for radius authentication.
    authport=1812
    # The name or ip address of the radius server.
    name=192.168.2.220
    # How many times should the plugin send the if there is no response?
    retry=1
    # How long should the plugin wait for a response?
    wait=1
    # The shared secret.
    sharedsecret=Periheli0/
}

```

15º Ahora indicaremos a OpenVPN que debe cargar el plugin para RADIUS con sus parámetros de configuración. Esto se hace editando su fichero de configuración **/etc/openvpn/server.conf** y añadiendo la siguiente línea al final del fichero: **plugin /usr/lib/openvpn/radiusplugin.so /etc/openvpn/radiusplugin.cnf**

```

plugin /usr/lib/openvpn/radiusplugin.so /etc/openvpn/radiusplugin.cnf

^G Ver ayuda ^O Guardar ^R Leer Fich ^Y RePág. ^K Cortar Texto^C Pos actual

```

16º A llegado el momento de iniciar el servicio OpenVPN. **sudo service openvpn start** Veremos al iniciarse una línea referente a RADIUS-PLUGIN, esto nos indica que el plugin se ha cargado correctamente.

```

root@servidor:/etc/openvpn# service openvpn start
 * Starting virtual private network daemon(s)...
 *   Autostarting VPN 'server'
Sat Jun 13 15:36:33 2015 RADIUS-PLUGIN: Configfile name: /etc/openvpn/radiusplugin.cnf.

```

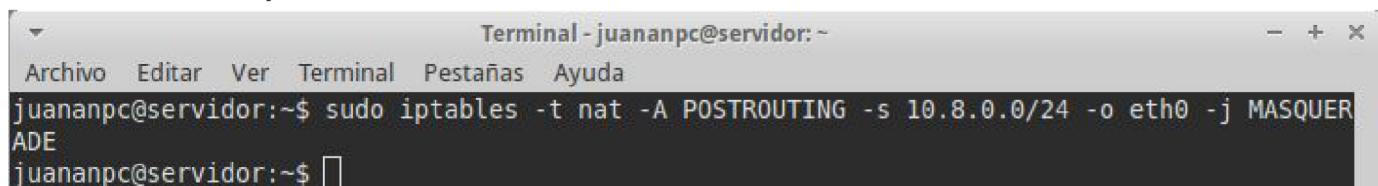
Podemos observar si ejecutamos el comando **ifconfig** como se ha creado una nueva interfaz de red llamada **tun0** con la ip del servidor VPN (10.8.0.1).

```

tun0      Link encap:UNSPEC  direcciónHW 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          Direc. inet:10.8.0.1 P-t-P:10.8.0.2 Másc:255.255.255.255
          ACTIVO PUNTO A PUNTO FUNCIONANDO NOARP MULTICAST MTU:1500 Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:100
          Bytes RX:0 (0.0 B)  TX bytes:0 (0.0 B)

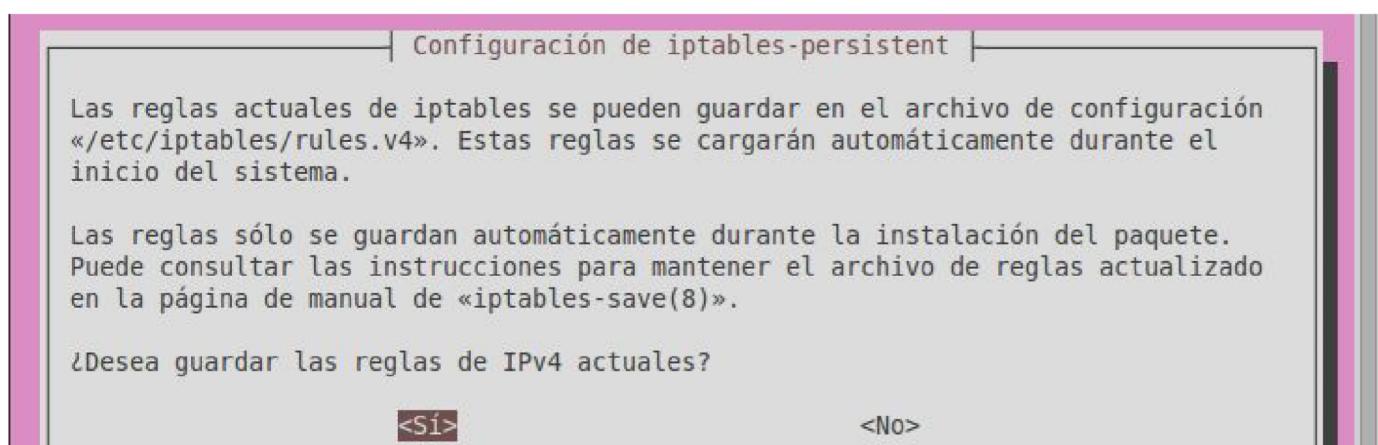
```

17º Por último será necesario añadir una regla IPtable en el servidor para que los clientes que se conecten a la VPN tengan conexión a Internet. Teniendo en cuenta que la red sobre la que trabajará la VPN es la 10.8.0.0/24 añadiremos la siguiente regla: `sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE`



```
Terminal - juananpc@servidor: ~
Archivo Editar Ver Terminal Pestañas Ayuda
juananpc@servidor:~$ sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE
juananpc@servidor:~$
```

18º Una vez añadida la regla, necesitamos hacerla permanente para que no se elimine en caso de reinicio del servidor. Para ello, después de haber introducido la regla, vamos a instalar un paquete del repositorio de ubuntu que se encargará de hacerla persistente automáticamente: `sudo apt-get install iptables-persistent` Seguidamente nos mostrará un asistente preguntandonos si queremos guardar las reglas IPtables actuales, le decimos que Sí y listo, ya tenemos nuestra regla de manera permanente en el servidor.



Con esto ya tenemos nuestro servidor VPN configurado como “cliente-servidor”, funcionando y comunicándose con nuestro servidor RADIUS, que a su vez se comunica con nuestro servidor LDAP, para realizar las autenticaciones de los usuarios que se conecten a la VPN.

Solo faltaría configurar el equipo del cliente que se conecta a la VPN. Detallaremos estos pasos en el punto **4. Manual de usuario**.

## FASE DE PRUEBAS

Una vez configurado el equipo cliente también, procederemos a realizar algunas pruebas para comprobar que todo funcione correctamente.

La primera prueba que haremos será un intento de conexión a la VPN usando las credenciales correctas para el usuario **profesor1**. Estas credenciales serán por una parte usuario/contraseña de su cuenta en LDAP y por otra parte el CA y su certificado/clave privada.

**Autenticación**

Tipo:	Contraseña con certificados (TLS)
Nombre de usuario:	profesor1
Contraseña:	.....
Certificado de usuario:	profesor1.crt
Certificado CA:	ca.crt
Clave privada:	profesor1.key
Contraseña de la clave privada:	

Una vez guardados los ajustes de la conexión, nos conectaremos a ASIR VPN a través del administrador de red del sistema.



Como vemos en la imagen de arriba, se ha establecido la conexión con éxito a la VPN. Vamos a comprobar que sea cierto. Abrimos un terminal en el equipo cliente y lanzamos el comando ifconfig

```
tun0      Link encap:UNSPEC  direcciónHW 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          Direc. inet:10.8.0.6  P-t-P:10.8.0.5  Másc:255.255.255.255
          ACTIVO PUNTO A PUNTO FUNCIONANDO NOARP MULTICAST MTU:1500  Métrica:1
          Paquetes RX:60 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:121 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:100
          Bytes RX:13919 (13.9 KB)  TX bytes:24539 (24.5 KB)
```

Como podemos observar, se ha creado una nueva interfaz de red **tun0** a la que se le ha asignado la IP 10.8.0.6 así que vamos a realizar ping a la IP del servidor VPN y a google para comprobar que la conexión sea absoluta.

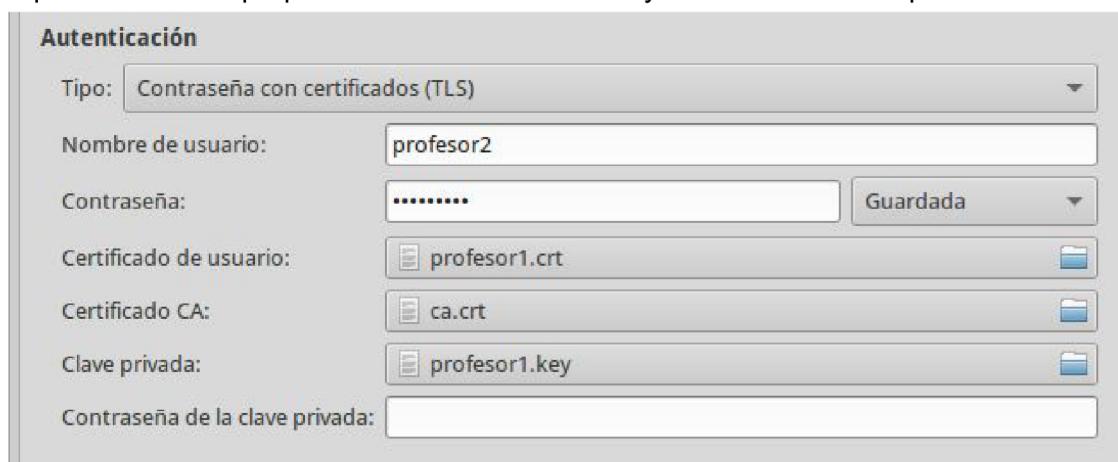
```
juanapc@desktop:~$ ping 10.8.0.1 -c 2
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=0.290 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=0.300 ms

--- 10.8.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.290/0.295/0.300/0.005 ms
juanapc@desktop:~$ ping google.es -c 2
PING google.es (216.58.211.195) 56(84) bytes of data.
64 bytes from mad01s25-in-f195.1e100.net (216.58.211.195): icmp_seq=1 ttl=55 time=21.6 ms
64 bytes from mad01s25-in-f195.1e100.net (216.58.211.195): icmp_seq=2 ttl=54 time=23.2 ms

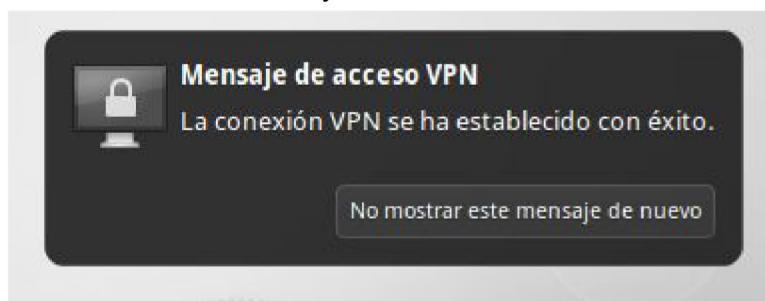
--- google.es ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 21.655/22.435/23.215/0.780 ms
```

Efectivamente, todo funciona correctamente. Ya estamos conectados a la VPN de ASIR!

Ahora vamos a comprometer un poco la seguridad. ¿Qué pasaría si el usuario **profesor2** se hiciera con el certificado y la clave privada de **profesor1**? Supuestamente para establecer la conexión en la VPN le hará falta también saber las credenciales (usuario/contraseña) de profesor1 en LDAP, pero ¿Qué pasaría si prueba con sus propias credenciales de LDAP y los certificados de profesor1? Veamos.



Realizamos la conexión y ...



Podemos ver que el usuario profesor2 se ha podido conectar usando el certificado y la clave privada de profesor1. Esto obviamente no debería de ser así, vamos a solucionarlo.

## 2.6. Estableciendo el CN del certificado del usuario como nombre de usuario para el Login.

Usar el plugin de RADIUS para OpenVPN nos ofrece la posibilidad de la autenticación por dos factores: usuario/contraseña y certificados de usuario. Pero existe un pequeño problema. Como acabamos de ver, en nuestro caso, el certificado y la clave pertenecen a profesor1 mientras que las credenciales usuario/contraseña son de profesor2 y la autenticación ha sido válida.

Para arreglar esto, haremos uso de un script que deberemos cargar con OpenVPN. Lo que hará este script será usar el CN del certificado del usuario como nombre de usuario para el Login, por lo que el CN del certificado y el nombre usuario de LDAP deben ser el mismo. Esto forzará a que se deba introducir la contraseña correcta de LDAP para el usuario del certificado.

1º Como OpenVPN se encuentra instalado en otro equipo distinto de FreeRADIUS y el script hará uso de la herramienta radclient de FreeRADIUS, lo primero queharemos será instalar el paquete **freeradius-utils** que se encuentra en los repositorios de ubuntu: `sudo apt-get install freeradius-utils`

A screenshot of a terminal window titled 'Terminal - juananpc@servidor:~'. The window menu includes 'Archivo', 'Editar', 'Ver', 'Terminal', 'Pestañas', and 'Ayuda'. The command `sudo apt-get install freeradius-utils` is being typed into the terminal. The output shows: 'Leyendo lista de paquetes... Hecho', 'Creando árbol de dependencias', and 'Leyendo la información de estado... Hecho'.

2º Crearemos un fichero de script .sh en **/etc/openvpn/** para añadir el contenido del script: sudo nano /etc/openvpn/cn\_radius\_auth.sh y añadimos las líneas del script como vemos en la imagen. (ver código en anexo-radius)

```
GNU nano 2.2.6                               Archivo: /etc/openvpn/cn_radius_auth.sh                               Modificado

#!/bin/bash

# $1 provides the temp file name provided by OpenVPN
# file has two lines: username and password, as entered by the user.
# We get the username from the user cert's CN (available via an envvar).

export PATH="/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin"

if [ ! -z ${common_name} ]; then
    username=`echo ${common_name}`
else
    username=..
fi

if [ ! -z $1 ] && [ -f $1 ]; then
    password=`tail -1 $1`
else
    password=..
fi

radius_server=servus
# shared secret for localhost (or your RADIUS server) from /etc/freeradius/
shared_secret="Perihelio/"

AUTHCHECK=`cat << EOF | /usr/bin/radclient -s ${radius_server} auth ${shared_secret} | grep approved | tr -d '\n' | tail -c 1
User-Name=${username}
User-Password=${password}
EOF`

if [[ $AUTHCHECK = 1 ]]; then
    exit 0
else
    exit 1
fi
```

3º Ahora editamos el fichero **/etc/openvpn/server.conf** y añadiremos las líneas encargadas de cargar el script que acabamos de crear.

```
# script cn verify
tmp-dir /dev/shm
auth-user-pass-verify /etc/openvpn/cn_radius_auth.sh via-file
```

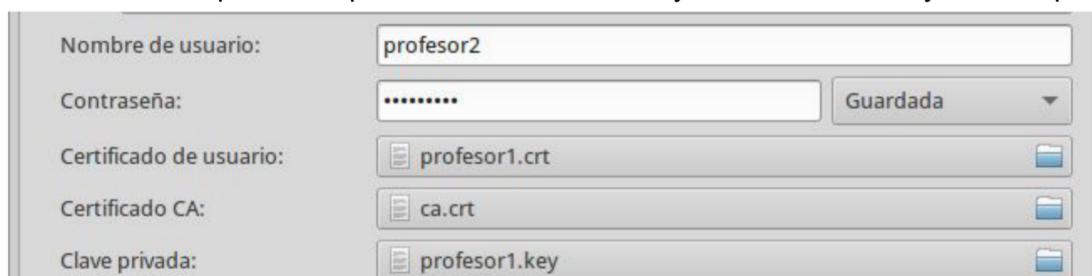
4º Para aplicar los cambios reiniciamos el servicio openvpn y listo: sudo service openvpn restart

```
juananc@servidor:~$ sudo service openvpn restart
 * Stopping virtual private network daemon(s)...
 *   Stopping VPN 'server'
Mon Jun 15 11:06:41 2015 RADIUS-PLUGIN: BACKGROUND AUTH: EXIT
Mon Jun 15 11:06:41 2015 RADIUS-PLUGIN: BACKGROUND ACCT: EXIT
```

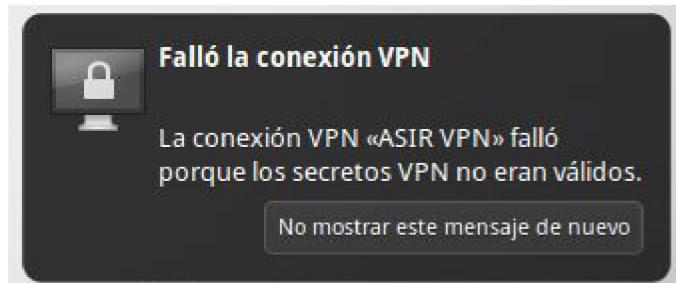
[ OK ]

## FASE DE PRUEBAS

Vamos ahora a realizar la misma prueba de antes. Nos intentaremos conectar a la VPN con las credenciales de profesor2 para usuario/contraseña y con el certificado y la clave privada de profesor1.



Realizamos la conexión y ...



Efectivamente pasa lo que debería de pasar, la conexión no se estableció. Ahora las credenciales usuario/contraseña de LDAP que se usarán para la autenticación deben ser las del mismo usuario para el que se creó el certificado y la clave privada.

Ahora si, ya tenemos el sistema completo funcionando como debería ser OpenVPN + FreeRADIUS + OpenLDAP estableciendo conexiones y validando autenticaciones de forma segura.

## 2.7. Lista de paquetes instalados

<b>Servus</b>	<b>Servidor</b>	<b>Cliente</b>
<b>OpenLDAP:</b> slapd ldap-utils libnss-ldap libpam-ldap gnutls-bin  <b>FreeRADIUS:</b> freeradius freeradius-ldap	<b>OpenVPN:</b> openvpn openvpn-auth-radius easy-rsa freeradius-utils	<b>Administrador de red GUI:</b> network-manager-openvpn

## 3. Manual de administración

El administrador del sistema, una vez que ya se ha realizado todo el proceso de instalación y se ha configurado todo correctamente como se ha descrito en el manual de instalación, solo debería de preocuparse de tres cosas:

- Crear un fichero con los ajustes de la conexión a la VPN que los usuarios deberán cargar en sus equipos clientes.
- Añadir o eliminar a los usuarios del grupo de LDAP “**grupoVPN**”.
- Creación de certificados y claves para los usuarios.

### 3.1. Crear un fichero con los ajustes de la conexión a la VPN que los usuarios deberán cargar

El administrador deberá crear un fichero de configuración para los clientes, el cual contendrá los parámetros necesarios para que pueda establecer la conexión con el servidor VPN. Este fichero lo podrá crear con cualquier editor, como por ejemplo nano: `nano ASIR-VPN.conf` y añadirá, en nuestro caso, los siguientes parámetros.

```
GNU nano 2.2.6           Archivo: ASIR-VPN.conf          Modificado

client
remote 192.168.2.210 1194
auth-user-pass
comp-lzo yes
dev tun
proto udp
nobind
auth-nocache
script-security 2
persist-key
persist-tun
user nobody
group nogroup
```

Una vez tengamos el fichero creado solo quedará suministrarselo a los usuarios.

### 3.2. Añadir o eliminar a los usuarios del grupo de LDAP “grupoVPN”

Como ya sabemos, hemos creado un grupo en nuestra base de datos de LDAP que se llama **grupoVPN**. Tal y como hemos configurado FreeRADIUS, será obligatorio pertenecer a este grupo para poder establecer una conexión con la VPN, si un usuario no pertenece no podrá conectarse.

Será el administrador el encargado de añadir o eliminar a los usuarios de LDAP al grupo grupoVPN. Esto se realizará desde **servus**, que es donde está instalado OpenLDAP.

**Para añadir un usuario a dicho grupo deberá realizar lo siguiente:**

1º Creará un fichero .ldif que usará como plantilla para añadir a los usuarios al grupo: **nano addvpnuser.ldif** y añadimos las líneas que vemos en la imagen.

```
GNU nano 2.2.6           Archivo: addvpnuser.ldif

dn: cn=grupoVPN,ou=Grupos,dc=ldap,dc=inf
changetype: modify
add: memberUid
memberUid: profesor2
```

Solo deberá ir cambiando o añadiendo (se pueden varios a la vez, uno detrás de otro) más **memberUid: nombreusuario** según las necesidades.

2º Ejecutará el fichero **addvpnuser.ldif** con la herramienta **ldapmodify** de OpenLDAP con la siguiente instrucción: **ldapmodify -D cn=admin,dc=ldap,dc=inf -W -f addvpnuser.ldif -v**

```
juanapc@servus:~$ ldapmodify -D cn=admin,dc=ldap,dc=inf -W -f addvpnuser.ldif -v
ldap_initialize( <DEFAULT> )
Enter LDAP Password:
add memberUid:
    profesor2
modifying entry "cn=grupoVPN,ou=Grupos,dc=ldap,dc=inf"
modify complete
```

Ejecutando esa instrucción se modificará la base de datos de LDAP añadiendo tantos usuarios, al grupo **grupoVPN**, como se especifiquen en el fichero **addvpnuser.ldif**.

**Para eliminar un usuario de dicho grupo deberá realizar lo siguiente:**

1º Creará un fichero .ldif que usará como plantilla para eliminar a los usuarios del grupo: `nano removevpnuser.ldif`

```
GNU nano 2.2.6          Archivo: removevpnuser.ldif

dn: cn=grupoVPN,ou=Grupos,dc=ldap,dc=inf
changetype: modify
delete: memberUid
memberUid: profesor2
```

Solo deberá ir cambiando o añadiendo (se pueden varios a la vez, uno detrás de otro) más **memberUid: nombreusuario** según las necesidades.

2º Ejecutará el fichero **removevpnuser.ldif** con la herramienta **ldapmodify** de OpenLDAP con la siguiente instrucción: `ldapmodify -D cn=admin,dc=ldap,dc=inf -W -f removevpnuser.ldif -v`

```
juanapc@servus:~$ ldapmodify -D cn=admin,dc=ldap,dc=inf -W -f removevpnuser.ldif -v
ldap_initialize( <DEFAULT> )
Enter LDAP Password:
delete memberUid:
    profesor2
modifying entry "cn=grupoVPN,ou=Grupos,dc=ldap,dc=inf"
modify complete
```

Ejecutando esa instrucción se modificará la base de datos de LDAP eliminando tantos usuarios, del grupo **grupoVPN**, como se especifiquen en el fichero **addvpnuser.ldif**.

### 3.3. Creación y revocación de certificados y claves para los usuarios

El administrador deberá crear o revocar también los certificados y las claves privadas para los usuarios, las cuales deberá suministrárselas junto con el certificado CA del servidor, para que OpenVPN pueda validarlas. Esto lo hará desde **servidor**, que es donde se encuentra instalado OpenVPN.

#### Creación de un certificado y su clave privada

De la misma forma que creamos el certificado y la clave privada para **profesor1** en el manual de instalación, el administrador deberá crear los certificados y las claves para los usuarios.

Para ello, deberá usar para mayor comodidad los script de easy-rsa que se encuentran instalados junto con OpenVPN.

1º Deberá loguearse como root y posicionarse en el directorio donde se encuentran dichos scripts: `/etc/openvpn/easy-rsa/`

```
juanapc@servidor:~$ sudo -s
root@servidor:~# cd /etc/openvpn/easy-rsa/
```

2º Cargará las variables del fichero `/etc/openvpn/easy-rsa/vars` con el siguiente comando: `source vars`

```
root@servidor:/etc/openvpn/easy-rsa# source vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
root@servidor:/etc/openvpn/easy-rsa# 
```

3º Deberá ejecutar el script “**pktool**” y especificarle el nombre para el usuario que se le va a crear el certificado y la clave. Este nombre será el CN del certificado así que deberá coincidir con su nombre de usuario en LDAP. Veamos un ejemplo para el usuario profesor2.

```
root@servidor:/etc/openvpn/easy-rsa# ./pktool profesor2
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'profesor2.key'
-----
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'ES'
stateOrProvinceName :PRINTABLE:'CA'
localityName         :PRINTABLE:'Cádiz'
organizationName     :PRINTABLE:'ASTB'
```

4º Una vez ejecutado el script, el certificado y la clave privada se encontrarán en el directorio `/etc/openvpn/easy-rsa/keys/`

```
root@servidor:/etc/openvpn/easy-rsa# ls -l keys/profesor2.*
-rw-r--r-- 1 root root 5415 jun 16 19:56 keys/profesor2.crt
-rw-r--r-- 1 root root 1070 jun 16 19:56 keys/profesor2.csr
-rw----- 1 root root 1704 jun 16 19:56 keys/profesor2.key
root@servidor:/etc/openvpn/easy-rsa# 
```

Una vez generados, habrá que proporcionárselos al usuario en cuestión junto con el certificado CA del servidor y el fichero de configuración de la conexión para el cliente.

### Revocación de un certificado

Los certificados x509 emitidos para los clientes OpenVPN tienen una fecha de inicio de validez y una fecha de expiración, durante este tiempo los clientes podrán autenticarse con el servidor OpenVPN usando el certificado. El servidor OpenVPN autorizará el acceso a los clientes siempre y cuando la fecha de expiración del certificado del cliente no haya expirado, sin embargo, habrá ocasiones en las que se desee invalidar o revocar un certificado para un cliente OpenVPN de manera que ya no pueda ser usado para propósitos de autenticación.

Entre las principales razones para revocar el acceso a la VPN podemos encontrar:

- Ya no se desea dar acceso a la VPN a un usuario en específico.
- La clave privada asociada con el certificado ha sido comprometida o robada.
- El usuario no recuerda la contraseña con la que se cifró la clave privada asociada a un certificado (éste no podrá ser nuestro caso ya que no usamos contraseñas para las claves privadas).

Para revocar el certificado de un cliente siga el siguiente procedimiento:

1º Deberá loguearse como root y posicionarse en el directorio donde se encuentran dichos scripts: `/etc/openvpn/easy-rsa/`

```
juanapc@servidor:~$ sudo -s
root@servidor:~# cd /etc/openvpn/easy-rsa/
root@servidor:/etc/openvpn/easy-rsa# 
```

2º Cargará las variables del fichero **/etc/openvpn/easy-rsa/vars** con el siguiente comando: source vars

```
root@servidor:/etc/openvpn/easy-rsa# source vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
root@servidor:/etc/openvpn/easy-rsa#
```

3º Deberá ejecutar el script “**revoke-full**” y especificarle el nombre del usuario que al que se le va a revocar el certificado. Este nombre será el CN del certificado, que ha de estar ubicado en el directorio **/etc/openvpn/easy-rsa/keys/**. Veamos un ejemplo para el usuario profesor2

```
Archivo Editar Ver Terminal Pestañas Ayuda
root@servidor:/etc/openvpn/easy-rsa# ./revoke-full profesor2
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Revoking Certificate 03.
Data Base Updated
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
profesor2.crt: C = ES, ST = CA, L = Cadiz, O = ASIR, OU = Informatica, CN = profesor2, name = EasyRSA, emailAddress = admin@mailasir.inf
error 23 at 0 depth lookup:certificate revoked
```

**Nota:** el error 23 en la última línea nos indica que la validación del certificado revocado falló, esto quiere decir que el certificado fue revocado exitosamente.

El comando revoke-full generará el fichero de la lista de revocación de certificados (CRL) **crl.pem** en el directorio **/etc/openvpn/easy-rsa/keys/**. Este fichero tendrá un enlace simbólico al directorio **/etc/openvpn/** ya que así se encuentra configurado el fichero de configuración del servidor OpenVPN.

Una vez realizado estos pasos, el certificado habrá sido revocado y el usuario profesor2 no podrá conectarse a la VPN.

#### 4. Manual de usuario

Será el usuario el que deba configurar su equipo cliente para poder establecer la conexión con la VPN de ASIR. Para ello, el administrador de la red deberá proporcionarle cuatro ficheros que tendrá que usar. Estos ficheros son:

- Un fichero con los ajustes de la conexión que deberá cargar en el administrador de red del sistema.
- El certificado CA del servidor VPN.
- El certificado (clave pública) del usuario.
- La clave privada del usuario.



ASIR VPN.conf



ca.crt



profesor1.crt



profesor1.key

Una vez que el usuario posea los cuatro ficheros realizará los siguientes pasos con el fin de configurar el administrador de red para poder conectarse a la VPN automáticamente sin necesidad de comandos.

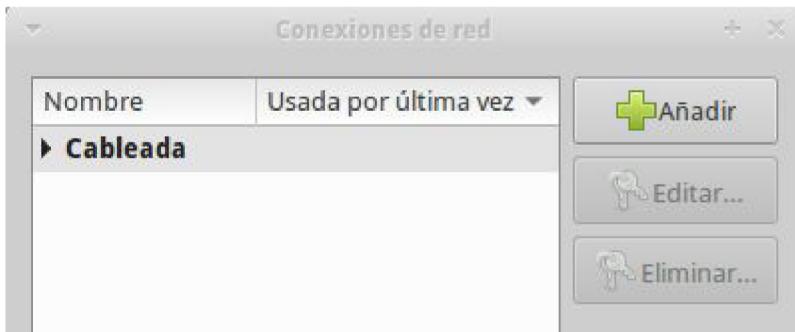
1º Lo primero que debe hacer es instalar el paquete network-manager-openvpn en su sistema: `sudo apt-get install network-manager-openvpn`

```
juanancp@desktop:~$ sudo apt-get install network-manager-openvpn
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

2º Deberá reiniciar el network-manager con el siguiente comando: `sudo restart network-manager`

```
juanancp@desktop:~$ sudo restart network-manager
network-manager start/running, process 15615
juanancp@desktop:~$ 
```

3º Irá al administrador de conexiones, normalmente situado junto al reloj del sistema, desplegará el menú → **Conexiones VPN → Configurar VPN...** Aparecerá una ventana y clicará sobre el botón **Añadir**.



4º Abrirá el menú desplegable de la nueva ventana y elegirá la opción “**Importar una configuración VPN guardada...**” y clicará en el botón **Crear...**



Se abrirá el explorador de ficheros y deberemos de elegir el fichero de configuración para el cliente que le ha suministrado el administrador (ASIR VPN.conf).

5º En la nueva ventana deberá elegir como tipo de autenticación “**Contraseña con certificados (TLS)**”.



Y más abajo deberá introducir sus credenciales usuario/contraseña de LDAP y especificar los certificados que les ha suministrado el administrador.

Nombre de usuario:	profesor1
Contraseña:	*****
Certificado de usuario:	<input type="file"/> profesor1.crt
Certificado CA:	<input type="file"/> ca.crt
Clave privada:	<input type="file"/> profesor1.key
Contraseña de la clave privada:	

Una vez realizado, pulsará en el botón **Guardar...** y ya solo queda establecer la conexión.

6º Para establecer la conexión, irá al **administrador de conexiones** → **Conexiones VPN** → **ASIR VPN**.



Ya se encuentra el usuario conectado a la VPN de ASIR.

## 5. Funcionamiento de la infraestructura creada

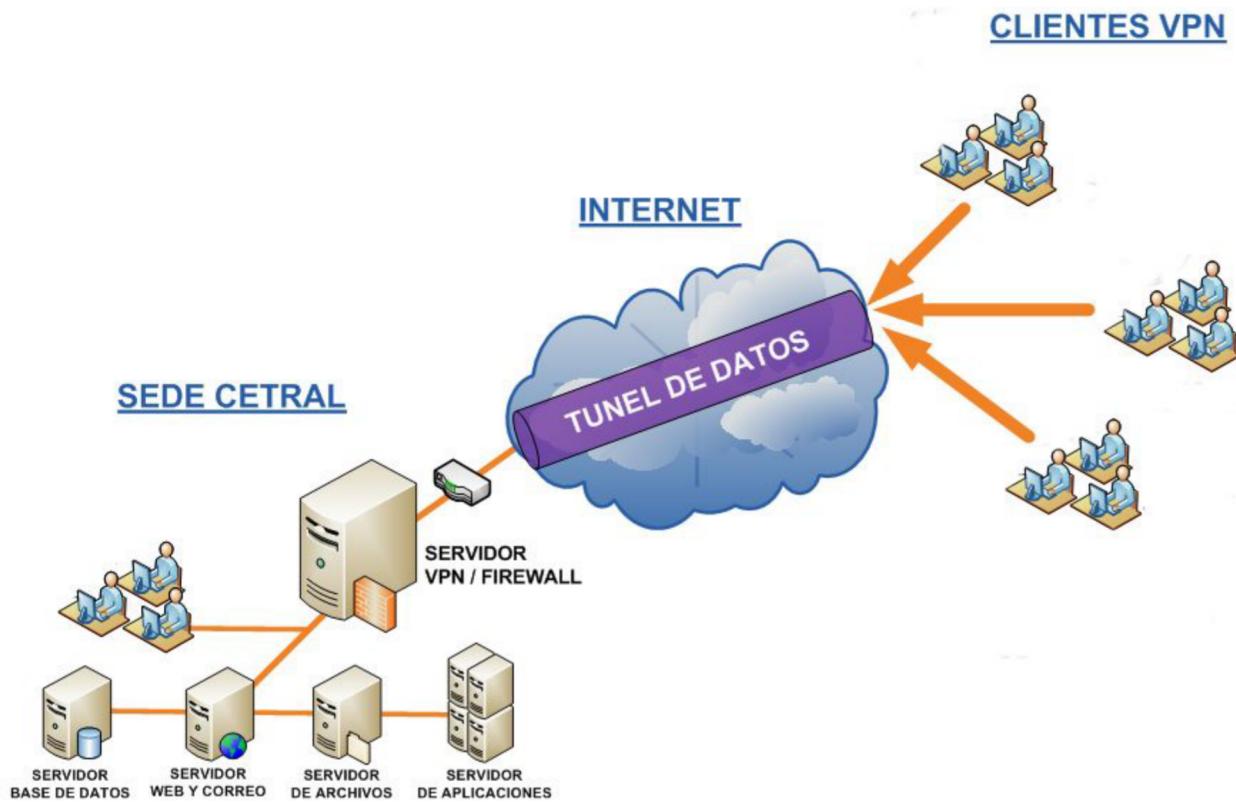
En este punto vamos a analizar como está implementada la infraestructura y como funciona cada uno de sus componentes.

Para ello tendremos en cuenta que el objetivo final del proyecto es crear una VPN en la red de ASIR a la que los usuarios, a los cuales se le permite el acceso, puedan conectarse y hacer uso de los recursos que ofrece la red de forma segura.

Así que vamos a ver cómo funciona OpenVPN (independientemente de FreeRADIUS y OpenLDAP)

Lo primero que debemos saber es el modelo de VPN que hemos realizado: **cliente-servidor**. Es quizás el modelo más usado actualmente y consiste en varios usuarios que se conectan a un servidor remoto utilizando Internet como vínculo de acceso. El cliente se autentica al servidor de acceso remoto, y el servidor se autentica ante el cliente. Una vez autenticados, los clientes tienen un nivel de acceso muy similar al que tendrían si estuvieran conectados a la propia red local donde se encuentra el servidor VPN.

Veamos una imagen de ejemplo:



#### Pero, ¿Qué es lo que ocurre a partir de que se inicia el servicio OpenVPN en el servidor?

Cuando se inicia el servidor, se configura la primera interfaz TUN disponible (tun0) con dirección IP 10.8.0.1 y con una dirección remota falsa 10.8.0.2 a la que no se tendrá acceso. Después de eso, el servidor se pone a escuchar en el puerto UDP 1194 esperando conexiones entrantes.

El cliente se conectará al servidor por este puerto. Después del saludo inicial (handshake) de TLS, haciendo uso de los certificados del cliente y servidor y suponiendo que es el primer cliente que se conecta a la VPN, se le asigna una dirección IP 10.8.0.6 (en realidad se le asigna un pequeño rango de direcciones que va desde la 10.8.0.4 a la 10.8.0.7). Seguidamente, en el cliente se configura la primera interfaz TUN disponible (tun0) utilizando esta información y después de esto se establece la conexión VPN.

Así funciona OpenVPN independientemente pero, ¿Qué ocurre realmente en nuestra infraestructura implantada donde además de OpenVPN entra en juego FreeRADIUS y OpenLDAP?

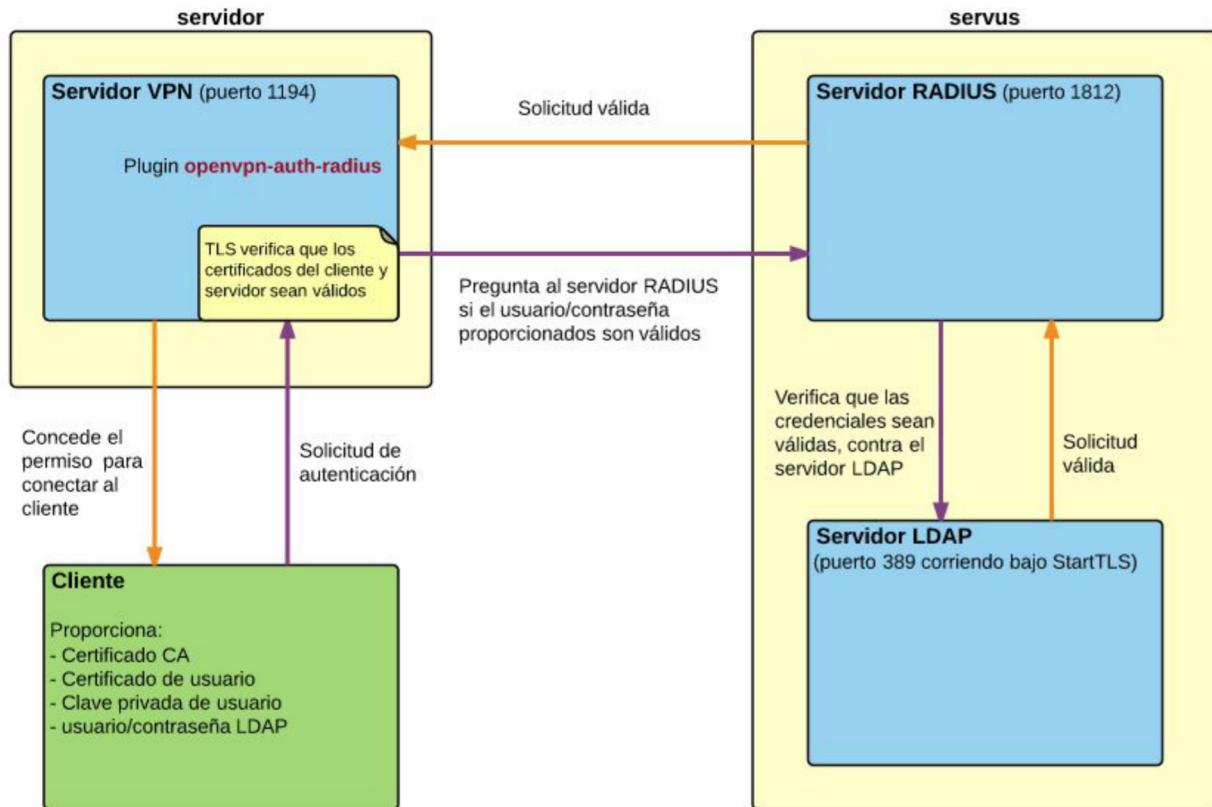
OpenVPN se encarga de crear el servidor VPN y establecer la conexión con los clientes, pero además, se encarga de validar mediante TLS los certificados entre cliente y servidor para aceptar o rechazar la conexión. Nosotros además, hemos implementado FreeRADIUS y OpenLDAP para aumentar la seguridad y un mejor control de los usuarios que se van a conectar a la VPN.

Con esto conseguimos realizar una autenticación por dos factores:

1. Validación de certificados mediante TLS por parte de OpenLDAP
2. Validación de usuario/contraseña por parte de FreeRADIUS + OpenLDAP, los cuales la comunicación entre sí también está cifrada por TLS.

Se han de cumplir los dos factores para realizar una conexión con éxito, si alguno fallara, la conexión a la VPN será rechazada.

Vamos a ver un esquema gráfico de un supuesto caso de éxito donde lo entenderemos mejor:



En resumen:

1. El cliente le manda una solicitud de autenticación a OpenVPN proporcionándole los certificados y usuario/contraseña.
2. OpenVPN a través de TLS, verifica que los certificados sean válidos y a través del plugin `openvpn-auth-radius`, le pregunta al FreeRADIUS si el usuario/contraseña que el cliente le ha proporcionado son válidos.
3. FreeRADIUS recoge la solicitud y verifica las credenciales contra OpenLDAP, a través de una conexión segura al puerto 389 bajo TLS.
4. OpenLDAP le envía una respuesta a FreeRADIUS validando la solicitud.
5. FreeRADIUS le envía una respuesta a OpenVPN validando la solicitud.
6. OpenVPN le concede el permiso al cliente de conectarse a la VPN.
7. Se inician los pasos de funcionamiento de OpenVPN descritos arriba.

## **6. Planificación del proyecto**

La planificación del proyecto ha sido dividida en las siguientes fases:

1. Documentación y estudio de diferentes soluciones para abordar el proyecto.
2. Creación del entorno de trabajo donde irá implantada la infraestructura.
3. Implementación, pruebas y solución de errores de las distintas soluciones elegidas.
  - a. OpenLDAP.
  - b. FreeRADIUS.
  - c. OpenVPN.
4. Interconexión de las distintas soluciones, pruebas y solución de errores.
5. Creación de la documentación del sistema implantado.

## **7. Conclusiones finales**

La realización de este proyecto ha sido un trabajo laborioso debido a la escasa documentación que hay sobre la infraestructura creada, OpenVPN + FreeRADIUS + OpenLDAP, como conjunto en sí. La metodología de trabajo ha consistido en buscar documentación sobre todas las partes implicadas por separado, entenderlas (exponiendo las dudas al profesor) y unirlas realizando prueba tras prueba hasta conseguir el resultado deseado. Como apunte debo decir que los **logs** te pueden salvar la vida.

Opino que una infraestructura como la que se ha conseguido crear sería viable ponerla en producción, quizás cuidando algunos detalles específicos para el escenario en el que se fuera a implantar.

### **7.1. Grado de cumplimiento con los objetivos fijados**

Valorando la realización de todo el proyecto, pienso que el grado de cumplimiento con los objetivos fijados en un principio es casi total. El objetivo principal, que trataba de crear una VPN y que usuarios remotos pudieran conectarse a ella de forma segura mediante un sistema de autenticación y autorización, está conseguido al 100%.

El único “inconveniente”, por así llamarlo, es que en un principio se propuso que el sistema de autenticación y autorización para los usuarios fuera gestionado completamente por FreeRADIUS, pero el plugin encargado de hacer la comunicación entre OpenVPN y FreeRADIUS no permite más que validaciones del tipo usuario/contraseña. Esto es debido a que OpenVPN implementa su propio sistema de seguridad (TLS/SSL), por lo tanto se optó por la opción de autenticación de doble factor, haciendo uso de TLS con OpenVPN para validar los certificados y FreeRADIUS para validar las credenciales usuario/contraseña.

### **7.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado**

Una propuesta de modificación puede ser implantar la VPN con IPsec ya que con la llegada de IPv6 no será necesario tener las redes NATeadas, además IPsec, mediante plugins parece ser que puede delegar el sistema de autenticación y autorización a un servidor RADIUS, cosa que con OpenVPN no se puede.

También podemos proponer la creación de una infraestructura para el manejo de los certificados de los usuarios (creación, almacenamiento, revocación, etc) un poco más elaborada.

## **8. Bibliografía**

La bibliografía/webografía consultada para llevar a cabo este proyecto ha sido la siguiente:

**Libro sobre VPN →**

[Redes Privadas Virtuales \(editorial Ra-Ma\)](#)

**Libro sobre OpenVPN →**

[OpenVPN 2 Cookbook \(editorial Packt Publishing\)](#)

**Web oficial OpenVPN →**

<https://openvpn.net/>

**Libro sobre RADIUS →**

[RADIUS / AAA / 802.1x \(editorial Ra-Ma\)](#)

**Web oficial FreeRADIUS →**

<http://freeradius.org/>

**Guía OpenVPN + FreeRADIUS + MySQL/LDAP →**

<https://www.roessner-network-solutions.com/beliebte-seiten-und-artikel/openvpn-radius-mysqlldap-howto/>

**Configuración de FreeRADIUS plugin y OpenVPN →**

<http://safesrv.net/setup-freeradius-plugin-and-openvpn-source/>

**Script CN como nombre de usuario →**

<http://my.galagzee.com/2013/09/21/openvpn-with-freeradius-common-name-as-username/>

**Guía WPA2 Enterprise + FreeRADIUS con EAP-TLS →**

<http://cubicspot.blogspot.com.es/2013/04/setting-up-wpa2-enterprise-aes-with.html>

**Web oficial OpenLDAP →**

<http://www.openldap.org/>

**Guia oficial de Ubuntu para OpenLDAP →**

<https://help.ubuntu.com/lts/serverguide/openldap-server.html>

**Manual instalación y configuración OpenLDAP →**

<http://somebooks.es/?p=3445>

## ANEXO-LDAP

### **config.ldif**

```
dn: cn=config
changetype: modify

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootDN
olcRootDN: cn=admin,cn=config

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootPW
olcRootPW: 0688

dn: olcDatabase={0}config,cn=config
changetype: modify
delete: olcAccess
```

### **backend.ldap.inf.ldif**

```
# Load dynamic backend modules
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulepath: /usr/lib/ldap
olcModuleload: back_hdb.la

# Database settings
dn: olcDatabase=hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: hdb
olcSuffix: dc=ldap,dc=inf
olcDbDirectory: /var/lib/ldap
olcRootDN: cn=admin,dc=ldap,dc=inf
olcRootPW: 0688
olcDbConfig: set_cachesize 0 2097152 0
olcDbConfig: set_lk_max_objects 1500
olcDbConfig: set_lk_max_locks 1500
olcDbConfig: set_lk_max_lockers 1500
olcDbIndex: objectClass eq
olcLastMod: TRUE
olcDbCheckpoint: 512 30
olcAccess: to attrs=userPassword by dn="cn=admin,dc=ldap,dc=inf" write by anonymous auth by self write by * none
olcAccess: to attrs=shadowLastChange by self write by * read
olcAccess: to dn.base="" by * read
olcAccess: to * by dn="cn=admin,dc=ldap,dc=inf" write by * read
```

### **frontend.ldap.inf.ldif**

```
# Create top-level object in domain
dn: dc=ldap,dc=inf
objectClass: top
objectClass: dcObject
objectclass: organization
o: Example Organization
dc: ldap
description: LDAP Example

# Admin user.
dn: cn=admin,dc=ldap,dc=inf
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
```

```
userPassword: 0688

dn: ou=Usuarios,dc=ldap,dc=inf
objectClass: organizationalUnit
ou: Usuarios

dn: ou=Grupos,dc=ldap,dc=inf
objectClass: organizationalUnit
ou: Grupos

dn: ou=Alumnos,ou=Usuarios,dc=ldap,dc=inf
objectClass: organizationalUnit
ou: Alumnos

dn: ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
objectClass: organizationalUnit
ou: Profesores

dn: cn=grupoAlumnos,ou=Grupos,dc=ldap,dc=inf
objectClass: posixGroup
cn: grupoAlumnos
gidNumber: 50000

dn: cn=grupoProfesores,ou=Grupos,dc=ldap,dc=inf
objectClass: posixGroup
cn: grupoProfesores
gidNumber: 50001

dn: uid=alumno1,ou=Alumnos,ou=Usuarios,dc=ldap,dc=inf
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alumno1
sn: Apellido de alumno 1
givenName: Alumno 1
cn: Alumno 1
displayName: Alumno 1
uidNumber: 5000
gidNumber: 50000
userPassword: alumno1
gecos: Usuario 1
loginShell: /bin/bash
homeDirectory: /home/pub/alumnos/alumno1
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877

dn: uid=alumno2,ou=Alumnos,ou=Usuarios,dc=ldap,dc=inf
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alumno2
sn: Apellido de alumno 2
givenName: Alumno 2
cn: Alumno 2
displayName: Alumno 2
uidNumber: 5001
gidNumber: 50000
userPassword: alumno2
gecos: Usuario 2
loginShell: /bin/bash
homeDirectory: /home/pub/alumnos/alumno2
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877
```

```
dn: uid=profesor1,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: profesor1
sn: Apellido de profesor 1
givenName: Profesor 1
cn: Profesor 1
displayName: Profesor 1
uidNumber: 6000
gidNumber: 50001
userPassword: profesor1
gecos: Profesor 1
loginShell: /bin/bash
homeDirectory: /home/pub/profesores/profesor1
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877

dn: uid=profesor2,ou=Profesores,ou=Usuarios,dc=ldap,dc=inf
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: profesor2
sn: Apellido de profesor 2
givenName: Profesor 2
cn: Profesor 2
displayName: Profesor 2
uidNumber: 6001
gidNumber: 50001
userPassword: profesor2
gecos: Profesor 2
loginShell: /bin/bash
homeDirectory: /home/pub/profesores/profesor2
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877

dn: cn=grupoUsuarios,ou=Grupos,dc=ldap,dc=inf
objectClass: posixGroup
cn: grupoUsuarios
gidNumber: 50002
memberUid: alumno1
memberUid: alumno2
memberUid: profesor1
memberUid: profesor2

dn: cn=grupoVPN,ou=Grupos,dc=ldap,dc=inf
objectClass: posixGroup
cn: grupoVPN
gidNumber: 50003
memberUid: profesor1
memberUid: profesor2
```

## ANEXO-FREERADIUS

### /etc/freeradius/modules/ldap

```
ldap {
    server = "servus"
    identity = "cn=admin,dc=ldap,dc=inf"
    password = 0688
    basedn = "ou=Profesores,ou=Usuarios,dc=ldap,dc=inf"
    filter = "(uid=%{${Stripped-User-Name}:-%{User-Name}})"
    base_filter = "(objectclass=radiusprofile)"
    ldap_connections_number = 5
    timeout = 4
    timelimit = 3
    net_timeout = 1

    tls {
        start_tls = yes
        cacertfile      = /etc/ssl/certs/cacert.pem
        cacertdir       = /etc/ssl/certs/
        certfile        = /etc/ssl/certs/servus_slapd_cert.pem
        keyfile         = /etc/ssl/private/servus_slapd_key.pem
        require_cert   = "demand"
    }

    dictionary_mapping = ${confdir}/ldap.attrmap

    edir_account_policy_check = no

    keepalive {
        idle = 60
        probes = 3
        interval = 3
    }
}
```

### /etc/freeradius/radiusd.conf

```
prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/freeradius
raddbdir = /etc/freeradius
radacctdir = ${logdir}/radacct

name = freeradius

confdir = ${raddbdir}
run_dir = ${localstatedir}/run/${name}

db_dir = ${raddbdir}

libdir = /usr/lib/freeradius

pidfile = ${run_dir}/${name}.pid

user = freerad
group = freerad

max_request_time = 30

cleanup_delay = 5

max_requests = 1024

listen {
    type = auth
```

```

    ipaddr = *
    port = 0
}

listen {
    ipaddr = *
    port = 0
    type = acct
}

hostname_lookups = no

allow_core_dumps = no

regular_expressions = yes
extended_expressions = yes

log {
    destination = files
    file = ${logdir}/radius.log
    syslog_facility = daemon
    stripped_names = no

    auth = yes
    auth_badpass = yes
    auth_goodpass = yes
}

checkrad = ${sbindir}/checkrad

security {
    max_attributes = 200
    reject_delay = 1
    status_server = yes
}

proxy_requests = no
#$INCLUDE proxy.conf

$INCLUDE clients.conf

thread pool {
    start_servers = 5
    max_servers = 32
    min_spare_servers = 3
    max_spare_servers = 10
    max_requests_per_server = 0
}

modules {
    $INCLUDE ${confdir}/modules/
    $INCLUDE eap.conf
}

instantiate {
    exec
    expr
    expiration
    logintime
}

$INCLUDE policy.conf

$INCLUDE sites-enabled/

```

### /etc/freeradius/clients.conf

```
client ldap {
    ipaddr = 192.168.2.220
    secret = Periheli0/
    nastype = other
}

client OpenVPN {
    ipaddr = 192.168.2.210
    secret = Periheli0/
    nastype = other
}
```

### /etc/freeradius/sites-available/ldap

```
authorize {
    preprocess
    ldap
    expiration
    logintime
}

authenticate {
    Auth-Type LDAP {
        ldap
    }
}

preacct {
    preprocess
    acct_unique
}

accounting {
    detail
    radutmp
    exec
    attr_filter.accounting_response
}

session {
    radutmp
}

post-auth {
    ldap
    Post-Auth-Type REJECT {
        attr_filter.access_reject
    }
}

pre-proxy {

}

post-proxy {
}
```

### /etc/openvpn/cn\_radius\_auth.sh

```
#!/bin/bash

# $1 provides the temp file name provided by OpenVPN
# file has two lines: username and password, as entered by the user.
# We get the username from the user cert's CN (available via an envvar).

export PATH="/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin"

if [ ! -z ${common_name} ]; then
    username=`echo ${common_name}`
```

```

else
    username="-"
fi

if [ ! -z $1 ] && [ -f $1 ]; then
    password=`tail -1 $1`
else
    password="-"
fi

radius_server=servus
# shared secret for localhost (or your RADIUS server) from /etc/freeradius/
shared_secret="Periheli0"

AUTHCHECK=`cat << EOF | /usr/bin/radclient -s ${radius_server} auth ${shared_secret} | grep approved | tr -d '\n' | tail -c 1
User-Name=${username}
User-Password=${password}
EOF`

if [[ $AUTHCHECK = 1 ]]; then
    exit 0
else
    exit 1
fi

```

## ANEXO-OPENVPN

### /etc/openvpn/server.conf

```
port 1194
proto udp
dev tun
server 10.8.0.0 255.255.255.0

push "route 192.168.2.0 255.255.255.0"
topology subnet

ca ca.crt
cert servidor.crt
key servidor.key
dh dh2048.pem

#crl-verify crl.pem

ifconfig-pool-persist ipp.txt
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 3

# openvpn-auth-radius
plugin /usr/lib/openvpn/radiusplugin.so /etc/openvpn/radiusplugin.cnf

# script cn verify
tmp-dir /dev/shm
auth-user-pass-verify /etc/openvpn/cn_radius_auth.sh via-file

daemon
log-append /var/log/openvpn.log
```

### /etc/openvpn/radiusplugin.cnf

```
NAS-Identifier=OpenVpn

Service-Type=5

Framed-Protocol=1

NAS-Port-Type=5

NAS-IP-Address=127.0.0.1

OpenVPNConfig=/etc/openvpn/server.conf

subnet=255.255.255.0

overwriteccfiles=true

nonfatalaccounting=false

server
{
    acctport=1813
    authport=1812
    name=192.168.2.220
    retry=1
    wait=1
    sharedsecret=Periheli0/
}
```