VPN HOWTO

Matthew D. Wilson, *matthew@shinythings.com* `mailto:matthew@shinythings.com` v 1.0, Dec 1999

Questo HOWTO descrive come configurare una Virtual Private Network con Linux. Traduzione: Michele Tognon *michele.tognon@tin.it* `mailto:michele.tognon@tin.it`

Contents

1 Introduzione

1.1 Perchè ho scritto questo HOWTO

Lavoro alla Real Networks e noi abbiamo bisogno di un servizio VPN. Questo era il mio primo vero progetto, e ho imparato molto di più Linux con questo lavoro che non con qualsiasi altro. Sono, quindi, finito ad usare la mia esperienza per scrivere questo documento , per condividere con altri quello che io avevo imparato così che possano fare cose molto interessanti con Linux!

1.2 Acknowledgements e Ringraziamenti

Voglio inizialmente ringraziare mia moglie Julie, senza di lei non sarei qui oggi. Voglio anche ringraziare Arpad Magosanyi, l'autore del primo VPN mini-howto e pty-redir, l'utilita del quale tutto ciò è stato possibile. Jerry, Rod, Glen, Mark V., Mark W., and David, You guys rock! Grazie a tutti per l'aiuto. (ndT: io, invece, voglio ringraziare chi mi aiuterà a correggere questo documento e, naturalmente, la mia Raffaella.)

1.3 Formato di questo documento

Questo documento è diviso in 5 sezioni.

Sezione 1: Introduzione

Questa sezione

Sezione 2: Teoria

Teoria base di una VPN. Che cos'è una VPN e come lavora. Leggi questa sezione se sei completamente nuovo alle VPN.

Sezione 3: Server

Questa sezione descrive come si configura un server VPN.

Sezione 4: Client

Questa sezione descrive come viene configurato un client VPN.

Sezione 5: Implementazione

Una implementazione passo-passo di una configurazione VPN di prova.

Sezione 6: Addenda

Altre informazioni che potrebbero essere utili.

1.4 Copyright and Disclaimer

1.5 Storia del Documento

Il mini-HOWTO originale sulle VPN è stato scritto da *Arpad Magosanyi* `mag@bunuel.tii.matav.hu` nel 1997. Egli mi ha dato permesso di prendere il suo documento ed estenderlo in un HOWTO in piena regola. Tutto questo non sarebbe stato ppossibile senza il suo documento originale. Grazie ancora Arpad. :)

La Versione 1.0 di questo HOWTO è stata completata il 10 Dicembre, 1999. La traduzione in Italiano è stata completata in Marzo 2001.

1.6 Documenti correlati

- *Networking Overview HOWTO* `/HOWTO/Networking-Overview-HOWTO.html`
- *Networking HOWTO* `/HOWTO/NET3-4-HOWTO.html`
- *VPN-Masquerade HOWTO* `/HOWTO/VPN-Masquerade-HOWTO.html`

2 Teoria

2.1 Che cos'è una VPN?

VPN stà per Virtual Private Network. Una VPN usa internet e i suoi meccanismi di trasporto (TCP/IP), mantenendo la sicurezza dei dati.

2.1.1 Ma in definitiva, che cos'è una VPN?

Bene, ci sono molte risposte per questa domanda. Dipende dalla struttura della tua rete. La configurazione più comune è quella di avere una singola rete interna, con i nodi remoti che usano la VPN per poter accedere pienamente alla rete interna centrale. I nodi remoti sono comunemente uffici periferici o impegati che lavorano a casa. Tu puoi anche collegare due piccole (oppure grandi!) reti per realizzare una rete più grande.

2.1.2 Quindi, come lavora?

Detto semplicemente, per fare una VPN, devi creare un secure tunnel tra due reti e instradare gli indirizzi IP attraverso di esso. Se ti sei già perso con questa descrizione, dovresti leggere *The Linux Networking Overview HOWTO* `http://www.linuxdoc.org/HOWTO/Networking-Overview-HOWTO.html` per capire il networking con Linux.

Vi prego di sopportarmi, la mia arte ASII potrebbe costringervi ad un certo lavoro di comprensione.

```
                            \          \
                --------    /          /      --------
  Remote ______| Client |______\ Internet \_____| Server |______ Private
  Network      | Router |      /          /     | Router |       Network
                --------    \          \      --------
                            /          /
```

```
                          Client Router
               ----------------------------------------------------
              |   /->    10.0.0.0/255.0.0.0    \                   |
 Remote       |  |-->  172.16.0.0/255.240.0.0  |--> Tunnel >---\   |
 Network >---|--|--> 192.168.0.0/255.255.0.0 /                 |--|----> Internet
192.168.12.0 |  |                                               |  |
              |   \-----> 0.0.0.0/0.0.0.0 --> IP Masquerade >--/   |
               ----------------------------------------------------
```

```
                         Server Router
             ----------------------------------------------------
            |                   /->    10.0.0.0/255.0.0.0    \   |
            |  /--> Tunnel >--|-->  172.16.0.0/255.240.0.0   |--|----> Private
Internet >--|--|                \--> 192.168.0.0/255.255.0.0 /   |      Network
            |  |                                                 |     172.16.0.0/12
            |  \-----> 0.0.0.0/0.0.0.0 -----> /dev/null          |    192.168.0.0/16
             ----------------------------------------------------
```

Il diagramma sopra riportato fa vedere come la rete dovrebbe essere configurata. Se non sai che cos'è l'IP Masquerade, probabilmente non dovresti essere qui. Vai a leggere

The Linux Networking Overview HOWTO `/HOWTO/Networking-Overview-HOWTO.html` e torna quando avrai una idea di cosa si tratta.

Il Client Router è un computer con Linux installato che funge da gateway/firewall per la rete remota. Come puoi vedere, la rete remota usa la numerazione IP per una rete locale 192.168.12.0. Per rendere semplice lo schema, ho messo le informazioni sul routing locale nei routers (client e Server). L'idea di base è di instradare il traffico di rete per tutte le reti private (10.0.0.0, 172.16.0.0, e 192.168.0.0) attraverso il tunnel. La configurazione indicata qui è un modo. Cioè, mentre la rete remota può vedere la rete privata, la rete privata non deve necessariamente vedere la rete remota. Per far in modo che questo accada, devi specificare che gli instradamenti del traffico di rete siano bidirezionali.

Dallo schema dovreste anche notare che tutto il traffico che esce dal client router sembra provenire dal client router stesso, cioè, tutto da uno stesso indirizzo IP. Potreste instadare indirizzi reali dall'interno della vostra rete ma ciò comporterebbe portarsi dietro tutti i problemi di sicurezza.

2.2 SSH e PPP

Il sistema di cui stò descrivendo l'implementazione della VPN usa SSH e PPP. Uso ssh per creare la connessione tunnel, e poi uso pppd per far transitare il traffico TCP/IP attraverso esso. Questo è ciò che fa il tunnel.

Il vero segreto per ottenere che ssh e pppd lavorino bene è l'utility scritta da Arpad Magosanyi che permette la re-direzione dell' I/O standard in una pseudo tty. Ciò permette a pppd di comunicare attraverso ssh come se fosse un collegamento seriale. Dal lato server, pppd è lanciato a livello utente in una sessione ssh, e con questo si completa il gircompletando il collegamento. Dopo di questo, tutto ciò che dovete fare è l'instradamento.

2.3 Sistemi VPN alternativi

Naturalmente ci sono altri modi per configurare una VPN, qui ne presenterò un paio.

2.3.1 PPTP

PPTP è la risposta Microsoft per la VPN. E' supportato da Linux, ma è conosciuto per avere seri problemi di sicurezza (NdT: tanto per cambiare ;-)). Non descriverò come usarlo dal momento che l'argomento viene coperto da *Linux VPN Masquerade HOWTO* `http://www.linuxdoc.org/HOWTO/VPN-Masquerade-HOWTO.html` .

2.3.2 IP Sec

IP Sec è un set di protocolli alternativi a SSH. Attualmente non conosco tanto di più a riguardo, così se qualcuno volesse aiutarmi con una loro descrizione sarebbe bene accolta. Anche con questi non mi cimenterò a descriverne il loro uso poichè l'argomento è coperto da

Linux VPN Masquerade HOWTO `http://www.linuxdoc.org/HOWTO/VPN-Masquerade-HOWTO.html` .

2.3.3 CIPE

CIPE è un sistema di crittazione della rete a livello kernel che è più adatto alle esigenze di configurazioni a livello enterprise. Puoi trovare ulteriori informazioni a

homepage del CIPE `http://sites.inka.de/sites/bigred/devel/cipe.html` . Ho pianificato di esaminare più a fondo la questione il prima possibile, così avrò notizie più approfondite fra qualche tempo.

3 Server

Questa sezione descrive come configurare le cose dal lato server, propongo prima questo poichè senza un server, il client è praticamente inutile.

3.1 Sicurezza - tenere fuori gli altri

La sicurezza è molto importante per una VPN. Questo perchè, in primo luogo, la responsabilità della costruzione della VPN è tua, giusto? Bisogna tenere a mente alcune cose mentre si configura il server.

3.1.1 Controlla i tuoi daemons

Dal momento che questo server passa dati da entrambi i lati del firewall, fino al traffico interno della rete, è una buona idea di rendere sicura la VPN box il meglio che sia possibile. Puoi leggere molto più sulla sicurezza di Linux in *Linux Security HOWTO* `/HOWTO/Security-HOWTO.html` per i miei scopi ho eliminato tutti i demoni che girano in background eccetto sshd e Roxen Web. Uso il server web per scaricare un paio di file (i miei scripts, ecc) quando ho l'occasione di configurare delle nuove macchine che accedono alla VPN. Non uso un server FTP dal momento che è più difficile configurarlo che rendere accessibili un paio di file tramite server web. In più, solo io devo essere in grado di scaricare i files. Se si vuole realmente far girare molteplici server sul gateway, si dovrebbe pensare ad un accesso ristretto alle sole macchine sulla rete privata.

3.1.2 Non permettere passwords

Con questo titolo ho avuto la tua attnezione, vero? No, non si devono usare passwords, bisogna disabilitarle completamente. Tutta l'autenticazione su questa macchina deve essere fatta attraverso un sistema di autenticazione a chiave pubblica tipo ssh. In questo modo, solo chi possiede la chiave può entrare, ed è praticamente impossibile ricordarsi una chiave binaria lunga 530 caratteri.

Così cosa si deve fare? Bisogna editare il file /etc/passwd. Il secondo campo contiene la stringa della password, o alternativamente 'x' significando che il sistema di autentificazione lo si può trovare nel file /etc/shadow. Quello che devi fare è cambiare il campo da leggere con '*'. Questo dice al sistema di autentificazione che non ci sono password, e che non sono richieste.

Qui si può vedere un tipico esempio di file /etc/passwd:

```
...
nobody:x:65534:100:nobody:/dev/null:
mwilson:x:1000:100:Matthew Wilson,,,:/home/mwilson:/bin/bash
joe:*:504:101:Joe Mode (home),,,:/home/vpn-users:/usr/sbin/pppd
bill:*:504:101:Bill Smith (home),,,:/home/vpn-users:/usr/sbin/pppd
frank:*:504:101:Frank Jones (home),,,:/home/vpn-users:/usr/sbin/pppd
...
```

Nota che ho fatto molto di più che editare il secondo campo. Dirò di più a riguardo degli altri campi in seguito.

3.2 Accesso degli utenti - far entrare la gente

L'accesso degli utenti è eseguito tramite uno schema di autenticazione ssh. Come detto sopra, questo è come gli utenti accedono al sistema, mantenendo, nel contempo, un alto livello di sicurezza. Se non hai familiarità con ssh, controlla *http://www.ssh.org/* `http://www.ssh.org/`

Nota che io ho usato ssh versione 1, non la versione 2. C'è una grande differenza, infatti la versione 1 è free mentre la 2 no.

3.2.1 Configurazione di `sshd`

Ora si avrà bisogno di configurare sshd. Le seguenti opzioni dovrebbero essere presenti. L'idea è di disabilitare l'autenticazione delle password e l'autenticazione di rhosts. Le seguenti opzioni dovrebbero essere presenti nel file `/etc/sshd_config`.

```
PermitRootLogin yes
```

```
IgnoreRhosts yes
StrictModes yes
QuietMode no
CheckMail no
IdleTimeout 3d
X11Forwarding no
PrintMotd no
KeepAlive yes
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
UseLogin no
```

3.3 Restrizione degli utenti

Ora che puoi tenere i "cattivi" fuori, e far accedere solo i "buoni", devi assicurarti che i "buoni" si vedano tra loro. Questa è la cosa più facile in assoluto poichè non devi fare null'altro oltre a lanciare pppd. Questo può essere necessario o meno. Ho ristretto l'accesso degli utenti perchè il sistema che mantengo è dedicato alla VPN, gli utenti non hanno nessuna attività da fare su di esso.

3.3.1 sudo si o no

Esiste un piccolo programma chiamato sudo che permette all'amministratore di un sistema Unix di garantire a certi utenti la possibilità di lanciare certi programmi come root. Questo è necessario nel caso che pppd debba girare come root. Si avrà bisogno di usare questo metodo se si vuole permettere l'accesso della shell agli utenti. Leggi come configurare e usare sudo nelle pagine man relative a sudo stesso. L'uso di sudo è la miglior cosa da fare su sistemi "multi-uso" che mantengono un piccolo numero di utenti certificati e sicuri.

Se si decide di non permettere a nessuno di accedere alla shell, allora il modo migliore è tenerli fuori è di far in modo che la loro schell sia pppd. Ciò può essere fatto nel file /etc/passwd. Puoi vedere qui 3.1.2 (sopra) quello che ho fatto per gli ultimi tre utenti. L'ultimo campo del file /etc/passwd è la shell utente. Non hai bisogno di fare nulla di speciale a pppd per far in modo che funzioni. Verrà eseguito come root quando l'utente si connette. Questa è certamente la più semplice configurazione che si possa fare, e anche la migliore e più sicura. Ho descritto esattamente tutto quello che deve essere fatto più avanti nel documento. Puoi 5.7 (andare avanti) se ti pare.

3.4 Networking

Ora che gli utenti hanno accesso al sistema, dobbiamo essere sicuri che abbiano anche accesso alla rete. Facciamo questo usando le impostazioni di firewalling del kernel di Linux e la tabelle di routing. Usando i comandi `route` e `ipfwadm`, potremo configurare il kernel per instradare il traffico di rete nel modo più appropiato. Per ulteriori informazioni su `ipfwadm`, `ipchains` e `route` vedi

Linux Networking HOWTO `http://www.linuxdoc.org/HOWTO/Linux-Networking-HOWTO.html` .

3.4.1 Il Kernel

In modo che tutto ciò funzioni, si deve avere il kernel configurato correttamente. Se non si sa come compilare il proprio kernel, allora può essere una utile lettura il *Kernel HOWTO* `http://www.linuxdoc.org/HOWTO/`

`Kernel-HOWTO.html` . Hai bisogno di essere sicuro che le seguenti opzioni del kernel siano attivate in aggiunta a quelle basilari sulla rete. Uso un kernel 2.0.38 nel mio sistema.

Per i kernel 2.0:

- CONFIG_FIREWALL
- CONFIG_IP_FORWARD
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE (optional)
- CONFIG_IP_MASQUERADE_ICMP (optional)
- CONFIG_PPP

Per i kernel 2.2:

- CONFIG_FIREWALL
- CONFIG_IP_ADVANCED_ROUTER
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE (optional)
- CONFIG_IP_MASQUERADE_ICMP (optional)
- CONFIG_PPP

3.4.2 Regole di filtraggio

Primo, scriveremo delle regole di filtraggio per il firewall che permetteranno ai nostri utenti di accedere alla nostra rete interna, mentre restringeremo l'accesso a richieste che arrivano da internet. Se questo suona strano, pensalo in questo modo: loro hanno già l'accesso ad internet, così perchè usare il tunnel della VPN per accedere alla rete? E' uno spreco di banda e tempo macchina.

Le regole di filtraggio che useremo dipendono da quali reti interne usiamo. Ma fondamentalmente diciamo: "Permettere che il traffico proveniente dalla VPN, e destinato alle nostre reti interne, ci arrivi". Allora, come dobbiamo farlo? Come sempre, dipende. Se è presente un kernel 2.0, si usa il tool chiamato `ipfwadm`, se d'altra parte stai usando un kernel 2.2, usa l'utility chiamata `ipchains`.

Per configurare le regole con `ipfwadm`, lancialo con le opzioni simile alle seguenti:

```
# /sbin/ipfwadm -F -f
# /sbin/ipfwadm -F -p deny
# /sbin/ipfwadm -F -a accept -S 192.168.13.0/24 -D 172.16.0.0/12
```

Per configurare le regole con `ipchains`, lancialo con le opzioni simili alle seguenti:

```
# /sbin/ipchains -F forward
# /sbin/ipchains -P forward DENY
# /sbin/ipchains -A forward -j ACCEPT -s 192.168.13.0/24 -d 172.16.0.0/12
```

Per quelli che usano il kernel 2.2, prego leggete 6.1.3 (questo).

3.4.3 Routing

Così, ora i nostri utenti hanno il permesso di accedere alle nostre reti, ora abbiamo bisogno di dire al kernel dove spedire i pacchetti. Sul mio sistema, ho due schede ethernet, una è per la rete esterna, mentre l'altra è la rete interna. Quasto aiuta a tenere le cose sicure, poichè il traffico per l'esterno è mascherato dal nostro gateway, e qualsiasi traffico entrante è filtrato e instradato dal router Cisco. Per la miglior configurazione possibile, il routing dovrebbe essere semplice.

Quello che facciamo è instradare tutto il traffico destinato per le nostre reti private attraverso l'interfaccia interna, e tutto il resto attraverso l'interfaccia esterna. I comandi specifici di instradamento dipendono da quale rete interna stai usando. Sotto è presentato un esempio di come dovrebbe essere fatto. Queste linee sono, naturalmente, in aggiunta agli instradamenti base per le tue reti locali. Dubito, comunque, che userai tutti e 3 i gruppi di numeri interni.

```
Assumendo che 172.16.254.254 sia il tuo gateway interno:


# /sbin/route add -net 10.0.0.0 netmask 255.0.0.0 gw 172.16.254.254 dev
eth1
# /sbin/route add -net 172.16.0.0 netmask 255.240.0.0 gw 172.16.254.254
dev eth1
# /sbin/route add -net 192.168.0.0 netmask 255.255.0.0 gw 172.16.254.254
dev eth1
```

Una nota addizionale sull'instradamento. Se stai usando due modi per l'instradamento, ad esempio un ufficio remoto, allora avrai bisogno di fare una cosa ulteriore. Avrai bisogno di configurare la tabelle di instradamento sul server che ritorna sul client. Il modo più facile di far ciò è di lanciare un job cron ogni minuto che silenziosamente setta all'indietro l'instradamento. Non è una buona idea se il client non è connesso, `route` sparerà fuori un errore (che ti converrà spedire a `/dev/null`.)

4 Client

Ora esamineremo il lato client. In pratica, quando è usata per permettere l'accesso ad una rete remota, questa linuxbox può facilmente servire come un server Samba (networking con Windows), server DHCP, e anche come server web interno. La cosa importante da ricordare è che questa linuxbox deve essere più sicura possibile, poichè serve tutta la rete remota.

4.1 Il Kernel

Per prima cosa, tu devi avere il ppp compilato nel tuo kernel. Se si vuole permettere a molte macchine di usare il tunnel, allora si ha bisogno di un servizio di firewall e di fowarding. Se il client si indirizza verso una sola macchina, ppp è sufficiente.

4.2 Creare il collegamento

Il collegamento è creato lanciando `pppd` attraverso un pseudo terminale creato da `pty-redir` e connesso a `ssh`. Questo è fatto con una sequenza di comandi simile a questa:

```
# /usr/sbin/pty-redir /usr/bin/ssh -t -e none -o 'Batchmode yes' -c
```

```
blowfish -i /root/.ssh/identity.vpn -l joe > /tmp/vpn-device
# sleep 10

# /usr/sbin/pppd `cat /tmp/vpn-device`
# sleep 15

# /sbin/route add -net 172.16.0.0 gw vpn-internal.mycompany.com netmask
255.240.0.0
# /sbin/route add -net 192.168.0.0 gw vpn-internal.mycompany.com netmask
255.255.0.0
```

Semplicemente, quello che fa è far girare ssh, redirezionare il suo imput e l'output su pppd. L'opzione passata a ssh lo configura per girare senza caratteri escape (-e), usando il blowfish crypto algorithm (-c), usando la specificazione dell'identità del file (-i), in terminal mode (-t), e con le opzioni 'Batchmode yes' (-o). I comandi di sleep sono usati per estendere l'esecuzione dei comandi così che ogni processo possa completare il suo setup prima che il successivo parta.

4.3 scripting

Naturalmente non vuoi dover digitare i comandi sopra riportati ogni volta che vuoi che il tunnel si attivi. Ho scritto un set di script bash che tengono il tunnel attivo e funzionante. Puoi scaricare il package da *qui* `http://www.shinythings.com/vpnd/vpnd.tar.gz` . Devi solamente scaricarlo e decomprimerlo in /usr/local/vpn. All'interno troverai tre files:

- vpnd: Script che controlla la connessione del tunnel.
- check-vpnd: Uno script da far girare tramite cron che controlla che vpnd sia attivo.
- pty-redir: un piccolo eseguibile richiesto per inizializzare il tunnel..

Avrai bisogno di editare lo script `vpnd` per settare cose come l'username dei client e il nome dei server. Avrai anche bisogno di modificare la sezione starttunnel dello script per specificare quale rete stai usando. Sotto c'è una copia dello script. Noterai che potresti mettere lo script in una directory differente, hai solamente bisogno di cambiare la variabile VPN_DIR.

```
#! /bin/bash
#
# vpnd: Monitor the tunnel, bring it up and down as necessary
#

USERNAME=vpn-username
IDENTITY=/root/.ssh/identity.vpn

VPN_DIR=/usr/local/vpn
LOCK_DIR=/var/run
VPN_EXTERNAL=vpn.mycompany.com
VPN_INTERNAL=vpn-internal.mycompany.com
PTY_REDIR=${VPN_DIR}/pty-redir
SSH=${VPN_DIR}/${VPN_EXTERNAL}
PPPD=/usr/sbin/pppd
```

```
ROUTE=/sbin/route
CRYPTO=blowfish
PPP_OPTIONS="noipdefault ipcp-accept-local ipcp-accept-remote local noauth
nocrtscts lock nodefaultroute"
ORIG_SSH=/usr/bin/ssh


starttunnel () {
   $PTY_REDIR $SSH -t -e none -o 'Batchmode yes' -c $CRYPTO -i $IDENTITY -l
$USERNAME > /tmp/vpn-device
   sleep 15

   $PPPD `cat /tmp/vpn-device` $PPP_OPTIONS
   sleep 15

   # Add routes (modify these lines as necessary)
   /sbin/route add -net 10.0.0.0 gw $VPN_INTERNAL netmask 255.0.0.0
   /sbin/route add -net 172.16.0.0 gw $VPN_INTERNAL netmask 255.240.0.0
   /sbin/route add -net 192.168.0.0 gw $VPN_INTERNAL netmask 255.255.0.0
}

stoptunnel () {
   kill `ps ax | grep $SSH | grep -v grep | awk '{print $1}'`
}

resettunnel () {
   echo "reseting tunnel."
   date >> ${VPN_DIR}/restart.log
   eval stoptunnel
   sleep 5
   eval starttunnel
}

checktunnel () {
   ping -c 4 $VPN_EXTERNAL 2>/dev/null 1>/dev/null

   if [ $? -eq 0 ]; then
      ping -c 4 $VPN_INTERNAL 2>/dev/null 1>/dev/null
      if [ $? -ne 0 ]; then
         eval resettunnel
      fi
   fi
}

settraps () {
   trap "eval stoptunnel; exit 0" INT TERM
   trap "eval resettunnel" HUP
   trap "eval checktunnel" USR1
}
```

```
runchecks () {
   if [ -f ${LOCK_DIR}/tunnel.pid ]; then
      OLD_PID=‘cat ${LOCK_DIR}/vpnd.pid‘
      if [ -d /proc/${OLD_PID} ]; then
         echo "vpnd is already running on process ${OLD_PID}."
         exit 1
      else
         echo "removing stale pid file."
         rm -rf ${LOCK_DIR}/vpnd.pid
         echo $$ > ${LOCK_DIR}/vpnd.pid
         echo "checking tunnel state."
         eval checktunnel
      fi
   else
      echo $$ > ${LOCK_DIR}/vpnd.pid
      eval starttunnel
   fi
}

case $1 in
    check)  if [ -d /proc/‘cat ${LOCK_DIR}/vpnd.pid‘ ]; then
               kill -USR1 ‘cat ${LOCK_DIR}/vpnd.pid‘
               exit 0
            else
               echo "vpnd is not running."
               exit 1
            fi ;;

    reset)  if [ -d /proc/‘cat ${LOCK_DIR}/vpnd.pid‘ ]; then
               kill -HUP ‘cat ${LOCK_DIR}/vpnd.pid‘
               exit 0
            else
               echo "vpnd is not running."
               exit 1
            fi ;;

   --help | -h)
            echo "Usage: vpnd [ check | reset ]"
            echo "Options:"
            echo "     check    Sends running vpnd a USR1 signal, telling it
to check"
            echo "              the tunnel state, and restart if neccesary."
            echo "     reset    Sends running vpnd a HUP signal, telling it to
reset"
            echo "              it’s tunnel connection." ;;
esac

ln -sf $ORIG_SSH $SSH
settraps
runchecks
```

```
while true; do
   i=0
   while [ $i -lt 600 ]; do
      i=((i+1))
      sleep 1
   done
   eval checktunnel
done
```

4.4 LRP - Linux Router Project

Io attualmente faccio girare questa configurazione su un Pentium 90 con una distribuzione LRP di Linux. LRP è una distribuzione di Linux che stà in un solo floppy disk. Puoi saperne di più a *http://www.linuxrouter.org/* `http://www.linuxrouter.org/` . Puoi scaricare il mio package LRP per il client VPN da *qui* `http://www.shinythings.com/vpnd/vpnd.lrp` . Avrai bisogno pure dei pacchetti ppp che ssh che puoi trovare nel sito di LRP.

5 Implementazione

In questa sezione spiegherò passo passo come settare un sistema VPN. Inizierò con il server, e poi mi muoverò sul client. Per una spiegazione migliore farò un esempio dove ho inventato una situazione che dovrebbe richiedere un paio di modi differenti di configurare una VPN.

5.1 Pianificazione

Immaginate di avere una società chiamata mycompany.com. Al nostro quartier generale, usiamo il network 192.168.0.0, dividendo la classe B in 256 classi C di network che permettono il routing. Abbiamo da configurare solamente due piccoli uffici remoti, e vogliamo aggiungerli alla nostra rete. Vogliamo, inoltre, permettere agli impiegati di lavorare da casa usando una linea DSL e le connessioni cable modem al posto di fargli usare una connessione dialup. Per iniziare, abbiamo bisongo di pianificare un pò le cose.

Ho deciso che voglio dare ad ogni ufficio remoto un network in classe C per permettergli di espanderlo se necessario. Così, riservo le sottoreti da 192.168.10.0 a 192.168.11.0. Decido, inoltre, che per gli utenti casalinghi darò indirizzi IP sufficienti da non aver bisogno di farne il masquerading dal lato server della VPN. Ogni client avrà un IP proprio interno. Così, ho bisogno di riservare un'altra classe C per questo, diciamo 192.168.40.0. La sola cosa che devo fare ora è aggiungere questi range al mio router. Immaginate che la nostra società possegga un piccolo Cisco (192.168.254.254) che gestisce tutto il traffico attraverso la nostra OC1. Configurando l'instradamento sul Cisco in modo che il traffico diretto a queste reti riservate vada sul nostro server VPN (192.168.40.254). Metterò il server VPN nella rete degli utenti casalinghi per una ragione che diverrà chiara in seguito. Chiamerò l'interfaccia esterna del server vpn.company.com, e quella interna vpn-internal.mycompany.com.

Per gli indirizzi esterni, non abbiamo bisogno di conoscerli. L'unica cosa che si deve avere è il proprio indirizzo, fornito dal proprio ISP.

5.2 Raccogliere gli strumenti

Ora abbiamo bisogno di alcuni software, trova i seguenti programmi e installali dove specificato.

5.2.1 Per il Server:

- pppd (versione 2.3 o superiore)
- ssh (versione 1.2.26 o superiore)

5.2.2 Per il Client:

- pppd (stessa versione del server)
- ssh
- *pty-redir* `ftp://ftp.vein.hu/ssa/contrib/mag/pty-redir-0.1.tar.gz`

5.3 Server: compilare il kernel

Per iniziare avrai probabilmente bisogno di ricompilare il kernel per il server. Devi essere sicuro che le seguenti opzioni del kernel siano attivate in aggiunta alle opzioni di networking base e al resto che tu pensi possa servirti. Se non hai mai ricompilato il kernel prima ad ora leggi *Kernel HOWTO* `/HOWTO/Kernel-HOWTO.html` .

Per i kernel 2.0:

- CONFIG_FIREWALL
- CONFIG_IP_FORWARD
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_PPP

Per i kernel 2.2:

- CONFIG_FIREWALL
- CONFIG_IP_ADVANCED_ROUTER
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_PPP

5.4 Server: Configurare la rete

Se stai approntando un server che ha solo una scheda di rete, suggerisco l'idea di acquistarne un'altra, e ricablare la tua rete. La cosa migliore per tenere la tua rete privata è di usare le schede su reti fisicamente distinte. Così se hai due schede di rete, avrai bisogno di sapere come configurarle entrambe. Useremo eth0 per l'interfaccia esterna, e eth1 per l'interfaccia interna.

5.4.1 Configurare le interfaccie di rete

Per prima cosa configureremo l'interfaccia esterna del server. Dovresti già sapere come fare, e probabilmente averlo già fatto. Se non lo hai ancora fatto, ora è il momento. Se non sai come, vai a leggere l'url *Networking HOWTO* `/HOWTO/NET3-4-HOWTO.html`

Ora affrontiamo l'interfaccia interna. In accordo con gli indirizzi scelti, l'interfaccia interna del server è 192.168.40.254. In questo modo abbiamo configurato l'interfaccia.

Per i kernel 2.0, usa le impostazioni seguenti:

```
# /sbin/ifconfig eth1 192.168.40.254 netmask 255.255.255.0 broadcast
192.168.40.255
# /sbin/route add -net 192.168.40.0 netmask 255.255.255.0 dev eth1
```

Per i kernel 2.2, usa le impostazioni seguenti:

```
# /sbin/ifconfig eth1 192.168.40.254 netmask 255.255.255.0 broadcast
192.168.40.255
```

A questo punto le interfaccie sono attive. Puoi parlare alle macchine attraverso entrambe le reti connesse al server.

5.4.2 Imposta i percorsi (routes)

Ora possiamo parlare alle macchine nella nostra rete locale, ma non possiamo farlo sul resto della nostra rete interna. Questo richiederà un pò di linee di codice. Per far in modo di raggiungere le altre macchine nelle altre sottoreti, abbiamo bisogno di avere un percorso che indichi al traffico di andare nel router Cisco. Ecco la linea di codice per far questo:

```
# /sbin/route add -net 192.168.0.0 gw 192.168.254.254 netmask 255.255.0.0
dev eth1
```

Questa linea dice al kernel che tutto il traffico destinato per la rete 192.168.0.0 dovrebbe uscire da eth1, e che dovrebbe essere passato fuori al Cisco. Il traffico per la nostra rete locale andrà dove deve poichè le tabelle di routing sono ordinate per formato di netmask. Se noi vogliamo avere altre reti interne nella nostra rete, ci servirà una linea di codice come quella sotto riportata per ogni rete.

5.4.3 Realizzare le regole di filtraggio. Making filter rules

Ok, così ora possiamo raggiungere le macchine di cui potremmo avere bisogno. Ora abbiamo bisogno di scrivere le regole di filtraggio del firewall che permettono o negano l'accesso attraverso il server VPN.

Per impostare le regole con `ipfwadm`, lancialo in questa maniera:

```
# /sbin/ipfwadm -F -f
# /sbin/ipfwadm -F -p deny
# /sbin/ipfwadm -F -a accept -S 192.168.40.0/24 -D 192.168.0.0/16
# /sbin/ipfwadm -F -a accept -b -S 192.168.10.0/24 -D 192.168.0.0/16
# /sbin/ipfwadm -F -a accept -b -S 192.168.11.0/24 -D 192.168.0.0/16
```

Per impostare le regole con `ipchains`, lancialo in questa maniera:

```
# /sbin/ipchains -F forward
# /sbin/ipchains -P forward DENY
# /sbin/ipchains -A forward -j ACCEPT -s 192.168.40.0/24 -d 192.168.0.0/16
# /sbin/ipchains -A forward -j ACCEPT -b -s 192.168.10.0/24 -d
192.168.0.0/16
# /sbin/ipchains -A forward -j ACCEPT -b -s 192.168.11.0/24 -d
192.168.0.0/16
```

Tutto ciò dice al kernel di rigettare tutto il traffico eccetto quello che arriva dalla rete 192.168.40.0/24 e destinato alla rete 192.168.0.0/16. In più la regola dice al kernel che il traffico passante tra le reti 192.168.10.0/24 e 192.168.0.0/16 è permesso, e lo stesso dicasi per la rete 192.168.11.0. Queste due ultime regole sono bidirezionali, questo è importante per permettere al routing di lavorare in entrambi i modi.

5.4.4 Routing

Per gli utenti casalinghi, tutto inizierà a funzionare bene da adesso. tuttavia, per gli uffici remoti, abbiamo bisogno di instradare ancora qualcosa. Prima di tutto, dobbiamo dire al router principale, o Cisco, che gli uffici remoti sono dietro al Server VPN. Ora specifichiamo gli instradamenti sul Cisco che dicono di spedire il traffico destinato agli uffici remoti al server VPN. Ora dobbiamo dire al server VPN cosa fare con il traffico destinato agli uffici remoti. Per far ciò, useremo il comando `route` sul server. Il solo problema è che per far in modo che il comando `route` funzioni, il link di rete deve essere attivo perchè se fosse disattivato, questo specifico routing andrebbe perso. La soluzione è quella di aggiungere gli instradamenti quando gli utenti si connettono, o più semplicemente, lanciare il comando route frequentemente. Così, crea lo script e aggiungilo nel tuo crontab per lanciarlo ogni pochi minuti, in esso, metti le seguenti linee:

```
/sbin/route add -net 192.168.11.0 gw 192.168.10.253 netmask 255.255.255.0
/sbin/route add -net 192.168.10.0 gw 192.168.11.253 netmask 255.255.255.0
```

5.5 Server: Configurare `pppd`

Ora configureremo pppd sul server per gestire le connessioni VPN. Se stai già usando questo server per gestire gli utenti in dialup o per connettere te stesso, allora potrai notare che queste modifiche avranno effetto su tutti i servizi. Parleremo di come evitare i conflitti alla fine di questa sezione.

5.5.1 `/etc/ppp/`

Questa directory deve contenere necessariamente alcuni files. Probabilmente avrai già un file chiamato `options`. Questo file contiente tutte le opzioni globali di `pppd`. Queste opzioni non possono essere sovrascritte da `pppd` da linea di comando.

5.5.2 `/etc/ppp/options`

Il file `options` dovrebbe contenere almeno le seguenti righe:

```
ipcp-accept-local
ipcp-accept-remote
proxyarp
noauth
```

Le prime due righe dicono a pppd cosa accettare dall'altro lato dell'indirizzo IP ricevuto. Questo è necessario quando vengono collegati gli uffici remoti, ma può essere disabilitato se si collegano solo utenti da casa. Va bene lasciarlo attivo, poichè non impedisce al server l'assegnazione degli indirizzi, esso dice solo che è pronto ad accettare le richieste del client.

La terza linea è molto importante. Dalle man page di pppd:

```
proxyarp
       Add an entry to this system's ARP [Address  Resolu-
       tion  Protocol]  table  with  the IP address of the
       peer and the Ethernet address of this system.  This
       will  have  the effect of making the peer appear to
       other systems to be on the local ethernet.
```

Questo è importante perchè se non viene fatto, il traffico locale non sarà in grado di ritornare attraverso il tunnel.

L'ultima linea è la più importante. Questa dice a pppd di permettere connessioni senza username e password. Tale procedura è sicura finchè l'autenticazione viene gestita da sshd.

5.5.3 Evitare conflitti

Se gestisci altri servizi con pppd, dovrai considerare che le configurazioni di questi altri servizi potrebbero non essere le stesse di cui avrà bisogno una VPN. pppd è disegnato in modo tale che opzioni del file principale /etc/ppp/options non possano essere sovrascritte da opzioni specificate in runtime. Questo è fatto, logicamente, per ragioni di sicurezza. Per evitare conflitti, si deve determinare quali opzioni causano il conflitto, e spostarle dal file principale in un file di opzioni separato che venga caricato quando l'applicazione appropriata di pppd stà girando.

5.6 Server: Configurare sshd

Qui di seguito descrivo il mio file /etc/sshd_config. Il tuo dovrebbe essere lo stesso o almeno simile:

```
# This is the ssh server system wide configuration file.

Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh_host_key
RandomSeed /etc/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts yes
StrictModes yes
QuietMode no
FascistLogging yes
CheckMail no
IdleTimeout 3d
X11Forwarding no
PrintMotd no
```

```
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
UseLogin no
```

Il punto importante da notare è che l'autenticazione delle password è disabilitata come i servizi `"r"`. Ho anche disabilitato il controllo dell'email e il 'messaggio del giorno' può confondere `pppd` dal lato client. Permetto ancora il login da root, ma può essere eseguito solamente con una chiave, ciò è adeguatamente sicuro.

5.7 Server: configurare gli account utente

Ora configureremo gli account utente.

5.7.1 Aggiungere il gruppo `vpn-users`

Lancia semplicemente:

```
# /usr/sbin/groupadd vpn-users
```

Ora, edita `/etc/group` e guarda l'ulima linea. Dovrebbe essere la linea del gruppo vpn-users. Nota il terzo campo. Questo è l'ID di gruppo (GID). Segnatela, ne avremo bisogno fra un minuto. Per questo esempio il GID è 101.

5.7.2 Creare la home directory di `vpn-users`

Useremo una home directory singola per tutti gli utenti del gruppo. Cosi' si dovrà lanciare semplicemente:

```
# mkdir /home/vpn-users
```

5.7.3 La directory `.ssh`

Ora creiamo la directory `.ssh` nella home directory di `vpn-users`.

```
# mkdir /home/vpn-users/.ssh
```

5.7.4 Aggiungere utenti

Ora inizia la parte divertente. Andiamo ad editare il file `/etc/passwd` a mano. :) Normalmente è il sistema a gestire questo file, ma per un setup 'sporco' come questo, è più semplice farselo. Per iniziare, apri il file `/etc/passwd` e verifica cosa si trova all'interno. Qui sotto c'è un esempio di ciò che dovresti trovare:

```
...
nobody:x:65534:100:nobody:/dev/null:
mwilson:x:1000:100:Matthew Wilson,,,:/home/mwilson:/bin/bash
joe:*:1020:101:Joe Mode (home),,,:/home/vpn-users:/usr/sbin/pppd
```

```
bill:*:1020:101:Bill Smith (home),,,:/home/vpn-users:/usr/sbin/pppd
frank:*:1020:101:Frank Jones (home),,,:/home/vpn-users:/usr/sbin/pppd
...
```

Il primo utente è essenzialmente di default. Il secondo sono io. :) Dopo questi sono stati fatti alcuni utenti specifici per vpn-users. Il primo campo è lo username, e il secondo è il campo password. Il terzo è lo user ID (UID) e il quarto è il l'ID di gruppo (GID). Dopo questi c'è un campo che specifica alcune informazioni su chi sia l'utente. Il sesto campo è l'home directory dell'utente e l'ultimo campo specifica la shell. Come puoi vedere, ogni campo è separato da un due punti. Guarda le ultime tre righe. La sola differenza tra loro è lo username nel primo campo, e le informazioni utente sul quinto campo. Quello che vogliamo fare è creare righe come queste per ogni utente. E' meglio non usare solo un utente per tutte le connessioni, facendo in quella maniera non saresti in grado di distinguere nulla. Così, copia l'ultima linea di questo file ed editala cosi che assomigli a quella sopra riportata. Sii sicuro che il secondo campo sia un asterisco (*). Il secondo campo dovrebbe essere uguale per tutte le righe del file. Io ho usato 1020. Tu puoi usare un numero sopra a 1000, poichè i numeri più bassi sono tipicamente riservati per usi del sistema. Il quarto campo dovrebbe essere riservato all'ID di gruppo di vpn-users. E' tempo, ora, di usare quanto segnatoci precedentemente. Così scrivi l'ID di gruppo qui. Come ultima cosa cambia la home directory in `/home/vpn-users`, e la shell in `/usr/sbin/pppd`. E questo è tutto. Ora copia la riga per aggiungere utenti.Modifica il primo e il quinto campo e l'utente sarà configurato.

5.8 Server: Amministrazione

Uno dei vantaggi di usare questo sistema per gli account utente è che tu puoi avvantaggiarti dei comandi di amministrazione degli utenti di UNIX. Dal momento che ogni client si connette come utente, tu puoi usare i metodi standard per avere le statistiche utente. I seguenti sono alcuni comandi che mi piace usare per vedere se il tutto funziona a dovere.

who

Visualizza gli utenti correntemente connessi, così come quando si sono connessi, da dove (nome o IP, e su quale porta).

w

Questo comando visualizza una lista molto estesa di chi è correntemente connesso. Ti dice pure l'uptime e i carichi per il sistema. In più specifica il processo corrente dell'utente (il quale dovrebbe essere -pppd per i client della VPN) così come l'idle time, e l'uso corrente della CPU per tutti i processi cosiì come il processo corrente. Leggi le man page di `w` per saperne di più.

last [username]

Questo visualizza lo storico del login di un utente specifico, o per tutti gli utenti se l'username passato non è presente. E' molto utile per testare quanto bene stà funzionando il tunnel e visualizza la lunghezza del tempo che l'utente è connesso, o lo stato nel quale l'utente è connesso al momento. Ti metto in guardia del fatto che se un sistema è attivo da molto tempo questa lista potrebbe essere molto lunga. Filtrala con una pipe attraverso `grep` o `head` per trovare esattamente quello che vuoi sapere.

Puoi anche controllare quali utenti hanno il permesso di connetersi modificando il file `/home/vpn-users/.ssh/authorized_keys`. Se rimuovi la linea con la chiave pubblica dell'utente da questo file, non riuscirà più a connettersi.

5.9 Client: compilare il kernel.

Ora prendiamo in considerazione il client. Per prima cosa dobbiamo ricompilare il kernel in modo che possa supportare tutte le funzioni di cui abbiamo bisogno. Le richieste minime sono di avere il ppp nel kernel. Dopo questo, avrai bisogno del forwarding, firewalling e del gatewaying solo se vuoi permettere ad altre macchine di accedere al tunnel. Per questo esempio, configurerò una delle macchine dell'ufficio remoto del mio schema di esempio. Aggiungiamo le seguenti opzioni sul tuo kernel. Ancora, se non hai mai compilato un kernel in vita tua, leggi *Kernel HOWTO* `/HOWTO/Kernel-HOWTO.html` .

Per i kernel 2.0:

- CONFIG_PPP
- CONFIG_FIREWALL
- CONFIG_IP_FORWARD
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE
- CONFIG_IP_MASQUERADE_ICMP

Per i kernel 2.2:

- CONFIG_PPP
- CONFIG_FIREWALL
- CONFIG_IP_ADVANCED_ROUTER
- CONFIG_IP_FIREWALL
- CONFIG_IP_ROUTER
- CONFIG_IP_MASQUERADE
- CONFIG_IP_MASQUERADE_ICMP

5.10 Client: Configurare la rete

Ora dovremo configurare la rete sul tuo PC client. Assumiamo di aver già configurato la rete esterna e che questa funzioni. Ora configureremo l'interfaccia interna del client in servizio nella tua intranet.

5.10.1 Interfaccia

Abbiamo bisogno per prima cosa di attivare l'interfaccia interna di rete. Per far cio, aggiungi le seguenti linee al tuo file (o equivalente) `/etc/rc.d/rc.inet1`:

Per i kernel 2.0:

```
/sbin/ifconfig eth1 192.168.10.253 broadcast 192.168.10.255 netmask
255.255.255.0
/sbin/route add -net 192.168.10.0 netmask 255.255.255.0 dev eth1
```

Per i kernel 2.2:

```
/sbin/ifconfig eth1 192.168.10.253 broadcast 192.168.10.255 netmask
255.255.255.0
```

5.10.2 Regole di filtraggio

Per configurare l'ufficio remoto, dovremo impostare le regole di filtraggio che permettano al traffico di andare in entrambe le direzioni attraverso il tunnel. Aggiungi le seguenti linee al tuo file (o equivalente) `/etc/rc.d/rc.inet1`:

Per i kernel 2.0:

```
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p deny
/sbin/ipfwadm -F -a accept -b -S 192.168.10.0/24 -D 192.168.0.0/16
```

Per i kernel 2.2:

```
/sbin/ipchains -F forward
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -j ACCEPT -b -s 192.168.10.0/24 -d 192.168.0.0/16
```

Avrai notato che queste linee assomigliano a quelle che abbiamo sul server. Questo perchè sono le stesse. Queste regole dicono semplicemente dove il traffico ha il permesso di andare, e cioè tra queste due reti.

5.10.3 Routing

Il solo instradamento extra di cui abbiamo bisogno è creato dallo script che attiva il tunnel.

5.11 Client: Configurare `pppd`

Non dovresti aver bisogno di editare il file `/etc/ppp/options` a questo punto. Controlla se l'opzione "auth" è presente, o alcune delle altre opzioni privilegiate. Provalo, e se fallisce, un `/etc/ppp/options` vuoto si attiverà semplicemente conservando le opzioni dal vecchio file per indagare su cosa si sia corrotto (se non è evidente). Forse non avrai bisogno di tutto ciò se usi `pppd` per null'altro che questo.

5.12 Client: Configurare `ssh`

Come root sul client, esegui le seguenti linee:

```
# mkdir /root/.ssh
# ssh-keygen -f /root/.ssh/identity.vpn -P ""
```

Questo creerà due files, `identity.vpn` e `identity.vpn.pub` nella directory `.ssh`. Il primo è la tua chiave privata, e dovrebbe essere tenuta al sicuro. *MAI SPEDIRLA ATTRAVERSO LA RETE* se non attravreso una sessione crittata. Il secondo file è la tua chiave pubblica, e puoi spedirla in qualsiasi posto tu voglia, essa serve solo per permettere l'accesso ad altri sistemi, e non può essere usata in sostituzione alla tua privata. E' un file di testo con una linea che rappresenta la tua chiave attuale. Alla fine della linea c'è il campo commento che puoi cambiare senza paura che la tua chiave sia sprotetta. Un esempio di chiave si presenta più o meno così:

```
1024 35 1430723736674162619588314275167.......250872101150654839
root@vpn-client.mycompany.com
```

La chiave reale è più lunga di questa, comunque, ma non voglio riempire lo schermo per rappresentarla tutta. Copia la chiave nel file sul server `/home/vpn-users/.ssh/authorized_keys`. Sii sicuro che ci sia una sola chiave per linea, e che ogni chiave non sia spezzata in più linee. Potrai alterare il campo commento con tutto ciò che ti sembrerà utile per ricordarti quale linea sia associata a quale utente. Ti raccomando caldamente questa pratica.

5.13 Client: Attivare la connessione

Ora proveremo ad attivare la connessione verso il server VPN. Primo avremo bisogno di tentare una connessione al server specificato nel file known_host nel file `ssh`. Lancia questo:

```
# ssh vpn.mycompany.com
```

Rispondi "yes" quando ti viene chiesto se vuoi continuare a connetterti. Il server ti risponderà "permission denied", ma è tutto ok. E' importante che usi lo stesso nome per il server che vuoi usare nello script di connessione. Ora scrivi le seguenti linee. Avrai bisogno di cambiare la maggior parte delle opzioni per mettere a punto la configurazione.

```
# /usr/sbin/pty-redir /usr/bin/ssh -t -e none -o 'Batchmode yes' -c
blowfish -i /root/.ssh/identity.vpn -l vpn-user vpn.mycompany.com >
/tmp/vpn-device

        (now wait about 10 seconds)

# /usr/sbin/pppd `cat /tmp/vpn-device` 192.168.10.254:192.168.40.254
```

Nota l'indirizzo IP specificato nella linea con pppd. Il primo è l'indirizzo del client alla fine del tunnel. Il secondo è l'indirizzo del server alla fine del tunnel, il quale è settato sugli indirizzi interni del server. Se tutto sembra funzionare, vai avanti. Se no, controlla se hai tutte le opzioni e se sono scritte correttamente. Se qualcosa continua a non funzionare, controlla la sezione 6.1 (sezione trabocchetti).

5.14 Client: Configurare gli instradamenti.

Ora configuriamo l'instradamento per spedire il traffico attraverso il tunnel. Lancia questo:

```
# /sbin/route add -net 192.168.0.0 gw vpn-internal.mycompany.com netmask
255.255.0.0
```

Dovresti ora essere in grado di comunicare con le macchine all'altro lato del tunnel. Fai una prova. Semplice, vero? Se non funziona, prova usando `ping` e `traceroute` per identificare dove si potrebbe annidare il problema. Se finalmente funziona, configura degli script che facciano il lavoro per te.

5.15 Client: Scripting

Usa lo script vpnd mostrato 4.3 (qui). Solamente devi modificarlo un poco. Esegui le seguenti modifiche:

- Cambia le variabili in cima per abbinarle alla tua configurazione. La maggior parte dovrebbero essere a posto, ma se ne hai bisogno puoi cambiarle.
- Line 27: aggiungi l'IP locale e remoto prima di $PPP_OPTIONS
- Line 31: Cambia questa linea, e le due dopo essa per configurare gli instradamenti per le tue reti interne.

5.15.1 Mantienila attiva

Anche se gli script bash sono generalmente stabili, abbiamo scoperto che possono fallire. Per essere sicuri che lo script `vpnd` rimanga attivo, aggiungiamo una linea alla crontab del client che lancia lo script `check-vpnd`. Viene lanciato ogni 5 minuti circa. Se `vpnd` stà effettivamente funzionando, `check-vpnd` non sprecherà risorse di CPU.

6 Addenda

6.1 Trabocchetti

Qui elencati ci sono alcuni problemi che mi sono accaduti utilizzando il sistema finora descritto. Li metto qui affinchè possano tornare utili. Se incappi in un nuovo problema per favore *spediscimelo via email* `mailto:matthew@shinythings.com` così che io ne possa tenere traccia e aiutare altri che si troveranno nella tua situazione.

6.1.1 read: I/O error

Questo errore arriva apparentemente da pppd. E' associato con una versione obsoleta di pppd. Se ti capita, prova ad aggiornare entrambi i lati della connessione (client e server) con l'ultima versione. Ho scoperto che la versione di pppd 2.2 ha questo problema, quindi uso in alternativa la versione 2.3.7 oppure la 2.3.8.

6.1.2 SIOCADDRT: Network is unreachable

Questo errore è generato da `route`. Ho notato che accade quando il tempo di ibernazione tra `ssh` and `pppd` non è abbastanza lungo. Se rilevi questo errore, lancia `ifconfig`, dovresti notare che non ci sono interfacce pppX attive. Questo significa che `ssh` non era stata fatta l'autenticazione prima che `pppd` fosse lanciato, e di conseguenza `pppd` non ha eseguito la connessione. Aumenta il tempo di attesa e i problemi si dovrebbero risolvere.

Mi meraviglierei se ci fossero alcune opzioni di pppd che risolvono questo problema.

6.1.3 IPv4 Forwarding e i kernel 2.2

Nei nuovi kernel 2.2, devi specificamente abilitare l'IP forwarding nel kernel al momento del boot. Questo con il seguente comando:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Senza questo, il kernel non inoltrerà alcun pacchetto, e il server non funzionerà, come non funzionerà nessun gateway per i client.

6.1.4 Routing

dovrebbe funzionare senza fiatare, ma bisogna fare attenzione quando processi indirizzi che non instradano traffico destinato all'indirizzo esterno del server della VPN attraverso il tunnel. Perchè non lo farà. (si, questa *è* per esperienza personale.)

6.2 Richieste Hardware e Software

6.2.1 Richieste Hardware Minime

Credeteci oppure no, questo sistema gira su un 486SX33 con 8 megabytes di RAM. Ovviamente non gira benissimo, infatti ha qualche problema quando c'è molto traffico.

Questo sistema lavora bene su un Pentium 75 con 16Mb di RAM, usando una distribuzione LPR caricata da floppy. con 6Mb di ramdisk, e 10Mb di spazio principale. Ho testato queste impostazioni caricando un filmato RealVideo 700kbit in streaming attraverso la VPN per oltre una ora.

Ora, però, faccio girare tutto su un Pentium 90 con una cheda Ethernet a 100Mbit economica.

6.2.2 Richieste Software

Questo sistema lavora sia con il kernel 2.0 che con il 2.2. Gli script che mantengono il tunnel attivo richiedono una shell bash ragionevolmente moderna. Ho notizia, tuttavia che certe versioni di bash presenti sulle distribuzioni Linux non lavorano molto bene con gli script.

Se qualcuno potesse aiutarmi a raffinare i miei script (oppure scrivere un eseguibile?) avrebbe i miei più infiniti ringraziamenti. Non sono sicuro del perchè, ma la mia shell bash non segue le regole e non sembra interpretare correttamente i segnali. Se realizzi qualche miglioramento, ti prego di spedirmelo via email a

matthew@shinythings.com `mailto:matthew@shinythings.com`