Mezzora SSH - JohnnyRun

www.lug-acros.org

INTRO

Ssh è un protocollo di comunicazione cifrata che permette principalmente di effettuare connessioni verso altri host, ottenendo un terminale remoto su cui lavorare. E' nato per essere l'evoluzione di “telnet”.
In un secondo momento l'implementazione client-server di questo protocollo ha permesso di sviluppare funzionalità “di supporto”, che esulano dal concetto di semplice terminale.
Questa Mezzora è dedicata a questi casi.

INTRODUZIONE ALLA CRITTOGRAFIA ASIMMETRICA

(o approfondimento, dipende da che parte si comincia)
Il protocollo si basa sul concetto di crittografia asimmetrica, ovvero sulla difficoltà computazionale di invertire una determinata operazione matematica. In soldoni questo tipo di crittografia si basa sul concetto che è più difficile fattorizzare il prodotto di due numeri primi che moltiplicarli tra loro.
[esempio da wikipedia]
E' più difficile fattorizzare 377 che moltiplicare tra loro i numeri primi 13 e 29.
Esistono altre operazioni difficilmente invertibili, ma il prodotto e la fattorizzazione è sicuramente il più usato.
La crittografia asimmetrica non avrebbe significato se questa difficoltà non diventasse “impossibilità computazionale”, ovvero la difficoltà di ottenere il risultato dell'operazione inversa in un tempo ragionevole.

RSA

p e q numeri primi
n=p*q
e più piccolo e coprimo di (p-1)(q-1)
d * e = 1 modulo (p-1)(q-1)
(n,e) -> chiave pubblica
(n,d) -> chiave privata

GENERAZIONE CHIAVI RSA

```
darkstar:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
ab:5b:80:b7:50:fd:57:61:d5:d0:b0:d1:0c:0e:2b:61 root@darkstar

UPLOAD DELLA CHIAVE
locale$ scp ~/.ssh/id_rsa.pub utenza@serverremoto:
Password:
utenza@serverremoto$ cat id_rsa.pub >> .ssh/authorized_keys2
utenza@serverremoto$ exit
e poi:

locale$ ssh utenza@serverremoto
Enter passphrase for key ....

SSH AGENT
gianni@darkstar:~$ env|grep SSH
SSH_AGENT_PID=4671
SSH_AUTH_SOCK=/tmp/ssh-jWuxED4630/agent.4630
Altrimenti, se non è avviato:
gianni@darkstar:~$ eval $(ssh-agent)
Tutti i processi figli avranno in comune questo environment, quindi potranno utilizzare l'ssh-agent

AGGIUNTA DI UNA CHIAVE ALL'AGENT:
$ ssh-add
[se la chiave privata è protetta da una passphrase, solo in questo momento ci verrà chiesto l'inserimento]
d'ora in poi:
$ssh utenza@serverremoto
permetterà il login sulla macchina remota senza chiederci alcuna password/passphrase.

Esempio tratto da un'esperienza personale:
“avviare un servizio su 10 server quasi contemporaneamente”

```
for host in server1 server2 server3 server4 server5 server6 server7 server8
server9 server10; do
ssh -o BatchMode=true utente@$host COMANDO &
done
```

TUNNEL con ssh
Pro:
-- facili da implementare
-- non c'è altro da installare (stunnel o altro)
-- ssh è presente quasi ovunque

Rimappare una porta locale a un host/porta remota

ssh -L 2222:*hostremoto*:22 utente@*serverremoto*
In questo caso, collegandosi alla porta locale (*localhost*) 2222 è come se ci si connettesse direttamente a *hostremoto* alla porta 22.
E' estremamente utile quando *localhost* e *serverremoto* si “vedono”, ma *localhost* e *hostremoto* non si “vedono” (per esempio: *hostremoto* è in *DMZ*).

Rimappare una porta remota a un host locale
ssh -R 2222:*hostlocale*:22 utente@*serverremoto*
In questo caso, collegandosi alla porta 2222 di *serverremoto*, è come se ci si collegasse alla porta 22 di *hostlocale*. I vantaggi sono sempre quelli di visibilità.
NOTA: la porta 2222 viene aperta su *serverremoto* solo sull'interfaccia lo, pertanto si può accedervi solo tramite *loopback*. Ma l'opzione -g permette l'apertura della porta su tutti i device di *serverremoto*.
Esperienza personale:
“oltre il firewall”

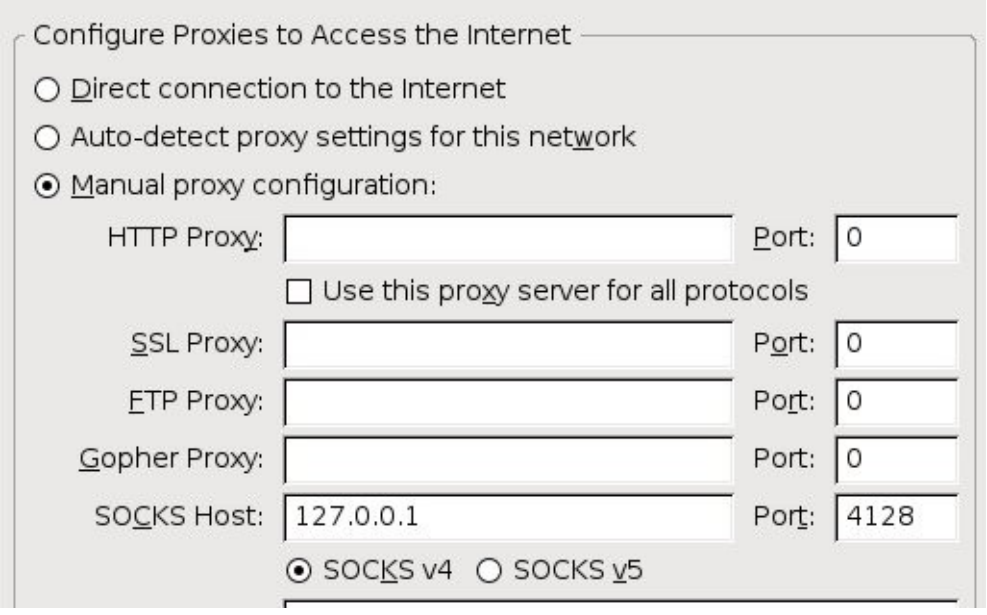
Avendo a disposizione due host, uno con ip pubblico su internet (completamente disponibile) e l'altro senza ip pubblico, volevo rendere accessibile un servizio della macchina nattata DA internet. Questo senza smanettare il router.
Dall'host con ip pubblico:

```
while :;
  ssh -L 2222:macchinanattata:22 -N localhost;
done
```

SSH SOCKS SERVER
Il problema è sempre quello della visibilità tra due host o tra un host e una rete.
Se volessimo che le nostre connessioni siano rimbalzate attraverso un altro host dove noi abbiamo un accesso, questo è possibile con l'opzione -D

$ssh -D 4128 -N utenza@*serverremoto*

[NOI]------>[*serverremoto*]------>[qualsiasi host in internet o LAN]
Per utilizzare i nostri programmi attraverso il socks server bisogna configurarli.
Esempio in firefox:



E' possibile rimbalzare tutte le connessioni TCP con *tsocks*
Dopo aver configurato tsocks.conf, basta dare “tsocks COMANDO”.
Esempio

```
$tsocks wget http://www.debian.org
```

Questa opzione è spesso usata per scavalcare regole troppo strette sui firewall aziendali, degli hotspot o delle postazioni alberghiere. Il “complice” è serverremoto, al quale ci si connette e rimbalzerà le nostre connessioni

Esempio:
Non riusciamo ad avviare gaim (ICQ/MSN) all'interno della rete universitaria, che però ci permette di usare ssh.

[NOI] --- |FIREWALL| --- (internet) --- [serverremoto:22]
[messenger.hotmail.com:1863]
Ancora:
ssh -D 4128 utenza@*serverremoto*
e un'appropriata configurazione di gaim.

Esperienza personale
“siamo tutti di Lucca”

Gli hotspot a pagamento non permettono di navigare senza aver pagato (ovvio, sono hotspot). Alcuni (pochi) permettono il traffico ssh, probabilmente per la manutenzione degli stessi
Ancora:
ssh -D 4128 utenza@*serverremoto*

VPN su SSH
Il problema è sempre quello della visibilità che manca, oppure quello di dover attraversare un tratto di rete considerato insicuro. Con i tunnel possiamo far passare tutto TCP, ma per gli altri protocolli?
Senza continuare a pensare allo scavallamento di un firewall è pensabile che un admin possa maneggiare la propria DMZ da internet, utilizzando tutti i protocolli, anche quelli non connessi
[NOI]---(internet)----[|FIREWALL aperto in ssh|]--- (DMZ)
Pro:
- serve solo ssh e pppd
- non serve impostare le rotte
- facile attraversamento dei firewall
- possibilità di ip dinamici
- tunnel multipli

Contro:
-I protocolli non connessi vengono trasportati su protocollo connesso (IP over TCP)
- alta latenza

- possibile caduta della connessione ssh

LATO CLIENT:
Prendere uno scriptino qui:
http://www.faqs.org/docs/Linux-mini/ppp-ssh.html
ed eseguirlo da root, o “suidato”.

SETUP LATO SERVER:
Sull'host remoto (firewall) bisogna permettere all'utente di eseguire pppd (sudo).
Per far questo servono i massimi privilegi (#).

Dopo aver avviato la connessione, bisogna impostare le rotte lato server, o l'ip forwarding, se necessario.

Esperienza personale:
Quando avevo l'utenza in fastweb era per me difficile utilizzare alcuni programmi che comunicavano bidirezionalmente in UDP. Questa era una soluzione non troppo invasiva.