OpenVPN & Roadwarrior

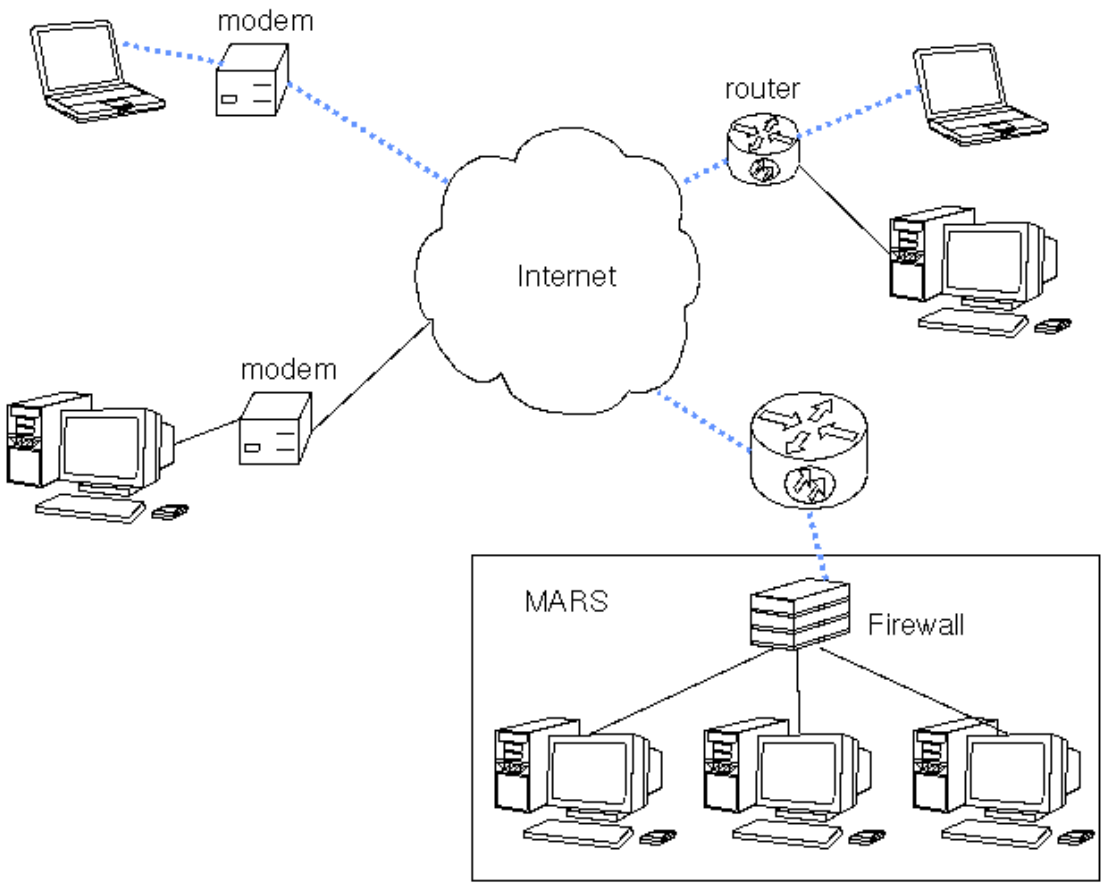
"Installazione di Openvpn su server Freebsd e configurazione dei clients"

Cos'è OpenVPN –

Openvpn è una soluzione tutto sommato semplice e funzionale per collegare 2 distinte LAN in ambito geografico.
Si basa su SSL quindi algoritmi di crittografia a 1024 o 2048 bit, garantendo un alto grado di sicurezza.

Topologia -

I client della rete MARS sono raggiungibili da internet attraverso il servizio OpenVPN installato sul firewall.
I roadwarrior si connettono ad internet tramite modem o router e possono raggiungere i client presenti nella rete MARS grazie alla VPN.



Fonte: http://gentoo-wiki.com/HOWTO_OpenVPN_RoadWarrior

Requisiti –

Uno dei requisiti fondamentali per far funzionare la vpn, è il la presenza del device `tun`.
Per verificare che sia presente nel proprio sistema si procede in questo modo:

```
ls /dev | grep tun0
tun0
```

Se non restituisce `tun0` significa che il device non è presente sul server ed è necessario ricompilare il kernel.

Ricompilazione kernel –

Per ricompilare il kernel, assicurarsi di avere i sorgenti in `/usr/src/sys`.
Se non sono presenti, esisto almeno 3 modi per avere i sorgenti:

- Eseguire `sysinstall` (`/stand/sysinstall` su FreeBSD di versione precedente alla 5.2) come `root`, scegliendo Configure, poi Distributions, poi src, poi sys.
- Dal cd "ufficiale" di FreeBSD.

```
mount /cdrom
mkdir -p /usr/src/sys
ln -s /usr/src/sys /sys
cat /cdrom/src/ssys.[a-d]* | tar -xzvf -
```

- Tramite cvsup, che è il metodo più sicuro per avere i sorgenti aggiornati.

```
cd /usr/ports/net/cvsup
make
make install
make clean
```

Ora che cvsup è installato, bisogna configurarlo ☺
Creiamo i seguenti files:

```
touch /usr/local/etc/cvsup/stable-supfile
touch /usr/local/etc/cvsup/ports-supfile
touch /usr/local/etc/cvsup/doc-supfile
touch /usr/sup/refuse
```

Il file `stable-supfile` conterrà:

```
*default host=cvsup.it.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_6_2
*default delete use-rel-suffix
src-all
```

Il file `ports-supfile` conterrà:

```
*default host=cvsup.it.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=*
*default delete use-rel-suffix
ports-all
```

Il file `doc-supfile` conterrà:

```
*default host=cvsup.it.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=*
*default delete use-rel-suffix
doc-all
```

Il file `refuse` conterrà:

```
doc/bn_*
doc/da_*
doc/de_*
doc/el_*
doc/es_*
doc/fr_*
doc/id_*
doc/it_*
doc/ja_*
doc/nl_*
doc/no_*
doc/pl_*
doc/pt_*
doc/ro_*
doc/ru_*
doc/sr_*
doc/tr_*
doc/zh_*
ports/arabic
ports/arabic
ports/chinese
ports/french
ports/german
ports/hebrew
```

```
ports/hungarian
ports/japanese
ports/korean
ports/polish
ports/portuguese
ports/russian
ports/ukrainian
ports/vietnamese
```

Bene, ora che abbiamo configurato `cvsup` aggiorniamo i sorgenti del sistema:

```
cvsup -g -L 2 stable-supfile
```

Ora che abbiamo i sorgenti del sistema copiamo la configurazione originale del kernel nella nostra customizzata.

```
cd /usr/src/sys/i386/conf
cp GENERIC MYKERNEL
```

Editiamo il file di config appena creato col nostro editor preferito e decommentiamo questa riga:

```
ident           MYKERNEL
device          tun             # Packet tunnel.
options IPFIREWALL #abilita il firewall nel kernel
options IPFIREWALL_VERBOSE #abilita il logging per il firewall,
solitamente in /var/log/security.
options IPFIREWALL_VERBOSE_LIMIT=200 #Limita il numero di logs in
/var/log/security
options IPDIVERT #abilita il divert socket, in pratica il NATD
options IPFIREWALL_DEFAULT_TO_ACCEPT #setta il firewall di base "Tutto
aperto"
#options ICMP_BANDLIM #abilita la protezione dai tipici attacchi
D.O.S.#(Denial Of Service).
options IPFIREWALL_FORWARD
```

Nota: *Ho dovuto abilitare queste funzioni (nat e firewall) perché il mio server openvpn è in dmz e utilizzo una rete diversa (192.168.254.0/24) dalla rete della DMZ (192.168.13.32/27).*

Salviamo il file e siamo pronti a compilare il nuovo kernel.

```
cd /usr/src
make buildkernel KERNCONF=MYKERNEL
make installkernel KERNCONF=MYKERNEL
```

Il nuovo kernel sarà copiato nella directory `/boot/kernel` come `/boot/kernel/kernel` e il kernel precedente sarà copiato in `/boot/kernel.old/kernel`. Ora, riavvia il sistema e riparti per usare il tuo nuovo kernel. Se qualcosa va storto...
http://www.freebsd.org/doc/it_IT.ISO8859-15/books/handbook/kernelconfig-building.html

SETUP componenti –

Installiamo OpenVPN:

```
cd /usr/ports/
make search name=openvpn
Port:   openvpn-2.0.6_7
Path:   /usr/ports/security/openvpn
Info:   Secure IP/Ethernet tunnel daemon
Maint:  matthias.andree@gmx.de
B-deps: lzo2-2.02_1
R-deps: lzo2-2.02_1
WWW:    http://openvpn.net/
make
make install
make clean
```

Creiamo le directory che conterranno i file di configurazione e le chiavi:

```
mkdir /usr/local/etc/openvpn
mkdir /usr/local/etc/openvpn/keys
```

Copiamo nella directory di openvpn la directory easy-rsa.

```
cp -R /usr/local/share/doc/openvpn/easy-rsa /usr/local/etc/openvpn/
```

La directory easy-rsa contiene i tools necessari per creare le chiavi sia del server che dei futuri clients.
Modificare nel file vars i parametri KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEYORG, e KEY_EMAIL con i propri valori.

```
cd /usr/local/etc/openvpn/easy-rsa
vi vars
sh
source ./vars
./clean-all
./build-ca
./build-key-server server
./build-key client1
./build-dh
openvpn --genkey --secret ta.key
exit
mv /usr/local/etc/openvpn/easy-rsa/keys/* /usr/local/etc/openvpn/keys/
cd keys
chmod 600 ca.key
chmod 600 server.key
chmod 600 ta.key
```

Configurazione del server –

Bene, configuriamo openvpn nel seguente modo:

```
local 192.168.13.36
port 1194
proto tcp # Meno sicuro ma necessario per me
;proto udp
dev tun
ca keys/ca.crt
cert keys/server.crt
key keys/server.key
dh keys/dh1024.pem
server 192.168.254.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway"
push "dhcp-option DNS 192.168.13.36"
keepalive 10 120
tls-auth keys/ta.key 0
cipher BF-CBC
comp-lzo
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
log-append  openvpn.log
verb 3
mute 20
```

Configurazione natd –

Come ho detto poco sopra, la mia necessità è di fare il nat degli ip della vpn in modo che al firewall si presenti solo l'ip della macchina in dmz, altrimenti sarebbe stato più complicato e avrei dovuto mettere altre regole sul firewall per far uscire la vpn verso internet.
Non escludo che un domani integrerò il documento con queste funzionalità.

```
dynamic yes #per indirizzi dinamici
use_sockets yes
same_ports yes
log yes
deny_incoming no
interface pcn0
unregistered_only yes
#redirect_address 10.15.254.6 192.168.14.34
#redirect_address 10.15.254.7 192.168.14.34
```

Configurazione firewall ipfw –

Di seguito invece il file config del firewall in cui viene richiamato il nat.
Questo firewall è molto banale, visto che esiste un firewall a monte della dmz.

```
cmd="/sbin/ipfw add"
skip="skipto 500"
pif=pcn0
#iif=hme1
ks="keep-state"
/sbin/ipfw -q  -f flush

$cmd 004 allow all from any to any via lo0
$cmd 100 divert natd ip from any to any via $pif
$cmd 101 check-state
$cmd 102 allow tcp from any to any established
$cmd 450 allow log ip from any to any
```

Avviamo il tutto –

Inseriamo nel fire /etc/rc.conf i parametri necessari per far partire il firewall, natd e openvpn al boot.

```
##################
# Firewall & NAT #
##################

firewall_enable="YES"
firewall_type="open"
firewall_logging="YES"
firewall_quite="NO"
firewall_script="/etc/rc.fire"
#firewall_flags=""
#ATTENZIONE:QUESTE RIGHE SOLO PER "CASO 3"
natd_enable="YES"
natd_interface="pcn0"
natd_flags="-f /etc/natd.conf"

############
# OpenVPN #
############

openvpn_enable="YES"
openvpn_configfile="/usr/local/etc/openvpn/server.conf"
```

Nota: ricordatevi di abilitare il forwarding, altrimenti la vpn avrà problemi ad uscire verso internet.

```
sysctl -a | grep forwarding
net.inet.ip.forwarding: 1
```

Per attivarlo:

```
sysctl -w net.inet.ip.forwarding=1
```

Procediamo col avviare openvpn, natd e ipfw.

```
sh /usr/local/etc/rc.d/openvpn start
sh /etc/rc.d/natd
sh /etc/rc.d/ipfw
```

Trasferire le chiavi ai clients -

Il metodo più sicuro per trasferire le chiavi ai clients è pgp, ma non lo spiegherò.

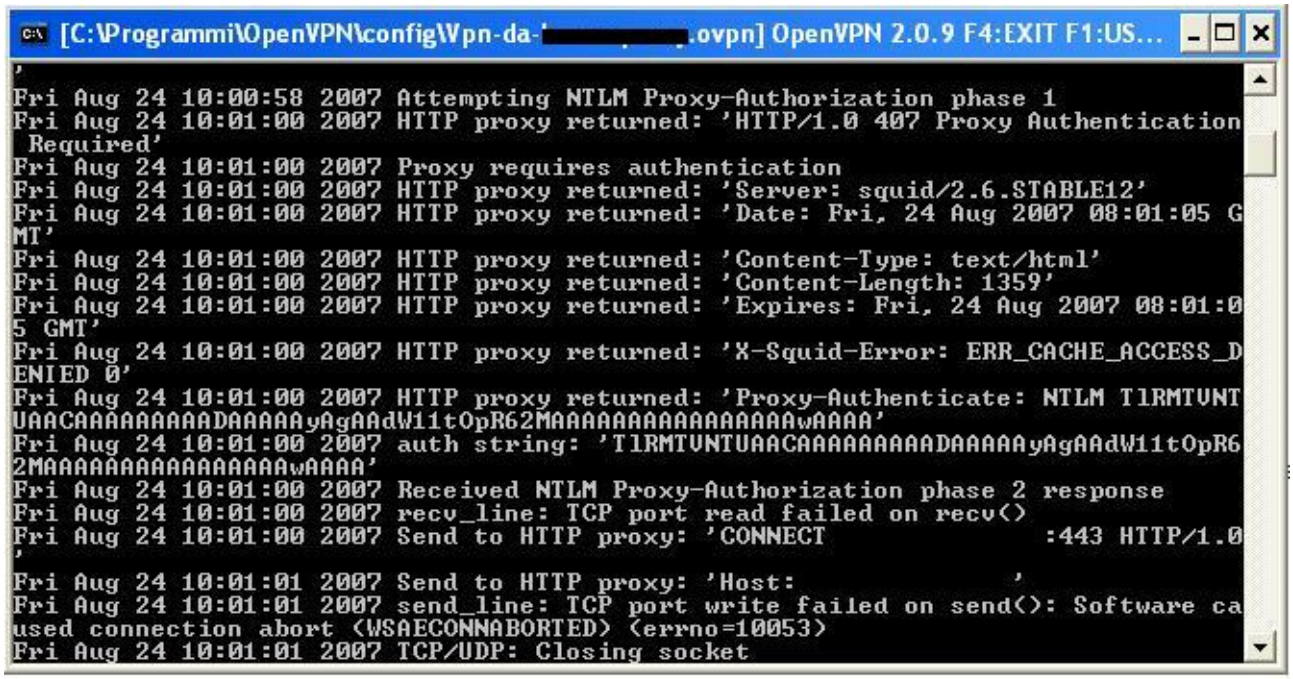
Configurazione del client windows –

Scarichiamo dal sito http://openvpn.net/download.html il pacchetto per windows e installiamolo.

Posizioniamoci in C:\Programmi\OpenVPN\config creiamo la directory keys ed infine editiamo il file default.ovpn e configuriamolo come segue:

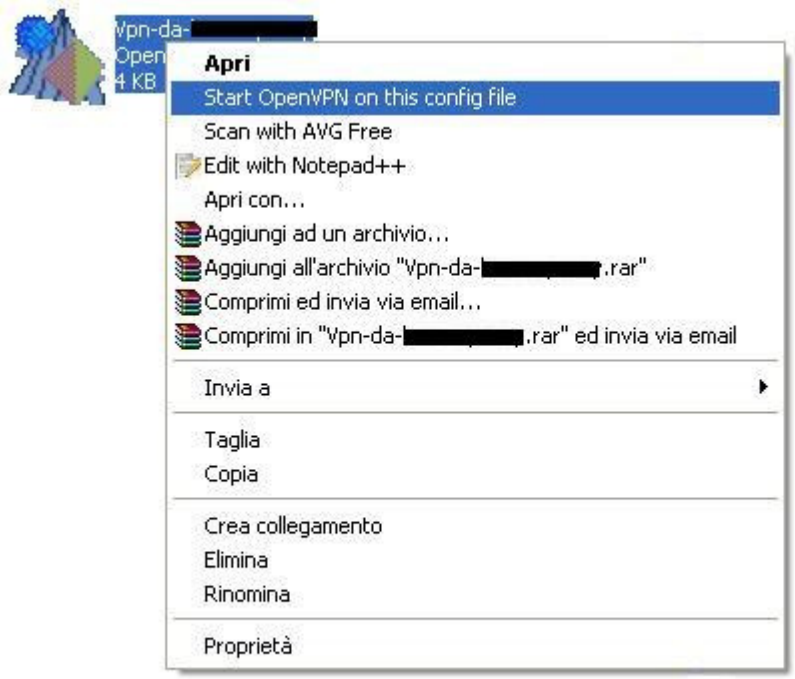
```
Client
dev tun
proto tcp
remote <ip_server> 443
resolv-retry infinite
nobind
persist-key
persist-tun
# Necessario per attraversare un proxy con autenticazione
http-proxy <ip_proxy> 3128
C:\\Programmi\\OpenVPN\\config\\autenticazione.txt ntlm
ca C:\\Programmi\\OpenVPN\\config\\keys\\ca.crt
cert C:\\Programmi\\OpenVPN\\config\\keys\\client.crt
key C:\\Programmi\\OpenVPN\\config\\keys\\client.key
tls-auth C:\\Programmi\\OpenVPN\\config\\keys\\ta.key 1
cipher BF-CBC
comp-lzo
verb 3
keepalive 10 60
ping-timer-rem
```

Il file autenticazione.txt contiene l'user e la pass, messi semplicemente su due righe diverse, per autenticarsi sul proxy.
Può capitare durante la fase di autenticazione della vpn un errore simile:

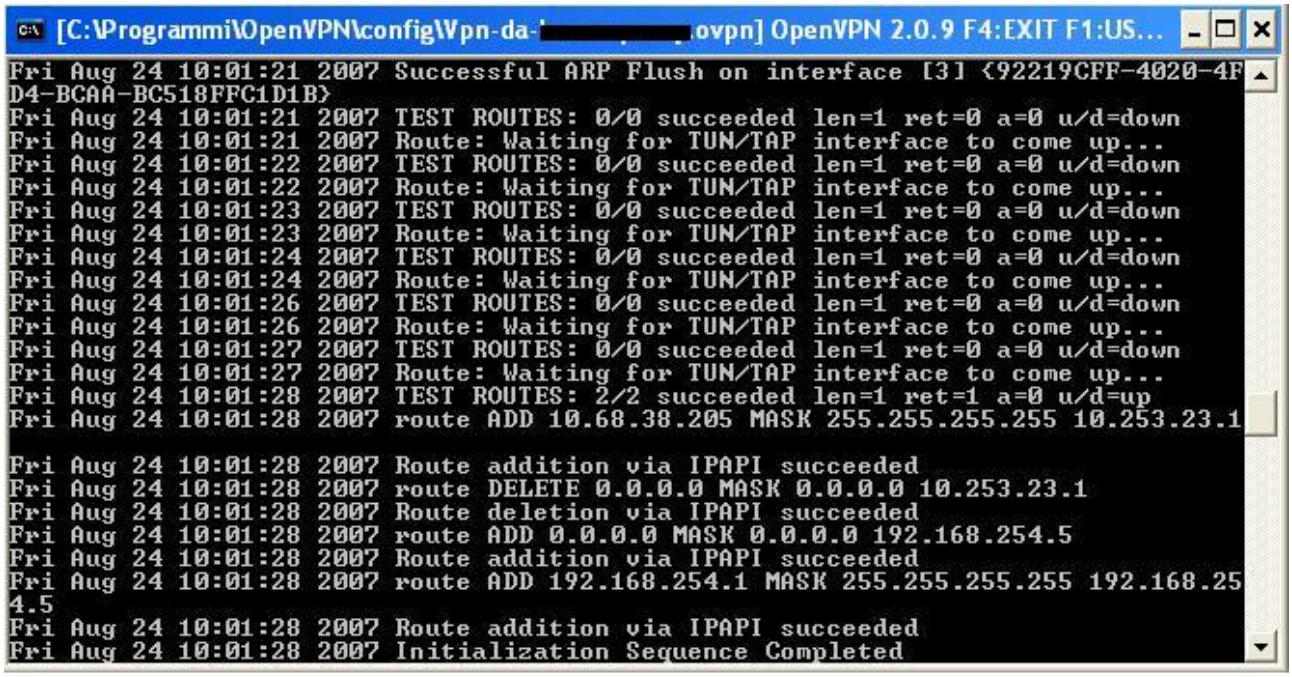


In questo caso è necessario aprire il browser, settare le direttive per passare dal proxy e richiamare una pagina web.

Salviamo il file e avviamolo.



Questo è il risultato se tutto procede per il verso giusto.



Segnalatemi pure critiche ed errori a <angelo at bsdsolutions.it>