Sommario

Introduzione

Il seguente articolo dovrebbe aiutare chiunque fosse interessato alla sicurezza nelle reti di calcolatori, infatti molte delle connessioni che troviamo in rete trasmettono e ricevono dati "in chiaro", questo è evidentemente un grosso rischio! Per questo motivo sono nati protocolli di criptazione e software in grado ti effettuare connessioni criptate, come ad esempio SSH.

Do per scontato che chiunque legga questo documento sappia cosa sia SSH (vedi " Come installare SSH su windows ") e VNC(per gli utenti windows vedi la guida a TightVNC), percui incominciamo con lo spiegare cosa è un *tunnel* .

Quando tra due computer c'é una connessione, ci sará uno scambio di dati, il quale puó essere criptato o meno.
Un "**tunnel**" é una connessione criptata, all'interno della quale é possibile far passare un'altra connessione che non lo é.
In questo articolo prenderemo in considerazione il protocollo SSH, per cui il flusso di dati in chiaro passerá dentro il tunnel SSH e potrá essere usato solo dai due pc all'estremitá della connessione, e mai dagli intermediari.

Il miglior modo per spiegare cosa sia un tunnel è fare un esempio pratico:

In un pc in rete,chiamato *remoto.pc.org* sono avviati sia il server ssh che quello vnc. Il firewall presente su questa macchina non consente l'accesso al server VNC [1] da remoto per motivi di sicurezza, cosa che generalmente è una buona idea.

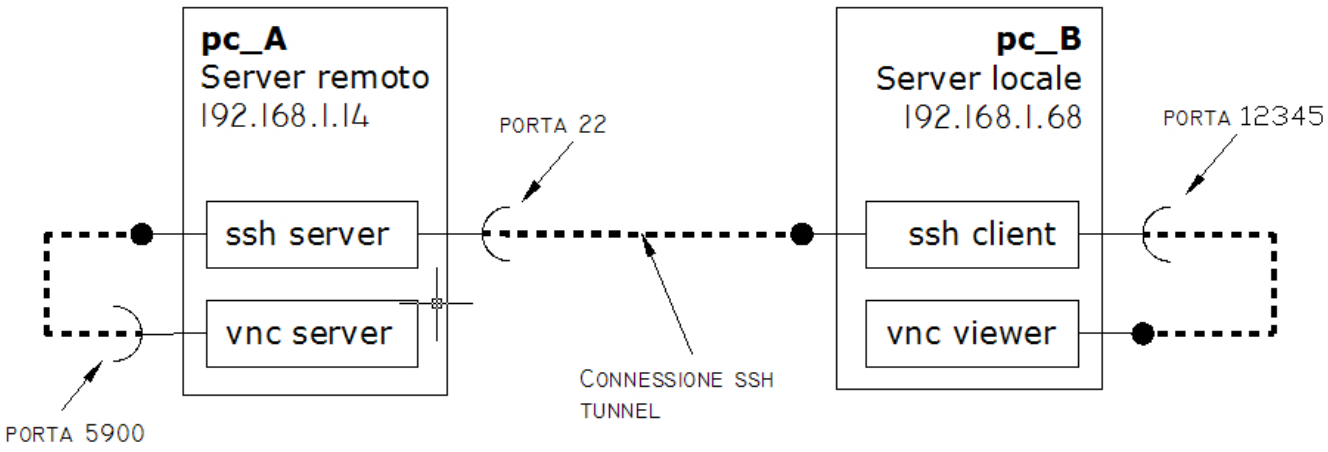
Il server vnc è in ascolto sulla porta 5901, mentre il server SSH sulla porta 22, che non viene blaccata dal firewall.

Noi a questo punto ci connetteremo al server VNC tramite una connessione SSH, usando una caratteristica del protocollo SSH chiamata "local port forwarding" o semplicemente **tunneling**.

Il port forwarding con SSH lo possiamo usare servendoci dell'opzione "**-L** [*bind_address*:]*port*:*h ost*
:
hostport
", con la quale si specifica la porta locale(
"port
") che verrà collegata tramite ssh alla porta remota("
hostport
").
Da questo momento, qualsiasi connessione fatta verso la porta locale specificata verrà trasmessa in automatico sulla porta remota.
Se per esempio usiamo il comando:

```
ssh -L 12345:hostremoto:5900 nomeutente@hostreamoto
```

verrà collegata la porta 12345 della macchina in cui lanciamo il client ssh alla porta 5900 di *hos tremoto*
attraverso un canale sicuro, cioè il client SSH si connetterà sempre alla porta 22 dell'pc remoto, ma all'interno scorrerà la nostra connessione 12345->5900. Un esempio pratico lo trovate in questo articolo
.



Ora che sappiamo usare SSH il nostro obiettivo sarà di aprire una connessione sicura con SSH e di riversare dentro quest'ultima un altra, quindi il flusso di dati vnc verrà criptato da ssh.

Il nostro tunnel dovrà essere quindi inizializzato:

```
[me@local]$ ssh -L 12345:remoto.pc.org:5900 me@remoto.pc.org
```

Dopo aver lanciato questo comando accade che:

1. Viene messo in ascolto una connessione SSH sulla porta 12345 del computer locale e collegata alla porta 5900 del pc remoto.
   La porta 12345 è solo un esempio e potrebbe esserci qualsiasi altra porta libera sul pc locale.
   Il risultato è che ogni richiesta fatta sulla porta 12345 verrà trasferita alla porta 5900 sul pc remoto e i dati saranno criptati ssh.

2. La connessione al server remoto SSH usa il nome utente "me", quindi è scontato che ci si possa connettere al server remoto con queste credenziali.

Per poterci collegare al server VNC remoto tramite il tunnel è necessario che la connessione SSH rimanga aperta e da un altro terminale locale è necessario connettersi alla porta locale 12345 con un client VNC.

```
[me@local]$ vncviewer 127.0.0.1:12345
```

Quello che risulterà , lo potrete verificare facendo un "*netstat*" sul server, è una connessione tra il server *remoto.pc.org* sulla porta 22 e il pc locale da cui è partito il comando su di una porta casuale.

Chiudendo il client vnc e facendo il logout dal server ssh il tunnel verrà distrutto.

Un modo migliore...

Questo metodo funziona ma non si può certo dire pratico, potremo migliorarlo mettendo in cascata i comandi da eseguire:

```
[me@local]$ ssh -f -N -L 12345:remoto.pc.org:5900 me@remoto.pc.org; vncviewer
127.0.0.1:25901:1
```

Fruttando le opzioni ***"-f -N"*** con le quali il programma ssh va in background un momento prima della esecuzione del programma.Queste due opzioni vanno impostate contemporaneamente perché se usassimo solo la "-f" dovremo inserire anche il comando da eseguire in remoto, mentre usando la "-N" noi eliminiamo quest'onere.

Naturalmente noi non potremo più usare la shell da cui abbiamo lanciato ssh per controllare il server remoto, ma, dato che con ssh dovevamo creare solo il tunnel, non dobbiamo inviare alcun comando e quindi ci va più che bene mandarlo in backgroud.

Il problema è che per terminare il programma ssh dovremo farlo manualmente. Infatti ache se chiudessimo la connessione vnc...il processo ssh rimarrebe sempre attivo.

Tunnel auto-terminante

Per superare il problema precedentemente esposto, ovvero che il programma rimane attivo anche quando non ne abbiamo più bisogno, è possibile ricorrere ad un trucchetto.

Se noi lanciassimo ssh con l'opzione "-f" dovremo dargli anche un comando da eseguire, a questo proposito potremo lanciare uno "sleep 10", che per 10 secondi metterebbe in pausa il processo, dopo di che questo riprenderebbe il suo normale funzionamento.

La scelta di usare sleep come comando è dovuta al fatto che:

1. Non consuma risorse
2. Possiamo impostarne la durata

Questo lo rende perfettamente adatto ai nostri fini.

Il processo SSH senza l'opzione "-N" a differenza di prima sarebbe creato in funzione dell'esecuzione dello *sleep* e non della creazione del tunnel quindi, dopo aver eseguito il comando, terminebbe dato che non avrebbe nulla da fare. Ma se in cascata faciamo passare nel tunnel il flusso dati di vncviewer allora il tunnel rimarrebbe aperto per tutto il tempo che la connessione vnc lo tiene occupato , dopo di che chiudendo il vncviewer terminerebbe anche il nostro tunnel.

Il comando da eseguire sul client è quindi:

[me@local]$ ssh -f -L 12345:127.0.0.1 [2] :5900 me@remoto.pc.org sleep 10; vncviewer 127.0.0.1:12345:1

Se vncviewer non venisse lanciato il tunnel verrebbe chiuso dopo 10 secondi , viceversa se vncviewer venisse lanciato esso dovrebbe aspettare 10 secondi di inattività dopo di che prenderebbe possesso del tunnel.

Alla fine chiudendo il vncviewer non ci dovrebbe rimanere traccia neanche del processo ssh.

Ulteriori accorgimenti

Se volessimo migliorare l'efficienza del tunnel potremo comprimere i dati specificandone anche il tipo di criptazione.

[me@local]$ ssh -C -c blowfish -f -L 12345:remoto.pc.org:5900 me@remoto.pc.org sleep 10; vncviewer 127.0.0.1:12345

Dove con "-C" si attiva la compressione dei dati e con "-c" si sceglie la codifica più o meno pesante che andrà ad influire quindi su "quanti dati vengono trasmessi". Nell'esempio riportato di sopra si è scelta la *blowfish* che è sufficiente per una normale criptazione dei dati.

Per ulteriori approfondimenti consiglio di leggere la guida per un tunnel SSH con Putty dove si spiega con un esempio pratico la procedura di creazione di un tunnel SSH, questo potrebbe essere gradito agli utenti windows abituati ad avere a che fare con le interfacce grafiche piuttosto che con shells.

Note

[1] Per gli utenti windows è possibile usare RealVNC che è semplicissimo da usare e configurare, vedi Guida RealVNC

[2]Potremo usare anche "*remoto.pc.org*" al posto di "*127.0.0.1*" , l'importante è capire che ci si riferisce al localhost del pc remoto a differenza del secondo "
127.0.0.1
" presente nel comando, il quale è invece riferito al local host sul client dove lanciamo il nostro viewer.