Conexión de terminales a VIVAit mediante OpenVPN

Fecha : 5 de octubre de 2017	**Número de revisión**: Versión 1
Objeto del documento : Pruebas y recomendaciones para conexiones de terminales a VIVAit mediante redes IP públicas (internet)	
Revisiones 1. Documento inicial	

Contenido

Ilustraciones

1 Introducción

La plataforma de comunicaciones unificadas y contact center VIVAit está basada totalmente en comunicaciones IP basadas en estándares (SIP, RTP) entre los terminales y los servicios centrales

Si bien es habitual que los terminales y los servicios centrales se encuentren conectados a través de una red IP privada del cliente (ya sea por estar en la misma red de área local, en una red IP privada o en una red IP pública con VPNs entre sedes) cada vez es más demandado la conexión de los terminales a través de redes IP públicas de manera nativa (como internet)

Una de las opciones que en determinados momentos se han adoptado (no mdtel) han sido las de abrir el protocolo SIP a la red IP pública para permitir un registro abierto desde la misma; este mecanismo ha evidenciado múltiples problemas de seguridad.

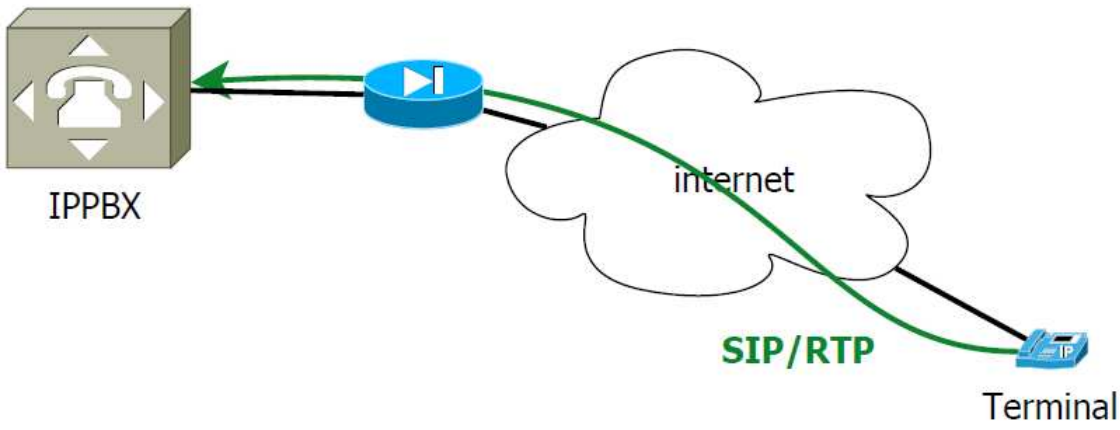


Ilustración 1.- SIP sobre internet

La alternativa tecnológica que vemos más sencilla y segura de implementar se basa en el uso de terminales telefónicos que dispongan de cliente vpn en su firmware, y que permitan establecer una conectividad segura a través de redes IP públicas

Multitud de fabricantes de terminales, entre ellos Yealink y Grandstream, están implementando clientes OpenVPN dentro de su firmware, por lo que podemos considerarlo una solución global y no propietaria

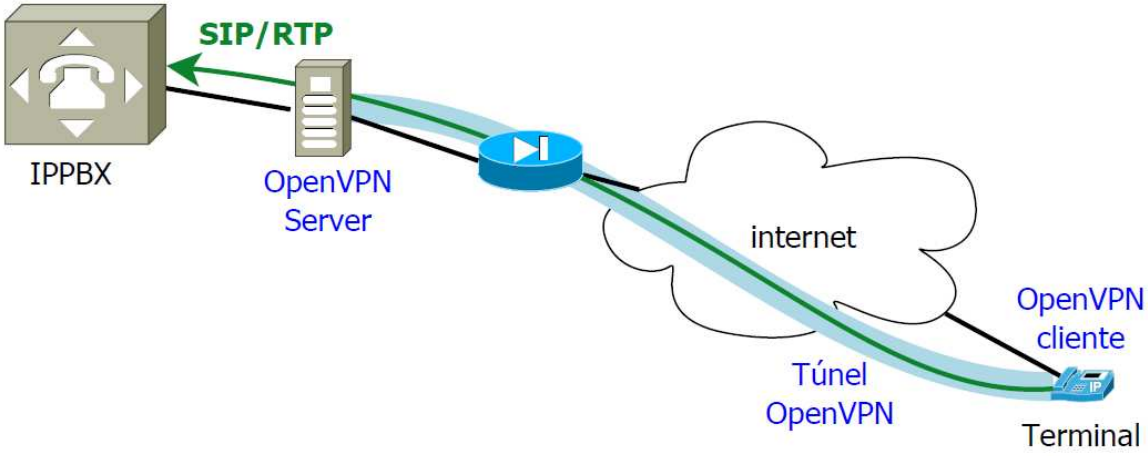


Ilustración 2.- SIP sobre OPenVPN

2 Solución global

La siguiente figura muestra de manera resumida una arquitectura de referencia para conectividad con sistemas VIVAit; nótese que se ubica en la misma máquina los procesos "OpenVPN server" e "IPpbx" de la ilustración anterior

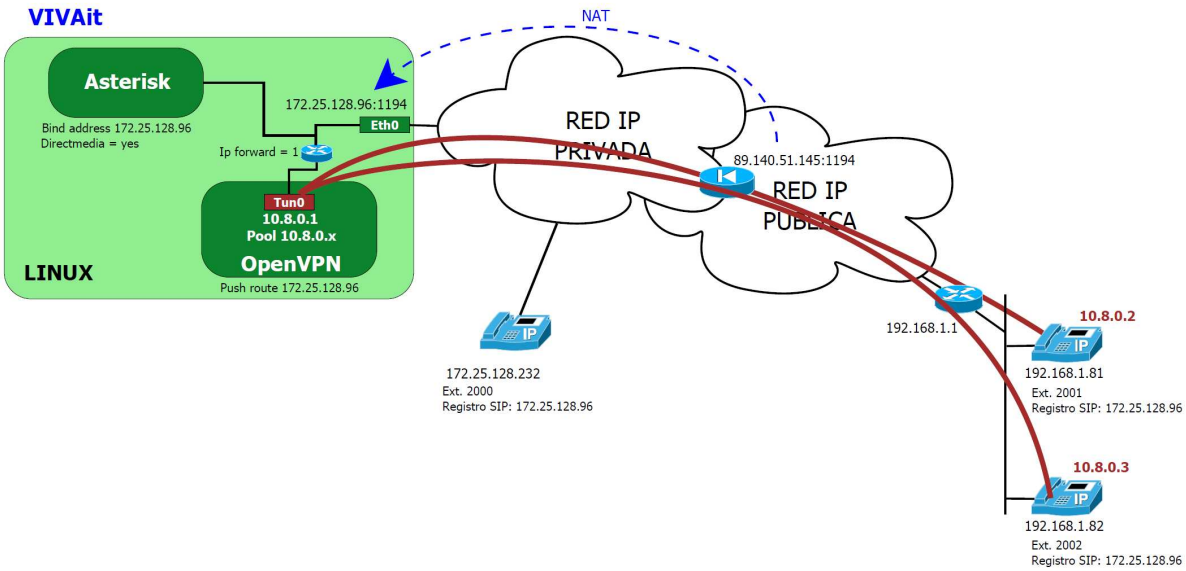


Ilustración 3.- Arquitectura de referencia

Sobre la arquitectura mostrada pueden existir múltiples variaciones relacionadas fundamentalmente con cuestiones de networking e incluso de seguridad

En la arquitectura recomendada el proceso de cifrado se realizará mediante seguridad por clave pública (modo SSL/TLS) usando certificados de cliente y servidor

3 Procedimientos habituales

3.1 Instalación de OpenVPN en VIVAit

La instalación de OpenVPN en VIVAit (en realidad en cualquier servidor Linux) pasa por:

- Descargar e instalar el paquete correspondiente

  apt-get install openvpn

- Configurar adecuadamente el servicio
- Podemos confirmar la correcta activación del servicio mediante el comando

  service openvpn status

Consideraciones de utilidad:

Elemento	Consideraciones
Servidor OpenVPN	Topology subnet (obligatorio) Comentar comp-lzo (no usar, recomendado) Cypher AES 128 ip forward = 1
Servidor OpenVPN, a cada terminal	Clave pública CA Clave privada terminal Clave pública terminal Generar config cliente OpenVPN Empaquetar todo (".tar")
Servidor OpenVPN, estado y trazas	/var/log/openvpn-status.log /var/lib/openvpn/ipp.txt /var/log/openvpn.log

3.2 Generación de claves

Como se ha indicado anteriormente, el proceso de cifrado se realizará mediante el uso de certificados en cliente y servidor

El proceso de generación de claves pasará por:

1. Generar una CA en el propio servidor OpenVPN, obteniendo el fichero con su clave pública
2. Generar para cada terminal una pareja clave privada/clave pública
3. Estas tres claves serán subidas al terminal durante el proceso de aprovisionamiento

3.2.1 Creación de CA y certificado para el servidor

- instalamos el paquete easy-rsa.

  apt-get install easy-rsa

- Por defecto los comandos de easy-rsa se encuentran en /usr/share/easy-rsa, los copiaremos al directorio /etc/openvpn/easy-rsa:

  mkdir /etc/openvpn/easy-rsa

  cp –R /usr/share/easy-rsa /etc/openvpn

- En /etc/openvpn/easy-rsa/ modificamos vars, configurando cada export al valor que nos interese.

  vi var

- A continuación en el directorio /etc/openvpn/easy-rsa/ introduciremos los siguientes comandos con el fin de inicializar la autoridad de certificación:

  source ./vars

  ./clean-all

- El siguiente comando generará el certificado CA y la clave de CA

  ./build-ca

  Quedando estas ubicadas en "/etc/openvpn/easy-rsa/keys"

- A continuación creamos el certificado y la clave para el servidor:

  ./build-key-server server

  Quedando también ubicado en "/etc/openvpn/easy-rsa/keys"

- Después, generamos los parámetros Diffie-Hellman:

  ./build-dh

- Los certificados y las claves necesarias se encuentran en el directorio /etc/openvpn/easy-rsa/keys/,del cual moveremos ciertos certificados y claves al directorio /etc/openvpn.

  cd /etc/openvpn/easy-rsa/keys

  mv ca.crt dh2048.pem server.key server.crt /etc/openvpn/

El fichero que copiaremos posteriormente a cada terminal será el ca.crt, que contiene la clave pública

3.2.2 Creación de certificados de cada cliente

Para la creación de los certificados conteniendo la clave privada y la clave pública en cada cliente procederemos de la siguiente manera:

- Creamos un certificado y una clave para cada cliente de la VPN.

  ./build-key nombre_cliente

  Donde el parámetro "nombre_cliente" irá variando

  Los certificados quedarán ubicados en "/etc/openvpn/easy-rsa/keys"

3.3 Generación de fichero ".tar" de cada cliente

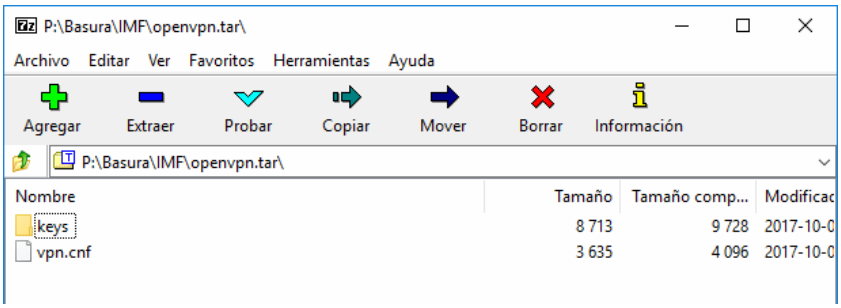
El procedimiento que a continuación se describe es específico para el terminal Yealink T21P; es necesario revisar el procedimiento para cada marca y modelo

A cada cliente se le proporcionará todo el contenido necesario para establecer la VPN en un fichero ".tar" que contendrá:

- Clave pública de la CA
- Clave privada del cliente
- Clave pública del cliente
- Configuración OpenVPN del cliente

Para cada cliente generaremos un fichero openvpn.tar (con cualquier aplicación) que contendrá:

- Carpeta keys con los certificados ca.crt, nombre_cliente.crt, nombre_cliente.key (clave pública de la CA, clave privada del cliente y clave pública del cliente)
- Plantilla vpn.cnf (configuración OPenVPN del cliente)
- Cada ".tar" generado deberá ser copiado en el servidor TFTP en la carpeta "/var/lib/phoneprov-tftp/bin"



3.4 Aprovisionamiento de terminales

Existen al menos dos posibilidades de aprovisionamiento de terminales

- Usar procedimientos de aprovisionamiento en la nube de cada fabricante de terminales (p. ej Yealink RPS)
- Usar procedimientos internos a VIVAit para el aprovisionamiento, usando el mecanismo habitual DHCP/TFTP

Por cuestiones de aseguramiento del funcionamiento, y para hacerlo independiente de la conexión a internet con el servicio del fabricante, se recomienda utilizar el segundo mecanismo descrito. Con este mecanismo "fuera de línea":

- El aprovisionamiento se realiza en una red específicamente habilitada para el aprovisionamiento; el terminal ha de ser desempaquetado y configuado de manera previa al envío
- Tras el proceso, el terminal queda totalmente configurado tantos en sus parámetros para la conexión OpenVPN como en sus parámetros para la configuración telefónica (excepción TFTP comentada más adelante)
- No se puede probar el resultado con facilidad, puesto que realizamos en una red una configuración para un destino final diferente de dicha red
- Los ficheros relevantes que se transfieren al terminal telefónico durante el proceso son:
    - Xml con configuración global de terminal
    - ".tar" con claves y configuración de OpenVPN, según se ha explicado en el apartado anterior
- Una vez instalado en su destino final, se habrá de cambiar a mano (desde la configuración web del terminal) su servidor TFTP para que quede el definitivo de la instalación y permitir reaprovisionamientos masivos

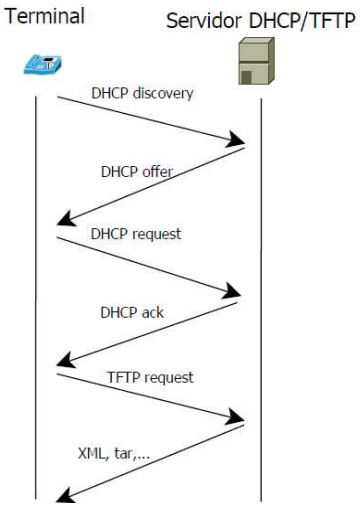


Ilustración 4.- Proceso de aprovisionamiento

4 Ejemplos de ficheros de configuración

4.1 server.conf

Se resaltan en negrita los parámetros modificados con respecto a la configuración por defecto; hacemos especial mención a la posibilidad de que los distintos clientes conectados al servidor se puedan ver entre sí, ya que serán terminales telefónicos que han de poder hablar (parámetro client-to-client)

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

# Which local IP address should OpenVPN
# listen on? (optional)
local 172.25.128.96

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

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
```

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

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one.  On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

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

management localhost 7505

# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh1024.pem 1024
```

```
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh /etc/openvpn/dh2048.pem

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

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

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

# Configure server mode for ethernet bridging
# using a DHCP-proxy, where clients talk
# to the OpenVPN server-side DHCP server
# to receive their IP address allocation
# and DNS server addresses.  You must first use
# your OS's bridging capability to bridge the TAP
# interface with the ethernet NIC interface.
# Note: this mode only works on clients (such as
# Windows), where the client-side TAP adapter is
# bound to a DHCP client.
;server-bridge

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

```
push "route 172.25.128.0 255.255.254.0"
;push "route 10.0.0.0 255.0.0.0"

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

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN.  This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients.  There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#     group, and firewall the TUN/TAP interface
#     for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#     modify the firewall in response to access
#     from different clients.  See man
#     page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
```

```
;push "redirect-gateway def1 bypass-dhcp"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
;push "dhcp-option DNS 208.67.222.222"
;push "dhcp-option DNS 208.67.220.220"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names.  This is recommended
# only for testing purposes.  For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
```

```
# on the server and '1' on the clients.
;tls-auth /etc/openvpn/ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC        # Blowfish (default)
cipher AES-128-CBC   # AES
;cipher DES-EDE3-CBC  # Triple-DES

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
;comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
user nobody
group nogroup

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status /var/log/openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
log         /var/log/openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
```

```
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages.  At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

4.2 Configuración cliente OpenVPN en plantilla del terminal

En los ficheros de plantilla de los terminales hemos de indicar que usarán Openvpn y de donde coger el ".tar" con las claves y configuración del cliente OpenVPN

Las siguientes lineas se han añadido a la plantilla de aprovisionamiento de un terminal Yealink T21P

```
network.vpn_enable = 1
openvpn.url = tftp://servidor_tftp/openvpn.tar
```

Donde "servidor_tftp" será sustituido por la IP o nombre del servidor TFTP

DUDA : COMO GESTIONAR UE TODOS LOS FICHEROS SE LLAMAN "OPENVPN.TAR"...NO PODRIAMOS TENER UNA PLANTILLA POR MODELO, SINO QUE SERIA UN FICHERO DIFERENTE POR TELEFONO

vpn.cnf

A partir del fichero de ejemplo ubicado en

/usr/share/doc/openvpn/examples/sample-config-files/ client.conf

Generamos la plantilla de configuración de cliente OpenVPN para los terminales

```
##############################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#                                            #
# This configuration can be used by multiple #
# clients, however each client should have   #
# its own cert and key files.                #
#                                            #
# On Windows, you might want to rename this  #
# file so it has a .ovpn extension           #
##############################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client
setenv SERVER_POLL_TIMEOUT 4
# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap0
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap
dev-type tun
# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
;proto tcp
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 89.140.51.145 1194
```

```
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nogroup
# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca /config/openvpn/keys/ca.crt
cert /config/openvpn/keys/yealink.crt
key /config/openvpn/keys/yealink.key

# Verify server certificate by checking
# that the certicate has the nsCertType
```

```
# field set to "server".  This is an
# important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server".  The build-key-server
# script in the easy-rsa folder will do this.
ns-cert-type server
reneg-sec 604800
sndbuf 100000
rcvbuf 100000
auth-retry nointeract

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth /config/openvpn/keys/ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
cipher AES-128-CBC

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
;comp-lzo no

# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```