Taller de Telemática

Interconexión de redes con VPN

Apéndice

Autores:

Pablo García

Marcos Rodriguez

Victor Alem

Docentes:

Ariel Sabiguero

Leandro Scasso

Índice

Instalación de OpenWrt

En WRT54GL

En este caso, el router ya tenía la versión backfire 10.03.1-rc4 instalada y no nos fue posible cambiar el firmware a través de la página de administración de OpenWrt. Dado esto seteamos unos segundos de booteo al arranque del sistema con las siguientes lineas:

```
$ nvram set boot_wait=on
$ nvram set boot_time=10
$ nvram commit && roboot ; exit
```

Luego procedemos a cargarle la imagen vía tftp haciendo lo siguiente:

Abrimos un terminal de comandos, esnchufamos y enchufamos el router y escribimos en la consola:

```
$ tftp 192.168.1.1
tftp> binary
tftp> trace
tftp> put openwrt-wrt54g-squashfs.fin
```

Luego de esto, esperamos a que se reinicie el router (puede tardar un par de minutos) y tendremos instalada la nueva imagen de OpenWrt.

En Tp-Link TL-WR1043ND

Usamos la distribución OPENWRT 10.03.1-rc5. Se puede descargar desde el sitio oficial.

Mediante la interfaz web, que viene por defecto con el router, en la opción de “Firmware Upgrade” tenemos la opción de actualizar el firmware por OPENWRT.

Luego de cargada el firmware de OPENWRT, reiniciamos el router.

Con el nuevo firmware cargado, desde una consola entramos por telnet para darle una contraseña al usuario root (único usuario del firmware):

```
$ telnet 192.168.1.1.1
$ passwd
$ exit
```

Por consola nos volvemos a loguear pero por ssh:

```
$ ssh root@192.168.1.1
```

Instalación y configuración del Servicio DNS Dinámico (DDNS)

Previo

Antes de instalar el cliente DDNS, debemos estar registrados en un servidor DDNS, como por ejemplo dyndns.org, no-ip.com, etc.

Instalación de los paquetes necesarios

Será necesario el paquete "`luci-app-ddns`", para instalarlo procedemos de la siguiente forma:

ingresamos al router:

```
$ ssh root@IP
```

Actualizamos la lista de paquetes:

```
$ opkg update
```

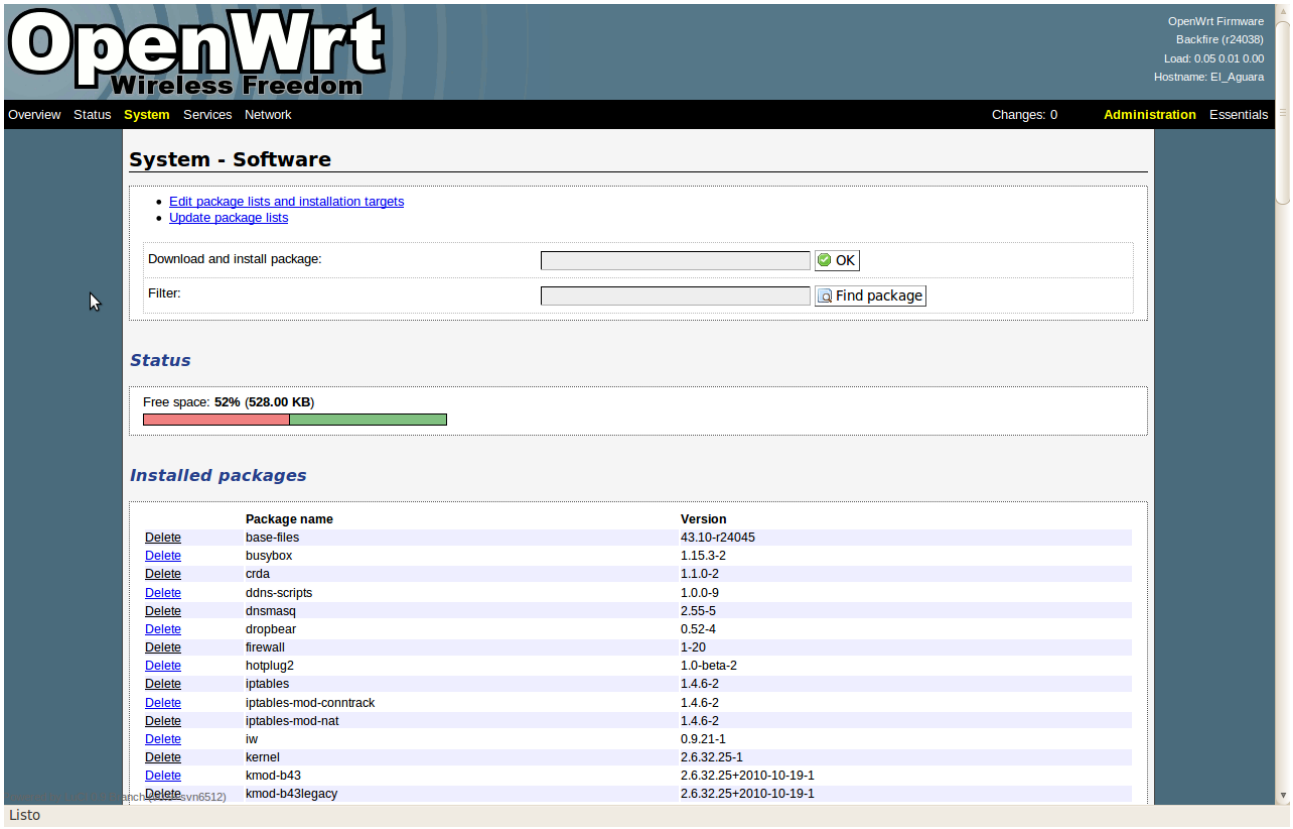
instalamos el paquete:

```
$ opkg install luci-app-ddns
```

Este procedimiento se puede realizar desde la página web de administración de OpenWrt de la siguiente forma:

Abrimos un navegador e ingresamos al router (http://192.168.1.1 por ejemplo), seleccionamos la vista de administración (sección superior izquierda) y accedemos al menú *System -> Software* como se muestra en la siguiente imagen:

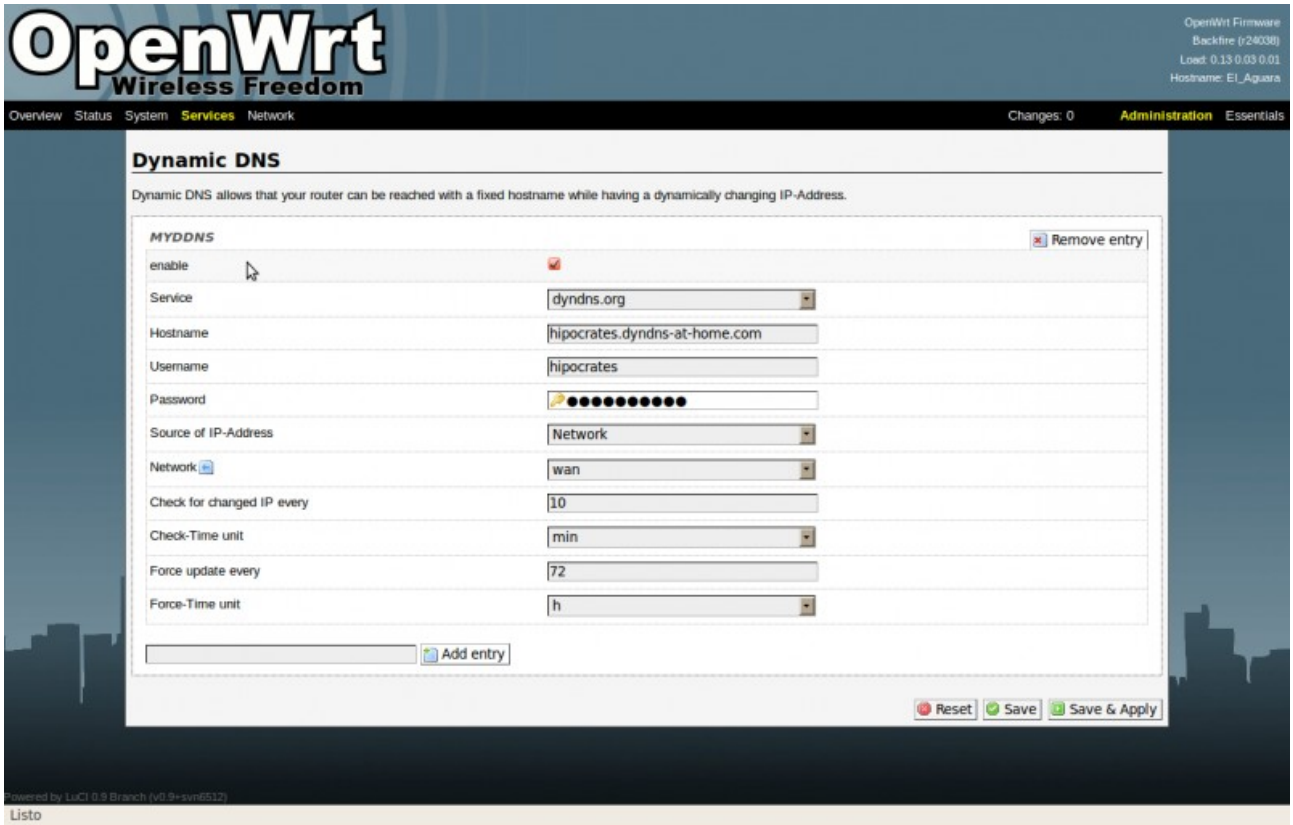
En la siguiente página, buscamos el paquete luci-app-ddns y lo instalamos:



Configuración

La configuración la haremos mediante la aplicación web. Luego de instalado el paquete nos dirigimos al menú “Services” y veremos que aparece un sub menú llamado Dynamic DNS,

ingresamos en él y procedemos a configurar el servicio.



Nos aparecerán variables debemos setear con los valorec correspondientes. Las variables son:

enable: activa o desactiva el servicio.

Service: Servidor ddns utilizado.

Hostname: nombre de dicho dominio.

Username: usuario en el servidor ddns.

Password: contraseña en el servidor ddns.

Source of IP-Address: Quien tiene la IP pública, generalmente generalmente Network aunque también tiene las opciones "Interface" y "URL".Network: (depende de la opción anterior), en este caso wan.

C**heck for changed IP every**: tiempo en el cual verifico si no cambié de IP, yo configuré 10.

Check Time unit: unidades de la variable anterior, este caso, min.

Force update every: cada cuanto tiempo forzar una actualización, 72.

Force-Time unit: unidad de la variable anterior, en este caso h.

También es posible instalar únicamente el paquete “ddns-script” y configurar el archivo /etc/config/ddns . La diferencia es que de esta forma no contamos con la aplicación web para configurar el ddns. El archivo antes mencionado queda de la siguiente manera:

```
config 'service' 'myddns'
       option 'ip_source' 'network'
       option 'ip_network' 'wan'
       option 'force_unit' 'hours'
       option 'check_interval' '10'
       option 'check_unit' 'minutes'
       option 'enabled' '1'
       option 'service_name' 'dyndns.org'
       option 'domain' 'elaguara.dyndns.org'
       option 'username' 'hipocrates'
       option 'password' 'victorAlem'
       option 'force_interval' '13'
```

Verificación

Para verificar que hemos configurado correctamente nuestro servicio de DDNS primero demos tiempo al script de configuración que mande la información necesaria al servidor de nombre. Luego de esto vayamos a la página de administración de OpenWrt y anotemos la dirección IP que tiene la interfaz que está conectada a Internet. Nos dirigimos al menú Status -> Interfaces y nos aparecerá una imagen similar a esta:

Vemos que la interfaz “wan”, que en mi caso es la que está conectada a Internet, tiene la dirección IP 186.48.25.181.

Ahora intentemos verificar si nuestro dominio tiene esa dirección, para ello usamos el comando host desde un terminal de comandos de la siguiente forma:

```
victor@victor-VBox:~$ host elaguara.dyndns.org
elaguara.dyndns.org has address 186.48.25.181
```

Ipsec

La estructura de directorios de IPsec es la siguiente. En el directorio /etc se ubican dos archivos ipsec.conf y ipsec.secrets. En ipsec.secrets va la información de donde está la llave privada y el passphrase para la clave. La sintaxis es la siguiente:

```
: RSA /etc/ipsec.d/private/clave.pem “passphrase”
```

En el directorio /etc/ipsec.d/ se encuentran otros directorios en los cuales se almacenan las claves y certificados, estos directorios son:

```
root@Victor-Openswan:~# ls -l /etc/ipsec.d/
drwxr-xr-x    2 root     root         1024 Nov 20  2010 aacerts
drwxr-xr-x    2 root     root         1024 Nov 20  2010 cacerts
drwxr-xr-x    2 root     root         1024 Aug 17 20:07 certs
drwxr-xr-x    2 root     root         1024 Nov 20  2010 crls
drwxr-xr-x    2 root     root         1024 Aug 17 19:22 examples
drwxr-xr-x    2 root     root         1024 Nov 20  2010 ocspcerts
drwxr-xr-x    2 root     root         1024 Aug 17 19:22 policies
drwx------    2 root     root         1024 Aug 18 21:35 private
```

En certs guarda las llaves públicas de los clientes de la VPN. En cacerts se encuentra la llave pública de la Autoridad Certificadora (CA). Y en private se encuentran las claves privadas. Los restantes directorios no los utilizamos.

Instalación de OpenSwan

```
$ opkg install openswan kmod-openswan ntpclient
```

Generación de claves

En el momento de instalarse, openswan, ya genera los certificados y las claves[1]. Para chequear la creación de la clave, lanzamos el comando:

```
$ ipsec showhostkey --left
```

Si quisiéramos generarlas manualmente es a través del comando:

```
$ ipsec newhostkey
```

1 En los routers es posible generar las claves pero en las máquinas virtuales no nos fue posible.

Configuraciones

La configuración (/etc/ipsec.conf) es la misma para las dos puntas, pero las llaves son generadas en cada extremo.

Archivo /etc/ipsec.conf :

```
conn net-to-net
      left=190.64.72.224
      leftsubnet=192.168.1.0/24
      leftid=@pablo.home.com
      leftrsasigkey=0sAQNsOHqr6rWmV8atQHFp...
      leftnexthop=%defaultroute
      right=164.73.234.126
      rightsubnet=10.5.1.0/24
      rightid=@cure.edu.uy
      rightrsasigkey=0sAQN/DBQw27P25yCQXyR6...
      rightnexthop=%defaultroute
      auto=add
```

Verificación

Reiniciamos ipsec con el comando:

```
$ ipsec setup restart
```

La salida de el comando anterior es:

```
root@Victor-Openswan:~# ipsec setup restart
ipsec_setup: Stopping Openswan IPsec...
ipsec_setup: stop ordered, but IPsec appears to be already stopped!
ipsec_setup: doing cleanup anyway...
ipsec_setup: Starting Openswan IPsec U2.6.29/K2.6.32.25...
ipsec_setup: rm: invalid option -- d
ipsec_setup: BusyBox v1.15.3 (2010-11-12 04:24:17 PST) multi-call binary
ipsec_setup: Usage: rm [OPTIONS] FILE...
ipsec_setup: Remove (unlink) the FILE(s). Use '--' to
ipsec_setup: indicate that all following arguments are non-options.
ipsec_setup: Options:
ipsec_setup: -i     Always prompt before removing
ipsec_setup: -f     Never prompt
ipsec_setup: -r,-R  Remove directories recursively
```

luego iniciamos la conexión con el comando:

```
$ ipsec auto --up net-to-net
```

Cuando el otro extremo se conecta, con los mismos comandos, vemos la siguiente salida:

```
root@HomeRocha:/etc# ipsec auto --up net-to-net
104 "net-to-net" #1: STATE_MAIN_I1: initiate
003 "net-to-net" #1: received Vendor ID payload [Openswan (this version) 2.6.29 ]
003 "net-to-net" #1: received Vendor ID payload [Dead Peer Detection]
106 "net-to-net" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "net-to-net" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "net-to-net" #1: received Vendor ID payload [CAN-IKEv2]
004 "net-to-net" #1: STATE_MAIN_I4: ISAKMP SA established {auth=OAKLEY_RSA_SIG
cipher=aes_128 prf=oakley_sha group=modp2048}
117 "net-to-net" #2: STATE_QUICK_I1: initiate
003 "net-to-net" #2: spdadd-client command exited with status 2
032 "net-to-net" #2: STATE_QUICK_I1: internal error
003 "net-to-net" #2: requested algorithm is not available in the kernel
032 "net-to-net" #2: STATE_QUICK_I1: internal error
003 "net-to-net" #2: requested algorithm is not available in the kernel
032 "net-to-net" #2: STATE_QUICK_I1: internal error
003 "net-to-net" #2: requested algorithm is not available in the kernel
032 "net-to-net" #2: STATE_QUICK_I1: internal error
```

Buscando información encontramos que es un error de kernel.

Implementación de IPsec en notebooks

Paquetes necesarios

```
$ apt-get install openswan
```

Configuración

```
$ dpkg-reconfigure openswan
```

En la reconfiguración crear un nuevo certificado autofirmado

Longitud de la clave RSA a crear: 2048

Código de país: UY

Estado: Rocha

Localidad:

Nombre de la organización: CURE

Unidad Organizacional:

Nombre común: nombre

Dirección de correo: correo@dominio.com

ejecutar:

```
$ ipsec verify
```

La salida de este comando recomienda que cambiamos los send_redirect y los accept_redirect por “0” en los directorios /proc/sys/net/ipv4/*/

editamos /etc/ipsec.conf

```
conn net-to-net
       left=192.168.3.5
       leftid=@xy.example.com
       leftrsasigkey=0sAwEAAcvcd0RL44ZI45imvqi...
       leftnexthop=%defaultroute
       right=192.168.2.11
       rightid=@ab.example.com
       rightrsasigkey=0sAwEAAct/dCxawz3bxfLzZhm...
       rightnexthop=%defaultroute
       auto=add
```

Tal como dice en la documentación de openswan[2].

2 Sacar los comentarios luego de las evaluaciones porque no funciona.

Despues ejecutamos:

```
$ ipsec setup restart
```

y nos da lo siguiente:

```
ipsec_setup: NETKEY support found. Use protostack=netkey in /etc/ipsec.conf to avoid
attempts to use KLIPS. Attempting to continue with NETKEY
```

para solucionar esto agregamos en /etc/ipsec.conf las siguientes líneas al principio:

```
config setup
       protostack=netkey
```

Quedando el archivo /etc/ipsec.conf de la siguiente forma:

```
config setup
       protostack=netkey

conn net-to-net
       left=192.168.3.5
       leftid=@xy.example.com
       leftrsasigkey=0sAwEAAcvcd0RL44ZI4...
       leftnexthop=%defaultroute
       right=192.168.2.11
       rightid=@ab.example.com
       rightrsasigkey=0sAwEAAct/dCxawz3bx...
       rightnexthop=%defaultroute
       auto=add
```

ejecutamos un ping y verificamos que los datos estén encriptados:

```
14:59:02.001920 IP victor-EasyNote-LJ65.lan.homerocha.com.uy.42213 >
sn1msg3020316.sn1.gateway.edge.messenger.live.com.msnp: Flags [P.], seq
2047590495:2047590500, ack 707591661, win 63855, options [nop,nop,TS val 476430 ecr
63810557], length 5
14:59:02.237521 IP fenix.lan.homerocha.com.uy > victor-EasyNote-
LJ65.lan.homerocha.com.uy: ESP(spi=0x196fddfc,seq=0x5), length 132
14:59:02.237618 IP fenix.lan.homerocha.com.uy > victor-EasyNote-
LJ65.lan.homerocha.com.uy: ICMP echo request, id 10768, seq 5, length 64
```

vemos el "ESP" que es la cabecera de IPsec

Esta configuración las pasamos a un router y a una máquina virtual con openwrt y no anduvo. Nos mostró el siguiente error:

```
104 "net-to-net" #1: STATE_MAIN_I1: initiate
003 "net-to-net" #1: ignoring unknown Vendor ID payload [4f45517b4f7f6e657a7b4351]
003 "net-to-net" #1: received Vendor ID payload [Dead Peer Detection]
106 "net-to-net" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "net-to-net" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "net-to-net" #1: received Vendor ID payload [CAN-IKEv2]
004 "net-to-net" #1: STATE_MAIN_I4: ISAKMP SA established {auth=OAKLEY_RSA_SIG
cipher=aes_128 prf=oakley_sha group=modp2048}
117 "net-to-net" #2: STATE_QUICK_I1: initiate
003 "net-to-net" #2: ERROR: netlink response for Add SA esp.da92a71c@190.132.166.99
included errno 2: No such file or directory
032 "net-to-net" #2: STATE_QUICK_I1: internal error
003 "net-to-net" #2: ERROR: netlink response for Add SA esp.da92a71c@190.132.166.99
included errno 2: No such file or directory
032 "net-to-net" #2: STATE_QUICK_I1: internal error
003 "net-to-net" #2: ERROR: netlink response for Add SA esp.da92a71c@190.132.166.99
included errno 2: No such file or directory
032 "net-to-net" #2: STATE_QUICK_I1: internal error
003 "net-to-net" #2: ERROR: netlink response for Add SA esp.da92a71c@190.132.166.99
included errno 2: No such file or directory
032 "net-to-net" #2: STATE_QUICK_I1: internal error
```

Según lo indagado en Internet y las recomendaciones de uno de los docentes (http://www.digitalenginesoftware.com/blog/archives/67-Troubleshooting-OpenSwan-with-NETKEY.html) es un problema del kernel de openwrt para el cual no hemos encontrado solución.

OpenVPN

Instalación de OpenVPN

```
$ opkg install openvpn ntpclient
```
[3]

Generación de claves

Las claves y certificados se generaron en una pc, usando la herramienta *easy-rsa*, que se instala al instalar OpenVPN.

Dentro del directorio:

`/usr/share/doc/openvpn/examples/easy-rsa/2.0/`

encontraremos las herramientas necesarias para generar las llaves y certificados para nuestro servidor y los clientes.

El primer paso para configurar OpenVPN es establecer la PKI (public key infrastructure). PKI consiste en:

- certificados separados (conocidos como llaves públicas) y llave privada para el servidor y cada cliente.
- Un Certificado de Autoridad maestro (CA), certificado y llave que se utilizará para firmar cada uno de los certificados de servidor y clientes.

Los siguientes comandos se realizarán estando dentro del directorio:

`/usr/share/doc/openvpn/examples/easy-rsa/2.0/`

1- Inicializar PKI:

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

3 Se instala ntpclient para mantener actualizada la hora del router, ya que si tras un corte de energía, el router levantara con la fecha incorrecta daría un error con los certificados.

Al final del comando build-ca, se creará el CA, y se pedirá los siguientes datos[4]:

```
Country Name (2 letter code) [UY]:
State or Province Name (full name) [RO]:
Locality Name (eg, city) [Rocha]:
Organization Name (eg, company) [CURE]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:OpenVPN
Email Address [me@myhost.mydomain]:
```

2-Generar llave y certificado para el servidor:

```
$ ./build-key-server homerocha
```

En este paso, el valor de los parámetros serán los mismo que los ingresados cuando se inicializó el PKI, a excepción del `Common Name` que ahí se debe ingresar el mismo nombre que se pasó por parámetro al comando, en este caso “homerocha”. Las siguientes 2 preguntas se le deben de contestar afirmativamente.

3-Generar las llaves y certificados para los clientes:

```
$ ./build-key el_aguara
$ ./build-key marcosro
```

En este paso, el valor de los parámetros serán los mismo que los ingresados cuando se inicializó el PKI, a excepción del `Common Name` que ahí se debe ingresar el mismo nombre que se pasó por parámetro al comando, en este caso “el_aguara” y “marcosro”. Las siguientes 2 preguntas se le deben de contestar afirmativamente.

4 los datos ingresados para el CA, deberán respetarse al generar las llaves del servidor y los clientes, a excepción del “`Common Name`” que será el nombre del dueño de la llave.

4-Generar Diffie Hellman

```
$ ./build-dh
```

En el subdirectorio keys se guardarán todos los archivos generados por los comandos anteriores, hay archivos que no los usaremos y otros que solo los tendrá que tener el servidor. Se detallan en la siguiente tabla[5]:

Filename	Needed By	Purpose	Secret
ca.crt	server + all clients	Root CA certificate	NO
ca.key	key signing machine only	Root CA key	YES
dh{n}.pem	server only	Diffie Hellman parameters	NO
server.crt	server only	Server Certificate	NO
server.key	server only	Server Key	YES
client1.crt	client1 only	Client1 Certificate	NO
client1.key	client1 only	Client1 Key	YES
client2.crt	client2 only	Client2 Certificate	NO
client2.key	client2 only	Client2 Key	YES
client3.crt	client3 only	Client3 Certificate	NO
client3.key	client3 only	Client3 Key	YES

Por lo cual las llaves y certificados de distribuiran de la siguiente manera:

Servidor homerocha:

- ca.crt
- ca.key
- dh1024.pem
- homerocha.crt
- homerocha.key

cliente el_aguara:

- ca.crt
- el_aguara.crt
- el_aguara.key

Cliente marcosro:

- ca.crt

5 tabla obtenida del HowTo de OpenVPN

- marcosro.crt
- marcosro.key

Configuraciones

Servidor

La configuración del servidor OpenVPN se guardará como:

`/etc/openvpn/openvpn.conf`

y tendrá el siguiente contenido:

```
port 1194
# Protocolo
proto udp
# Dispositivo
dev tun

# Certificados
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/homerocha.crt
key /etc/openvpn/keys/homerocha.key
dh /etc/openvpn/keys/dh1024.pem

# IP de la VPN
server 10.8.0.0 255.255.255.0

# Compresion habilitada
comp-lzo

# Mantener la conexion viva
keepalive 10 120

# log de sucesos
status /tmp/openvpn.status

# log de entrega de ip's
ifconfig-pool-persist /tmp/openvpn.ipp.txt

# en este directorio se encuentran archivos
# con las redes internas de cada cliente
client-config-dir /etc/openvpn/ccd
# ruta local para llegar a red cliente 10.5.0.0
route 10.5.0.0 255.255.0.0
# ruta local para llegar a red cliente 172.16.0.0
route 172.16.0.0 255.255.0.0

# Modo
client-to-client
# envio de rutas a los clientes para que lleguen a los demás clientes
push route 10.5.0.0 255.255.0.0
push route 172.16.0.0 255.255.0.0

# envio de ruta a clientes para que lleguen a red del servidor
push route 192.168.0.0 255.255.0.0
# Gestion por telnet
management 127.0.0.1 1194
```

Al arrancar con la configuración de OpenVPN, elegimos trabajar con el protocolo UDP en vez del TCP. Esto se debe a que, aunque TCP es más confiable, provoca una sobrecarga en la conexión debido a los continuos reconocimientos de paquetes (ACK). En cambio con UDP no tenemos esa sobrecarga obteniendo un mayor rendimiento en la VPN. Cabe destacar, que si algún paquete UDP de la VPN se pierde, el TCP encapsulado dentro del UDP se encargará del reenvío de los datos.

Con el archivo anterior inyectaremos en la tabla de rutas de los clientes (entre otras configuraciones) las vías para alcanzarse entre ellos, pero no queremos inyectarle (nuevamente) la ruta hacia la red del cliente, para eso hemos agregado la siguiente línea en el archivo de configuración del servidor: `client-config-dir /etc/openvpn/ccd` , en este directorio se encuentran archivos que contienen información de las redes internas de cada cliente con la siguiente sintaxis: `iroute 192.168.4.0 255.255.255.0.` Luego, para que sea alcanzable la red de un cliente por otro cliente, son necesarias las siguientes líneas:

```
client-to-client # esta es la línea para habilitar el dialogo entre clientes
push route 10.5.0.0 255.255.0.0 # para inyectar rutas a los clientes
push route 172.16.0.0 255.255.0.0 # para inyectar rutas a los clientes
push route 192.168.0.0 255.255.0.0 # para inyectar rutas a los clientes
```

En el firewall hay que agregar la regla de que acepte por la interfaz que da acceso a Internet y el puerto 1194 las conexiones entrantes. Por ejemplo en el archivo /etc/config/firewall agregar:

```
config 'rule'
       option 'src' 'wan'
       option 'target' 'ACCEPT'
       option 'proto' 'udp'
       option 'dest_port' '1194'
       option '_name' 'openvpn'
```

Cliente

```
client
dev tun
proto udp
remote homerocha.dyndns.org 1194
nobind
ca /etc/openvpn/ca.crt
cert /etc/openvpn/el_aguara.crt
key /etc/openvpn/el_aguara.key
dh /etc/openvpn/dh.pem
comp-lzo
```

Configuración firewall

Tanto en el servidor, como en los clientes hay que configurar el firewall para que, en las interfaces que queremos que interactuen con la vpn permitan forwarding.

Por ejemplo, en el archivo /etc/config/firewall:

```
config 'zone'
       option 'name' 'lan'
       option 'input' 'ACCEPT'
       option 'output' 'ACCEPT'
       option 'forward' 'ACCEPT'
```

Además hay que habilitar el forward en el default para que interactuar con el tunel:

```
config 'defaults'
       option 'syn_flood' '1'
       option 'input' 'ACCEPT'
       option 'output' 'ACCEPT'
       option 'forward' 'ACCEPT'
```

Verificación

Se cheque, tanto en el servidor como en los clientes (después de lanzar OpenVPN) que aparezca la interfaz de OpenVPN que es la tun0:

```
root@zeus:~# ifconfig tun0
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.1  P-t-P:10.8.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:105107 errors:0 dropped:0 overruns:0 frame:0
          TX packets:76377 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:126436199 (120.5 MiB)  TX bytes:7027339 (6.7 MiB)
```

Luego, chequeamos las rutas que se crearon, tanto en el servidor como en los clientes:

Rutas en servidor :

```
root@zeus:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
200.40.22.2     *               255.255.255.255 UH    0      0        0 pppoe-wan
10.8.0.2        *               255.255.255.255 UH    0      0        0 tun0
192.168.3.0     *               255.255.255.0   U     0      0        0 br-wlan
192.168.2.0     *               255.255.255.0   U     0      0        0 eth0.3
10.8.0.0        10.8.0.2        255.255.255.0   UG    0      0        0 tun0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0.2
10.5.0.0        10.8.0.2        255.255.0.0     UG    0      0        0 tun0
172.16.0.0      10.8.0.2        255.255.0.0     UG    0      0        0 tun0
default         uni1bras1.antel 0.0.0.0         UG    0      0        0 pppoe-wan
```

Rutas en cliente :

```
root@OpenWrt:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
200.40.22.8     *               255.255.255.255 UH    0      0        0 ppp0
10.8.0.9        *               255.255.255.255 UH    0      0        0 tun0
10.8.0.0        10.8.0.9        255.255.255.0   UG    0      0        0 tun0
10.5.1.0        *               255.255.255.0   U     0      0        0 eth0.0
10.5.2.0        *               255.255.255.0   U     0      0        0 wl0
172.16.0.0      10.8.0.9        255.255.0.0     UG    0      0        0 tun0
192.168.0.0     10.8.0.9        255.255.0.0     UG    0      0        0 tun0
default         puuni1bras1.ant 0.0.0.0         UG    0      0        0 ppp0
```

Luego, la prueba final es ver el tráfico en la tun0 con un tcpdump:

```
root@zeus:~# tcpdump -i tun0 -vn
tcpdump: listening on tun0, link-type RAW (Raw IP), capture size 65535 bytes
15:44:10.086949 IP (tos 0x0, ttl 63, id 28923, offset 0, flags [DF], proto TCP (6), length 60)
    10.5.2.100.39736 > 192.168.1.1.10050: Flags [S], cksum 0x80a1 (correct), seq 2706192241, win
5840, options [mss 1368,sackOK,TS val 1525642 ecr 0,nop,wscale 5], length 0
15:44:10.087148 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.1.10050 > 10.5.2.100.39736: Flags [S.], cksum 0xac9d (correct), seq 3777650978, ack
2706192242, win 5840, options [mss 1460,nop,nop,sackOK,nop,wscale 1], length 0
15:44:10.380865 IP (tos 0x0, ttl 63, id 28924, offset 0, flags [DF], proto TCP (6), length 40)
    10.5.2.100.39736 > 192.168.1.1.10050: Flags [.], cksum 0x0383 (correct), ack 1, win 183, length
0
15:44:10.381445 IP (tos 0x0, ttl 63, id 28925, offset 0, flags [DF], proto TCP (6), length 51)
    10.5.2.100.39736 > 192.168.1.1.10050: Flags [P.], cksum 0xdf9a (correct), seq 1:12, ack 1, win
183, length 11
15:44:10.381560 IP (tos 0x0, ttl 64, id 30577, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.1.1.10050 > 10.5.2.100.39736: Flags [.], cksum 0xf8c6 (correct), ack 12, win 2920,
length 0
15:44:10.383213 IP (tos 0x0, ttl 64, id 30578, offset 0, flags [DF], proto TCP (6), length 45)
    192.168.1.1.10050 > 10.5.2.100.39736: Flags [P.], cksum 0x4533 (correct), seq 1:6, ack 12, win
2920, length 5
```

De esta forma nos aseguramos que el trafico está siendo enrutado a través de la vpn.

Agregamos un script en el arranque de OpenWRT para que cada vez que se reinicie o incluso se apague el router, cuando este arranca nuevamente, se inicia la VPN. Creamos el siguiente script:

```
root@zeus:~# cat /etc/init.d/vpn
#!/bin/sh /etc/rc.common
# Copyright (C) 2008 OpenWrt.org
START=99

start() {
  openvpn /etc/openvpn/openvpn.conf &
  echo 0
}

stop() {
  killall openvpn
  echo 0
}

restart() {
       stop
       sleep 1
       start
}
```

Luego, hay que crear un link simbólico en el directorio /etc/rc.d apuntando a este script:

```
root@zeus:~# ls -l /etc/rc.d/
lrwxrwxrwx    1 root     root           18 Jul 14 21:41 K50dropbear -> ../init.d/dropbear
lrwxrwxrwx    1 root     root           17 Jul 14 21:41 K90network -> ../init.d/network
lrwxrwxrwx    1 root     root           18 Jul 14 21:41 K95luci_bwc -> ../init.d/luci_bwc
lrwxrwxrwx    1 root     root           22 Jul 14 21:41 K95luci_fixtime -> ../init.d/luci_fixtime
lrwxrwxrwx    1 root     root           14 Jul 14 21:41 K98boot -> ../init.d/boot
lrwxrwxrwx    1 root     root           16 Jul 14 21:41 K99umount -> ../init.d/umount
lrwxrwxrwx    1 root     root           19 Jul 14 21:41 S05defconfig -> ../init.d/defconfig
lrwxrwxrwx    1 root     root           22 Jul 14 21:41 S05luci_fixtime -> ../init.d/luci_fixtime
lrwxrwxrwx    1 root     root           14 Jul 14 21:41 S10boot -> ../init.d/boot
lrwxrwxrwx    1 root     root           13 Jul 14 21:41 S39usb -> ../init.d/usb
lrwxrwxrwx    1 root     root           17 Jul 14 21:41 S40network -> ../init.d/network
lrwxrwxrwx    1 root     root           18 Jul 14 21:41 S45firewall -> ../init.d/firewall
lrwxrwxrwx    1 root     root           14 Jul 14 21:41 S50cron -> ../init.d/cron
lrwxrwxrwx    1 root     root           18 Jul 14 21:41 S50dropbear -> ../init.d/dropbear
lrwxrwxrwx    1 root     root           16 Jul 14 21:41 S50uhttpd -> ../init.d/uhttpd
lrwxrwxrwx    1 root     root           27 Jul 14 21:41 S59luci_dhcp_migrate ->
../init.d/luci_dhcp_migrate
lrwxrwxrwx    1 root     root           17 Jul 14 21:41 S60dnsmasq -> ../init.d/dnsmasq
lrwxrwxrwx    1 root     root           23 Aug 27 21:24 S60zabbix_agentd -> ../init.d/zabbix_agentd
lrwxrwxrwx    1 root     root           14 Jul 14 21:41 S95done -> ../init.d/done
lrwxrwxrwx    1 root     root           18 Jul 14 21:41 S95luci_bwc -> ../init.d/luci_bwc
lrwxrwxrwx    1 root     root           13 Aug 26 18:41 S96led -> ../init.d/led
lrwxrwxrwx    1 root     root           18 Jul 14 21:41 S97watchdog -> ../init.d/watchdog
lrwxrwxrwx    1 root     root           15 Sep  7 19:28 S99openvpn -> /etc/init.d/vpn
lrwxrwxrwx    1 root     root           16 Jul 14 21:41 S99sysctl -> ../init.d/sysctl
```

Observamos aquí que también hemos creado otro enlace simbólico apuntando al script /etc/init.d/zabbix_agentd para que cuando arranca el router arranque también el agente Zabbix.

La letra “S” quiere decir que lo que hay que pasarle al script es un “start” y el número a continuación es el orden en que se ejecutan estos scripts. Hay que aclarar que no es en cualquier orden que debemos iniciar la VPN, básicamente debe ser después de tener levantada la red (“network”) y el “firewall”.

Zabbix

Descargamos una máquina virtual Zabbix pre-instalada desde el sitio http://www.zabbix.com/download.php y la instalamos en una máquina dentro de la red de uno de los clientes. Hay que configurar el servidor web Apache para que acepte conexiones desde la/las redes que nos interesa ingresar a Zabbix. En nuestro caso configuraremos Apache para poder configurar Zabbix desde todas las redes de la VPN.

Monitorizaremos lo siguiente:

1. monitorización de BW entrante y saliente de LAN / WAN de cada router
2. monitorización de filesystems de cada router

Instalación

Luego, en cada router a monitorizar instalamos el agente de Zabbix con el siguiente comando:

```
$ opkg install zabbix-agent
```

Configuración

Después editamos el archivo de configuración /etc/zabbix/zabbix_agent las siguientes lineas:

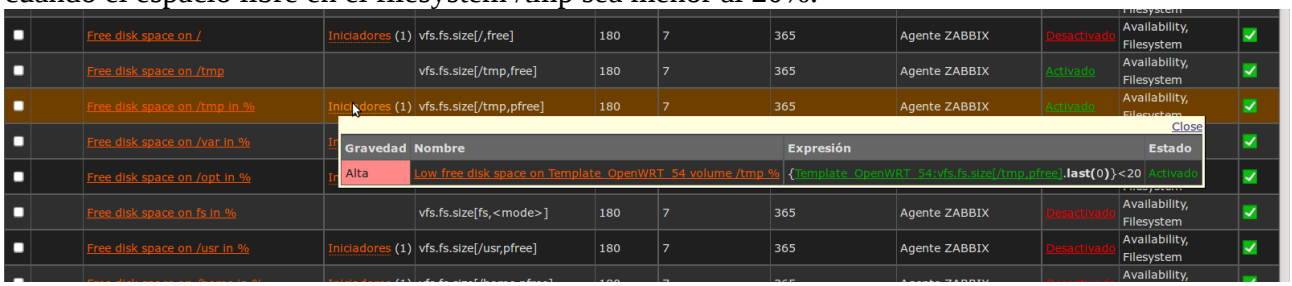
```
Server=10.5.1.240

Hostname=ElAguara
```

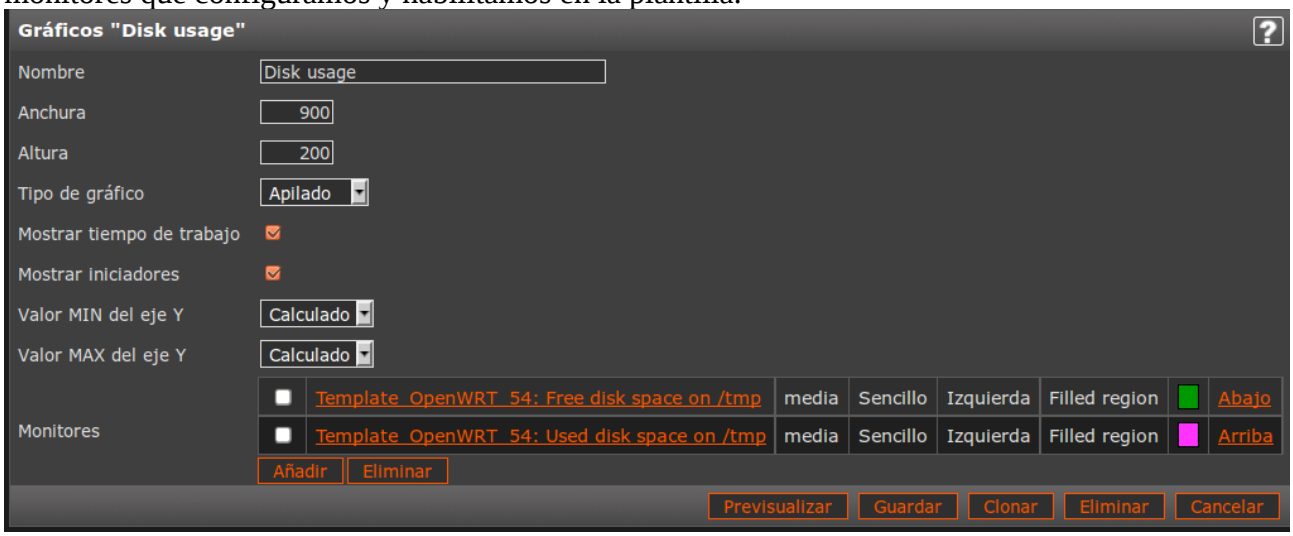
Como vemos, lo único que hay que configurar es la dirección IP del servidor Zabbix y el nombre del router.

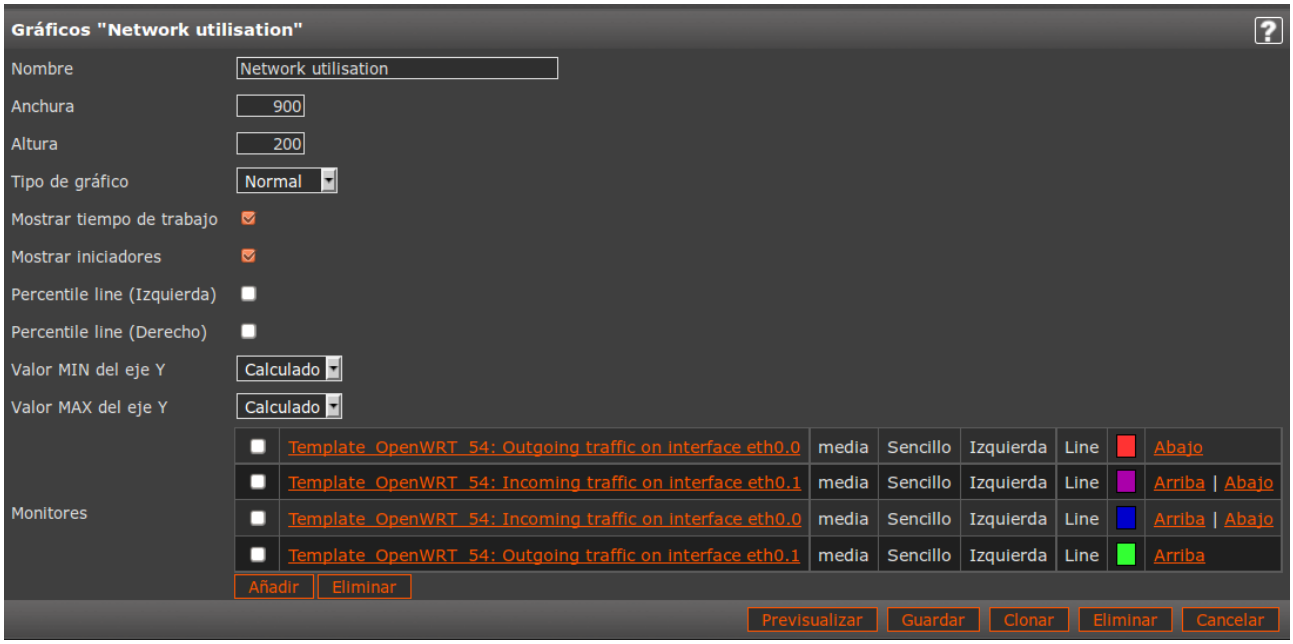
El servidor Zabbix presenta una interfaz web para realizar cambios administrativos. Haremos todas las configuraciones necesarias a través de este “front end”.
Como temenos dos tipos de routers creamos dos plantillas distintas. Para esto vamos a Configuración --> Plantillas y seleccionamos el “Template_Linux” y lo clonamos (botón “clonar”) y le cambiamos el nombre por “Template_OpenWRT_54” y “Template_OpenWRT_tp-link”.
Luego de esto, nos dirigimos a la sección “Monitores” del template y desabilitamos todos los monitores y creamos los que nos interesan. Para esto clonamos el Monitor “Incoming trafic on interface eth0” y generamos un nuevo monitor para cada interfaz del router. Realizamos lo mismo para el monitor “Outgoing trafic on interface eth0”.

El próximo paso es habilitar el “Iniciador” y el “Monitor” para que Zabbix dispare una alarma cuando el espacio libre en el filesystem /tmp sea menor al 20%.



Ahora configuremos los gráficos. Vamos a la parte de Configuración > Plantillas > Gráficos (de las plantillas creadas anteriormente). Tendremos dos gráficos, uno de la utilización de la red y otra sobre el uso del disco. Editamos las gráficas eliminando los monitores que tiene y añadiendo los monitores que configuramos y habilitamos en la plantilla.





Modificación de OpenWRT

Crearemos tres módulos, dos para desconectar a los clientes y uno para ver el status de OpenVPN. Con esto se agregan links al front end web LuCi que ejecutan scripts y despliegan la salida en texto plano. Estos módulos se crean en:

```
/usr/lib/lua/luci/controller/
```

Creación de los Módulos

Módulo de desconexión de clientes:

```
module("luci.controller.vpn-desco-el_aguara", package.seeall)

function index()
     entry( {"admin", "services", "vpn-desco-el_aguara"}, call("action_tryme"), "VPN
Desconexion El aguara").dependent=false
end

function action_tryme()
   luci.http.prepare_content("text/plain")
   luci.http.write(luci.sys.exec("sh /etc/openvpn/scripts/desconec-el_aguara.sh"))
end
```

Módulo de status de OpenVPN:

```
module("luci.controller.vpn-status", package.seeall)

function index()
     entry( {"admin", "services", "vpn-desco-status"}, call("action_tryme"), "VPN
Estado").dependent=false
end

function action_tryme()
   luci.http.prepare_content("text/plain")
   luci.http.write(luci.sys.exec("sh /etc/openvpn/scripts/status.sh"))
end
```

La función “action_tryme”, para desconectar un cliente ejecuta el siguiente script:

```
root@zeus:/etc/openvpn/scripts# cat desconec-el_aguara.sh
#!  /bin/bash
( sleep 1
echo kill el_aguara
sleep 1
echo quit ) | telnet 127.0.0.1 1195
```

La función “action_tryme”, para mostrar el status de la VPN ejecuta el siguiente shell script:

```
root@zeus:/etc/openvpn/scripts# cat status.sh
#!  /bin/bash
( sleep 1
echo status
sleep 1
echo quit ) | telnet 127.0.0.1 1195
```

Estos scripts se escriben en texto plano y simplemente los copiamos dentro del directorio /etc/openvpn/scripts y luego le cambiamos los permisos para que puedan ser ejecutados.