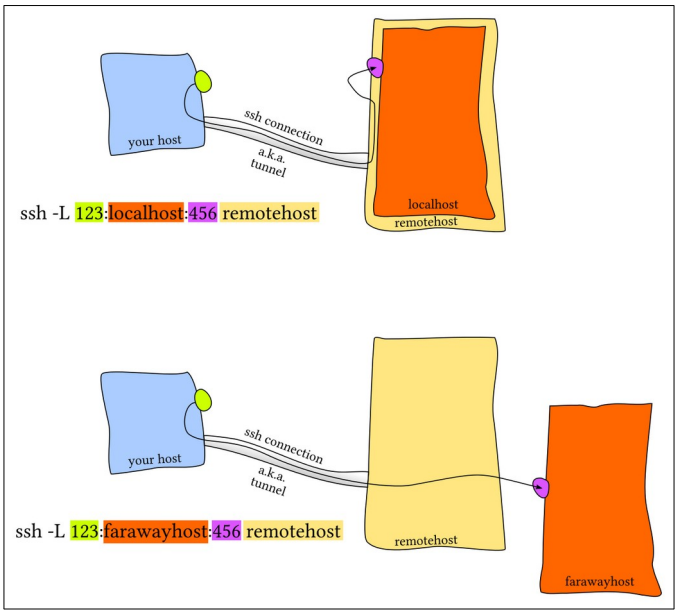
Túnels SSH

Una de les capacitats que aporta el protocol SSH és la del reenviament de connexions a un tercer, el qual es pot fer de diferents formes segons l'objectiu que volguem aconseguir. Més concretament, els tipus de reenviaments i les opcions que tenim per controlar-los són els següents:

* *LocalForward* (o paràmetre *-L*): Aquesta opció s'utilitza per establir una connexió que reenviarà el trànsit des d'un port local (client SSH) a la màquina remota (servidor SSH) per fer, a partir d'aquesta, un túnel cap a un tercer destí. Aquest reenviament és útil quan des de la màquina client no s'hi pot accedir directament al tercer destí (degut per exemple a l'acció d'un tallafocs) però sí s'hi pot accedir des d'un determinat servidor SSH que sí es públic. El valor d'aquesta opció/paràmetre té la forma *x:destiTercer:y* on "x" hauria de ser el port local al qual es vol dirigir el trànsit, "destiTercer" és l'adreça IP o nom DNS del destí de tercers i "y" és el seu port on es vol dirigir tot aquest trànsit.

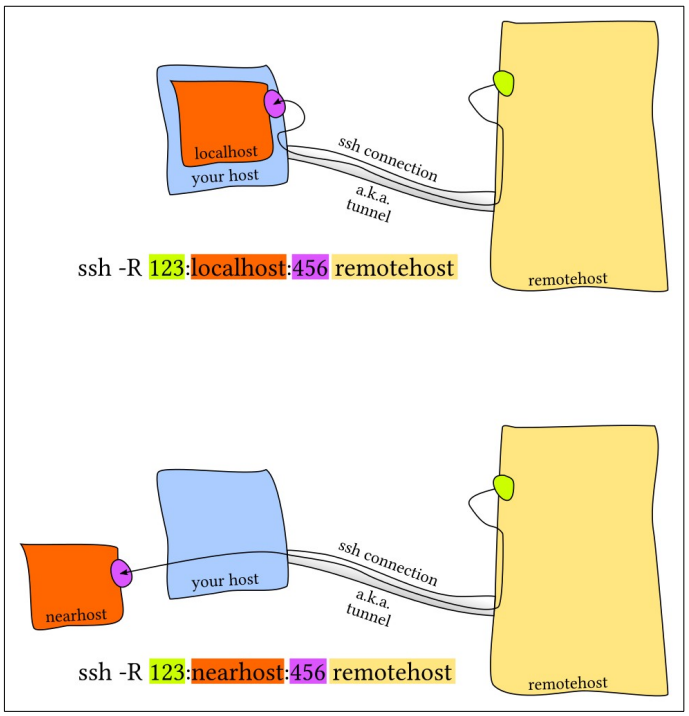
A continuació es mostra un diagrama on s'il·lustra el funcionament d'aquest tipus de túnel, indicant a més la comanda corresponent (on la caixa blava representa el client SSH, la groga el servidor SSH i la taronja el servidor remot de tercers, el qual pot funcionar, perfectament, a la mateixa màquina on s'executa el servidor SSH, podent així fer servir la paraula "localhost"). Un cop haguem establert el túnel en qüestió, per accedir al servidor de tercers només hauríem de fer servir el client adient (és a dir, un navegador si el servidor de tercers és un servidor web; un client VNC si és un servidor VNC, etc) fent- lo apuntar al port local (el marcat amb el color verd al diagrama anterior; aquest color indica el punt on ens connectarem i el color lila indica el punt on realment hi arribarem a través del túnel).



* *RemoteForward* (o paràmetre *-R*): Aquesta opció s'utilitza per establir una connexió que reenviarà el trànsit des d'un determinat port remot (servidor SSH), funcionant ara com a port client, a un tercer destí (emprant en aquest cas el client SSH com a "servidor" intermediari). Aquest reenviament és útil quan des de la màquina servidora no s'hi pot accedir directament a un tercer destí (degut a un tallafocs, normalment) però sí des del client SSH. El valor d'aquesta opció/paràmetre té la forma *x:destiTercer:y* on "x" hauria de ser el port del servidor SSH des del qual es vol dirigir el trànsit fins "destiTercer", que representa l'adreça IP o nom DNS del destí de tercers i "y", el seu port

A continuació es mostra un diagrama on s'il·lustra el funcionament del túnel "RemoteForward", indicant a més la comanda corresponent (on la caixa blava representa el client SSH, la groga el servidor SSH i la taronja el servidor remot de tercers, el qual pot funcionar, perfectament, a la mateixa màquina on s'executa el client SSH, podent així fer servir la paraula "localhost"). Un cop haguem establert el túnel en qüestió, per accedir al servidor de tercers només hauríem de fer servir el

client adient (és a dir, un navegador si el servidor de tercers és un servidor web; un client VNC si és un servidor VNC, etc) fent- lo apuntar al port remot (el marcat amb el color verd al diagrama anterior; aquest color indica el punt on ens connectarem i el color lila indica el punt on realment hi arribarem a través del túnel).



NOTA: A priori, hom podria pensar que per poder arribar al servidor de tercers caldria tenir activat l'"Ip-forwarding" al servidor SSH (en el reenviament *-L*) o al client SSH (en el reenviament *-R*) si aquest servidor de tercers estigués en una altra xarxa d'on estigués el client SSH (en el primer cas) o el servidor SSH (en el segon cas). No obstant, si el procés SSH en qüestió pot fer servir totes les tarjes del sistema (per defecte és així), ja se n'encarrega ell mateix de transferir les dades d'una a l'altra en aquest nivell 7 de forma autònoma sense que haguem de configurar res a més baix nivell. Cal dir, però, que això serà així sempre que la directiva *AllowTCPForwarding* del servidor SSH estigui establerta a *yes* (per defecte ja és així). De fet, aquesta característica ens podria estalviar moltes vegades haver d'implementar un DNAT al router per tal de publicar els diversos servidors interns de la LAN: només amb què estigui accessible el servidor SSH ja n'hi hauria prou.

* *DynamicForward* (o paràmetre *-D*): Aquesta opció és similar a la de *LocalForward* en el sentit de què permet establir un port local al client SSH per enviar-hi allà tot el trànsit de qualsevol altre programa client (navegador, etc) de forma que, mitjançant la intermediació del servidor SSH remot, aquest sigui reenviat a qualsevol destí de tercers. La diferència és, com s'ha marcat, que ara no hi ha un túnel amb una servei de tercers (IP i port) concret -i, per tant, no cal preocupar-se d'indicar-ne cap a l'hora d'implementar-ho- sinó que el túnel s'implementarà de forma dinàmica amb tots els destins on vulgui accedir el programa client (navegador, etc. Aquest túnel és interessant, doncs, quan volem accedir a destins (i ports) que poden ser canviants ("dinàmics"), i/o molts.

Tot això és possible gràcies a l'ús del protocol (de nivell 5-6) anomenat SOCKS5; és a dir, l'opció "DynamicForward" del client SSH converteix al servidor SSH en un servidor SOCKS (però amb l'afegit del xifrat que proporciona SSH). En general, la funció d'un servidor SOCKS és simplement la de fer d'intermediari a nivell de capa de transport entre un client i qualsevol altre servidor de tercers on aquest client es vulgui connectar; el fet de què s'implementi via SSH fa que, si més no, entre el programa client (que connecta amb el túnel dins de la seva pròpia màquina) i el servidor SSH remot el trànsit estigui xifrat; un cop surti aquest del servidor SSH, l'existència del xifratge ja dependrà de si les dades en sí ho són o no (HTTPS vs HTTP, per exemple)

NOTA: El protocol SOCKS permet reenviar qualsevol tràfic TCP, UDP i també ofereix la possibilitat d'autenticar

La manera d'implementar un servidor SOCKS5 amb SSH és executant la comanda *ssh -D nºportlocal usuari@nomServSSH* A partir d'aquí, qualsevol programa local que vulgui emprar aquest servidor SOCKS s'haurà de configurar convenientment. Per exemple, en el cas del Firefox, per fer-ho cal anar al panell "Preferències" i pulsar sobre el botó "Paràmetres" de l'apartat "Paràmetres de xarxa"; al quadre que apareixerà caldrà sel·leccionar l'opció "Configuració manual del proxy" i escriure, a l'apartat "Ordinador central SOCKS", la 127.0.0.1 i el port *nºportlocal* indicat a la comanda anterior (indicant la versió "Socksv5"). A partir de llavors, totes les connexions que vulgui fer el Firefox amb servidors HTTP/S remots passaran primer pel túnel SOCKS, sortint a l'"exterior", d'aquesta manera, pel servidor SSH remot (que és qui veurà el servidor de destí final). El Firefox és un exemple, però qualsevol altre client compatible amb SOCKS funcionaria de la mateixa manera, un cop configurat

NOTA: Segons l'explicat als paràgraf anteriors, és fàcil veure que implementar un servidor SOCKS via SSH pot tenir molts usos interessants. Per exemple, suposant que aquest servidor estigués en un país diferent, podria servir per saltar-se restriccions de tipus geogràfic. O si el tenim en qualsevol lloc, per no ser "espiat" a través de la Wifi d'un cibercafé, o per accedir a fitxers ubicats a la xarxa local del servidor SOCKS, etc.

NOTA: Una cosa a tenir en compte, però, és la resolució de noms. Segons l'aplicació client emprada i la configuració del servidor SOCKS, serà possible especificar si volem resoldre localment els noms DNS (i llavors indicar-li al servidor SOCK l'adreça IP del destí ja resolta o, per contra, delegar en el servidor SOCKS la resolució de noms per a què la faci ell (a vegades això és necessari si des de la xarxa on ens trobem no podem resoldre determinats noms de sistemes als quals volem accedir a l'altre costat del servidor SOCKS o bé si no volem que ens puguin espiar amb qui contactem), o provar els dos mètodes. Per exemple, Firefox té el paràmetre *network.proxy.socks_remote_dns* (accessible via *about:config*) que ens permet especificar que es resolgui remotament. Per defecte es resol localment.

NOTA: Les aplicacions client que no suportin SOCKS les podem "socksificar". Això consisteix en carregar una llibreria addicional que detecti peticions a la pila TCP/IP i modificar-les per redirigir-a través del servidor SOCKS, de manera que la comunicació vagi per ell sense que l'aplicació s'hagi hagut de programar específicament amb suport de SOCKS. Les utilitats "socksificadores" més conegudes són **tsocks** (http://tsocks.sourceforge.net) i **proxychains** (http://proxychains.sourceforge.net). Ambdues funcionen de forma molt semblant: només cal llançar l'aplicació que volem "socksificar" precedint-les d'elles i ja està. Per exemple, si volem "socksificar" la comanda "wget", podem executar simplement *proxychains wget http://una.url.com* Per a què això funcioni, no obstant, cal haver configurat "proxychains" per dir-li a quin servidor SOCKS volem que es connecti; això ho podem fer a l'arxiu "/etc/proxychains.conf" amb les següents línies (suposant que tenim el client *ssh -D 5555* ... ja funcionant):

```
[ProxyList]
SOCKS5 127.0.0.1 5555
#També li podem dir que les peticions DNS les faci el servidor SOCKS en comptes de fer-ho
#localment amb la següent línia
proxy_dns
#Per no mostrar els missatges informatius que treu el proxychains
quiet_mode
```

Molt sovint, a l'hora de realitzar túnels (de qualsevol dels tipus anteriors) s'hi afegeix a la comanda *ssh* indicada els paràmetres *-f* (equivalent a la directiva de client *ForkAfterAuthentication yes*) i *-N* (equivalent a la directiva de client *SessionType none*): el primer paràmetre serveix per posar la comanda *ssh* en segon pla (i així deixar el terminal lliure...per trencar el pont caldrà llavors matar el procés fent *killall -s 9 ssh* o similar); el segon serveix per indicar que no es vol fer servir la comanda *ssh* per obrir cap terminal (sinó per, en aquest cas, només reenviament de connexions).

Si no volem, però, que el nostre servidor SSH pugui fer-se servir com a servidor de reenviament/SOCKS, caldria deshabilitar explícitament aquesta funcionalitat (per defecte està activada). Concretament, la directiva de servidor *AllowTcpForwarding no* desactivaria qualsevol tipus de redireccionament TCP, mentre que la directiva de servidor *DisableForwarding yes* és més radical, ja que deshabilitaria qualsevol altre tipus de redireccionament (no només TCP sinó també d'altres que no hem estudiat, com ara el de "sockets" Unix o el de tipus "X11", entre altres). Si el que volguéssim, però, no fos desactivar el redireccionament per complet sinó només limitar-lo a certs destins es pot usar, enlloc de les anteriors, la directiva *PermitOpen destiTercer:y* (en el cas del de tipus *LocalForward*) o la directiva *PermitListen destiTercer:y* (en el cas del de tipus *RemoteForward*); aquestes dues darreres directives fins i tot es poden incloure en una secció *Match {User|Address|...}* per restringir (encara més) el seu efecte només als elements indicats.

EXERCICIS:

0.-Assegura't que tinguis una màquina virtual amb dues tarjes de xarxa: una en mode adaptador pont i una altra en mode xarxa interna. Clona-la de forma enllaçada i arrenca totes dues. A la primera (l'anomenarem "MaquinaA") instal·la el servidor SSH i a la segona (l'anomenarem "MaquinaB") instal.la el servidor Apache. Finalment, configura (com et sigui més còmode: via *ip address add*, via *systemd-networkd*, etc) les tarjes en mode xarxa interna de les dues màquines per a què tinguin cadascuna una IP de la xarxa 10.0.0.0/8 (en aquest cas suposarem que "MaquinaA" tindrà la IP 10.0.0.1/8 i "MaquinaB" tindrà la IP 10.0.0.2/8). Assegura't també que el tallafocs de les dues màquines estigui aturat (*systemctl stop firewalld/ufw*)

Túnel directe: ***quan des del client no es té accés a un servidor de tercers (Apache) però des del servidor SSH sí***

1.-a) Executa a la màquina real la comanda *ssh -fNL 5555:10.0.0.2:80 usuari@192.168.12.x* (on "192.168.12.x" representa la IP de la tarja en mode adaptador pont de "MaquinaA" que tingui en aquest moment) i tot seguit, a la màquina real també, obre un navegador i escriu la URL *http://127.0.0.1:5555* ¿Què veus? ¿Per què?

b) ¿Què mostra la comanda *ss -tn "dport = :22"* executada a la màquina real? ¿I la comanda *ss -tln "sport = :5555"* , també executada a la màquina real? ¿Què mostra la comanda *ss -tn* executada a "MaquinaA"? ¿I la mateixa comanda executada a "MaquinaB"? (refresca la pàgina al navegador si cal recarregar les connexions) En aquest sentit, ¿quina adreça IP d'origen (és a dir, del client) veus que es registra a l'arxiu "/var/log/apache2/access.log" (o "/var/log/httpd/access_log" en Fedora)?

> NOTA: El nostre client SSH pot donar servei a altres màquines per a què facin servir el túnel establert. Per això només cal posar a escoltar el port local a totes les IPs (en comptes de la IP 127.0.0.1, que és el que fa per defecte). Això s'aconsegueix executant la comanda *ssh -fNL* ***0.0.0.0:****5555:10.0.0.2:80 usuari@192.168.12.x* (on en comptes de "0.0.0.0" es podria haver indicat l'adreça IP concreta d'alguna de les tarjes del client SSH, si en volguéssim restringir l'escolta a una tarja en particular) D'aquesta manera, en una altra màquina es podria obrir el navegador i escriure la URL http://ip.client.ssh:5555 per accedir al servidor Apache de "MaquinaB". Una altra manera d'aconseguir el mateix seria, en compte d'haver d'escriure la IP assenyalada en negreta, afegir el paràmetre ***-g***, així: *ssh -gfNL 5555:10.0.0.2:80 usuari@192.168.12.x*

Túnel invers: ***quan des del servidor SSH no es té accés a un servidor de tercers (Apache) però des del client sí)***

2.-a) Mata el túnel establert (fent *killall -s 9 ssh* a la màquina real) i desfés també la connectivitat entre "MaquinaA" i "MaquinaB" per a què no es puguin fer "ping" entre elles (pots fer simplement *sudo ip link set down dev enp0s8* en cadascuna d'elles)

b) Executa a la màquina real la comanda *ssh -fNR 5555:192.168.12.y:80 usuari@192.168.12.x* (on "192.168.12.x" representa igualment la IP de la tarja en mode adaptador pont de "MaquinaA" que tingui en aquest moment mentre que "192.168.12.y" representa la IP de la tarja en mode adaptador pont de "MaquinaB"). Tot seguit, a "MaquinaA", executa la comanda (equivalent a obrir un navegador, per si no en teniu cap a mà en estar usant un sistema sense escriptori) *curl http://127.0.0.1:5555* . ¿Què veus? ¿Per què?

c) Executa en un terminal de "MaquinaA" el següent (per mantenir activa la comanda *curl* fent peticions sense parar: *while [[true]]; do curl http://127.0.0.1:5555; done* Tot seguit, respon: ¿què mostra la comanda *watch -n 1 ss -tn* executada a "MaquinaA"? ¿I la mateixa comanda executada a "MaquinaB"? ¿I la comanda *watch -n 1 ss -tn "dport = :80"* executada a la màquina real? ¿Quina serà l'adreça IP d'origen (és a dir, del client) que es registrarà a l'arxiu "/var/log/apache2/access.log" (o "/var/log/httpd/access_log" en Fedora)?

> NOTA: El servidor SSH pot donar servei a altres màquines per a què facin servir el túnel establert com a clients "extra". Per això només cal posar a escoltar el port-punt-dentrada remot a totes les IPs (en comptes de la IP 127.0.0.1, que és el que fa per defecte). Això s'aconsegueix executant la comanda *ssh -fNR* ***0.0.0.0:****5555:192.168.12.y:80 usuari@192.168.12.x* (on en lloc de "0.0.0.0" es podria haver indicat l'adreça IP concreta d'alguna de les tarjes del servidor SSH, si en volguéssim restringir l'escolta a una tarja en particular) D'aquesta manera, en una altra màquina de la seva xarxa es podria obrir el navegador i escriure la URL http://ip.*servidor.ssh:5555* per accedir al servidor Apache de la màquina real. No obstant, això no funcionarà (per motius de seguretat) fins que a la configuració del servidor ***GatewayPorts clientspecified***

Túnel SOCKS:

3.-a) Mata el túnel establert (fent *killall -s 9 ssh* a la màquina real) i apaga "MaquinaB" (no la farem servir de moment més)

b) Executa a la màquina real, la comanda *ssh -fND 5555 usuari@192.168.12.x* (on "192.168.12.x" representa la IP de la tarja en mode adaptador pont de "MaquinaA" que tingui en aquest moment). Tot seguit configura el Firefox de la màquina real tal com diu la teoria i, finalment, escriu a la seva barra de direccions qualsevol URL del món. ¿Què veus? ¿Per què? ¿Què veus si executes a "MaquinaA" la comanda *ss -tn*?

c) Desfés la configuració del Firefox (no la tornarem a fer servir). Finalment, mantenint el túnel anterior, executa ara a la màquina real la comanda *curl -L -x socks5://127.0.0.1:5555 www.hola.com* ¿Què veus? ¿I què veus ara si trenques el túnel i tornes a executar la mateixa comanda?

> NOTA: Si s'especifica el protocol *socks5h://* en comptes de *socks5://*, la resolució de noms la farà el servidor SOCKS en comptes de fer-ho localment
> NOTA: El nostre client SSH pot donar servei a altres màquines per a què facin servir el túnel establert. Per això només cal posar a escoltar el port local a totes les IPs (en comptes de la IP 127.0.0.1, que és el que fa per defecte). Això s'aconsegueix executant la comanda *ssh -fND **0.0.0.0:**5555 usuari@192.168.12.x* (on en comptes de "0.0.0.0" es podria haver indicat l'adreça IP concreta d'alguna de les tarjes del client SSH, si en volguéssim restringir l'escolta a una tarja en particular) D'aquesta manera, en una altra màquina es podria obrir el navegador i escriure la URL http://ip.client.ssh:5555 per gaudir del servidor SOCKS
> NOTA: A l'exercici anterior s'ha utilitzat el servidor SOCKS per accedir a un servidor de tercers de tipus HTTP, però cal recordar que el servidor de tercers pot ser de qualsevol tipus!

"Bastion hosts":

S'anomenen "bastion hosts" a servidors assegurats que funcionen de cara al públic i que ofereixen un punt d'entrada al sistemes que hi ha darrera ells (en ubicacions més restringides, darrera un tallafocs, etc). La comanda *ssh* permet, gràcies al paràmetre *-J* (o la directiva equivalent *ProxyJump*)**,** accedir a un servidor SSH intern a través d'un "bastion host" SSH intermediari que ho permeti (exercint de "trampolí") Concretament, si executem *ssh -J usuari1@ip.bastion.host usuari2@ip.servidor.intern* podrem accedir al servidor SSH intern (el qual tindrà una adreça IP privada) connectant al "bastion host", que és, de fet, l'únic servidor disponible públicament. En realitat, el que fa aquesta comanda és simplement iniciar sessió SSH primer en el bastió, i un cop allà, iniciar sessió SSH al servidor intern.

> NOTA: Si el bastió i/o el servidor intern escolten en un port diferent del 22, es pot indicar aquest port després d'escriure la seva IP o nom (i sense deixar cap espai) amb el sufixe ":nºport"
> NOTA: Es pot indicar més d'un bastió si s'especifiquen entre comes. Per defecte, per cada bastió es demanarà la contrasenya de l'usuari corresponent, fins demanar l'usuari del servidor SSH final però això es pot canviar si s'implementa l'autenticació per claus, de manera que només se'n pregunti la contrasenya del servidor SSH final, o només els dels bastions, o cap ni una.
> NOTA: La directiva *ProxyJump* és una alternativa més moderna i simplificada a l'ús combinat de la directiva (més general) *ProxyCommand* combinada amb el paràmetre *-W* del client ssh

4.- Arrenca "MaquinaB" de nou i assegura't de què torni a tenir la IP 10.0.0.2/8 a la seva tarja en mode "xarxa interna". Des de la màquina real executa la comanda *ssh -J usuari@192.168.12.x usuari@10.0.0.2* (on "192.168.12.x" representa la IP de la tarja en mode adaptador pont de "MaquinaA" que tingui en aquest moment). ¿Què passa, on hi accedeixes? ¿Què es veu a la sortida de la comanda *ss -tn* de "MaquinaA"? ¿Què passa si, tenint la sessió SSH iniciada, atures "MaquinaA"?