

Advanced SSH

Hans Peter Verne

Geofag/UIO

2016-11-11

This presentation is an adaption from
<http://www.mn.uio.no/geo/english/services/it/help/using-linux/ssh-tips-and-tricks.html>

- Drop-in replacement for rsh/rcp (from the 80's)
- To log in:
`$ ssh username@remote.host`
- To log in with X11 forwarding and Compression:
`$ ssh -X -C remote.host`
- To copy a local file to the remote host:
`$ scp localfile remote.host:remotefile`

Note the use of \$ as a shell prompt, and remote.host etc. as a placeholder.

Disclaimer: I'm not a cryptographer.

All data can be assumed to be (sequences of) numbers.

- Function f is used to encrypt a message to crypto-text:

$$c = f_k(m)$$

- The inverse function g is used to decrypt:

$$m = g_k(c)$$

- The functions f and g are not secret.
- The index k is secret. We call it a *key*.
- k is typically 128–256 bits (ca. $10^{38} - 10^{77}$).

Historical example, the Caesar shift

- Each letter in m is shifted n places up the alphabet.
- key is n .
- pseudocode, excluding rollover:
 $f(m,n) = \text{foreach char } x \text{ in } m \{ \text{print } (x + n) \}$
 $g(c,n) = \text{foreach char } x \text{ in } c \{ \text{print } (x - n) \}$
- rot13

- Two functions with different index:

$$c = q_j(m)$$

$$m = p_i(c)$$

- Functions are not secret.
- Large indices (i, j) , typically 2048 bits (ca. 10^{616}).
- The indices (i, j) can be generated as a pair,
but you *cannot compute one from the other!*

Alice and Bob



- Alice wants to send secret messages to Bob.
- We suspect that some third party will listen.
- We suspect that some third party will try to impersonate Alice.
- (or Bob).

Remember: $c = f_k(m)$ and $m = g_k(c)$

- Alice and Bob needs a secret key k .
- They want a one-time session key.
- They can't send the key in the clear...

Session key negotiation (simplified)

Remember: $m = p_i(q_j(m))$ (asymmetric cryptography)

- Alice calls up Bob: «Hi, it's me, Alice!»
- Bob generates a key pair (i, j) , sends Alice j .
- Bob: «Say Alice, can you choose a random session key k , compute $K = q_j(k)$, and send the result K back to me?»
- Only Bob knows i and can decrypt the result, $k = p_i(K)$.
- Both Alice and Bob now have a session key k .

(ssh uses a more complex algorithm, but the idea is similar.)

Remember: $c = f_k(m)$

- Bob can't know yet if he's really talking to Alice.
- Alice can now use k and encrypt any message and send to Bob.
- ... for instance her password.
- But Alice can do better!

Remember: $m = p_i(q_j(m))$

- Alice generates a set of (i, j) .
- She calls j her *private key*, and i her *public key*.
- She keeps her private key (j) secret!
- She hands her public key (i) to Bob.

Remember: $m = q_j(p_i(m))$

- Alice calls up Bob:
«Hi, it's me, Alice!»
- How can Bob know it's really Alice?
- Bob picks a random number r , computes $R = p_i(r)$.
- Bob presents a challenge:
«Hi, Alice. Here's a number R . Please compute $g_j(R)$?»
- Alice computes $r' = g_j(R)$, sends it back to Bob.
- If $r' = r$, Bob knows he's talking to Alice.

Enough math!

- `ssh-keygen`
- Creates a key pair, stores them in
 `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`
- Text files!
- The private key *must* be protected by a passphrase.

Putting it all together, cont.

- `$ scp .ssh/id_rsa.pub remote.host:tmp.key`
- `remote$ cat tmp.key >> .ssh/authorized_keys`
- `remote$ chmod go-rwx .ssh/ .ssh/authorized_keys`

Now try

- `$ ssh remote.host`
- Enter passphrase for key (...):

Solution: Load the key in to memory!

The ssh agent should hold the key in memory.

Do we have a running agent?

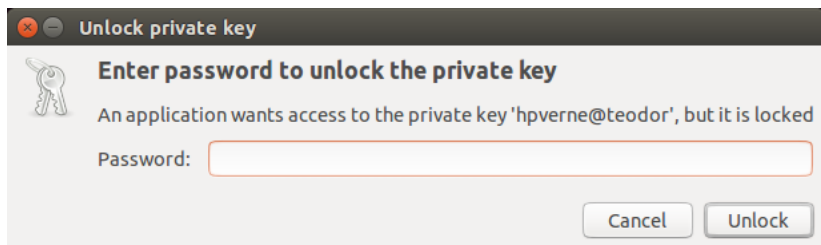
- `$ ssh-add -l`

Possible replies:

- 2048 4d:d0:c7:5e:(...) hpverne@tai.uio.no (RSA)
 - OK!
- The agent has no identities.
 - The agent is running, but has no keys.
- Could not open a connection to (...) agent.
 - The agent is not running.

The agent II

In a desktop environment (Gnome/KDE/Mac), you probably already run a keyring manager. Just run `ssh` to upload the key.



(The «Password» prompt is misleading. Type your key passphrase here.)

The agent III

- Otherwise, to start the agent, run:
\$ eval `ssh-agent`
- and, to upload the key:
\$ ssh-add

- You forgot the passphrase for the private key. What to do?
- Your private key is compromised! What must be done?
- *generate a new key pair, replace in .ssh/authorized_keys.*
- Under what circumstances would an unencrypted private key be acceptable?
- *Whole-disk encryption? Also, depends on how the key is used.*

- `$ ssh -A remote.host`
- `remote$ ssh different.host`

The ssh server (sshd) will act as an agent.

Only use it when you trust root on `remote.host`.

- Private/public key pairs, used to identify hosts.
- Public key transferred at first connection.
- Messy when machine is reinstalled!

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```

Download putty from

<http://www.chiark.greenend.org.uk/%7Esgtatham/putty/>

- puTTY : ssh client and terminal emulator.
- PSCP : secure copy
- PuTTYgen : Key generator (or just use Unix ssh-keygen).
- Pageant : ssh agent.

HOWTO:

https://www.howtoforge.com/ssh_key_based_logins_putty

- Run a remote command:

```
$ ssh remote.host "ls"
```

- Capturing output:

```
$ ssh remote.host \  
"cd /somedir/ ; tar cf - ." | tar xvf -
```

- Sending input:

```
$ tar cf - . | ssh remote.host \  
"cd /somedir/ ; tar xvf -"
```

Efficient copying of large datasets

- `$ rsync -av remote.host:/some/directory/ /target/`
- `$ rsync -av /some/directory/ remote.host:/target/`

Note the slash (/) on the end of `/some/directory/`. See the manual for `rsync(1)` for details.

Using ssh to mount a remote directory.

- Filesystem in userspace (fuse)
- `$ mkdir mountpoint`
- `$ sshfs remote.host:/directory/ mountpoint/`
- `$ fusermount -u mountpoint/`

Do not use a mountpoint on some network drive
(like your home directory)!

To access a remote TCP service, you need to know the port number. (Often implicit, or listed in `/etc/services`.)

Some commonly-used ports:

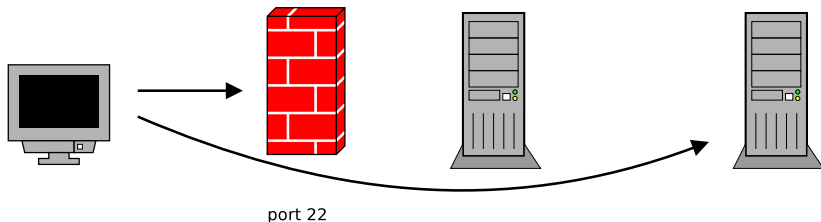
Service	Port	Description
http	80	Hyper-Text Transfer Protocol, ordinary web access
https	443	Secure HTTP
ssh	22	Secure shell
rdp	3389	Remote Desktop Protocol
pop3	110	Post Office Protocol (version 3)

Sometimes you can specify a different port, e.g.

```
$ xfreerdp -t 4000 server
```

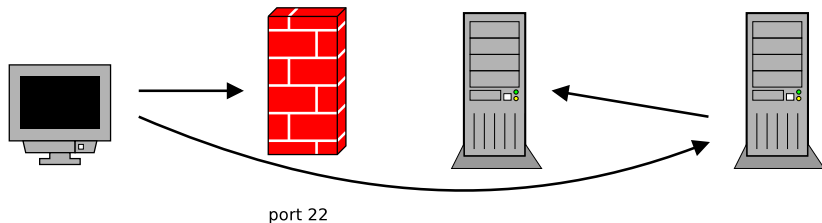
Firewalls: The server (or the network equipment) might deny the connection, based upon the IP address (network) of the client.

SSH traffic is usually not restricted by firewalls.



ssh tunnel : port forwarding

We want to find a way to let ssh forward the traffic to other ports:



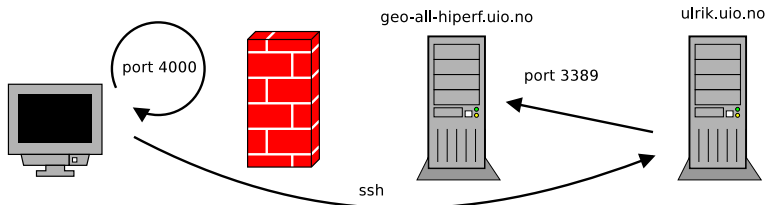
ssh tunnel setup

- Log in to ulrik with ssh, ask ssh to forward local port 4000 to port 3389 («remote desktop») on geo-all-hiperf.uio.no:

```
$ ssh -L 4000:geo-all-hiperf.uio.no:3389 ulrik
```

- Now, in an other shell (terminal window):

```
$ xfreerdp -t 4000 -u username localhost
```



Caveat : services which do host authentication will protest!

```
$ xfreerdp -t 4000 -u hpverne localhost
connected to localhost:4000
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: CERTIFICATE NAME MISMATCH! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The hostname used for this connection (localhost)
does not match the name given in the certificate:
geo-all-hiperf.uio.no
```

You must be root on the local machine to forward ports < 1024 .

```
$ ssh -L 80:www.uio.no:80 ulrik.uio.no
```

Privileged ports can only be forwarded by root.

Java apps in the browser will likely not work.

Multiple port forwarding

- `$ ssh -L 8000:www.somewhere.no:80 \`
`-L 4430:www.somewhere.no:443 ulrik.uio.no`
- `http://localhost:8000/`
- `https://localhost:4430/`

Links within the document will perhaps not work...

Better solution: Set up ssh as a SOCKS5 proxy:

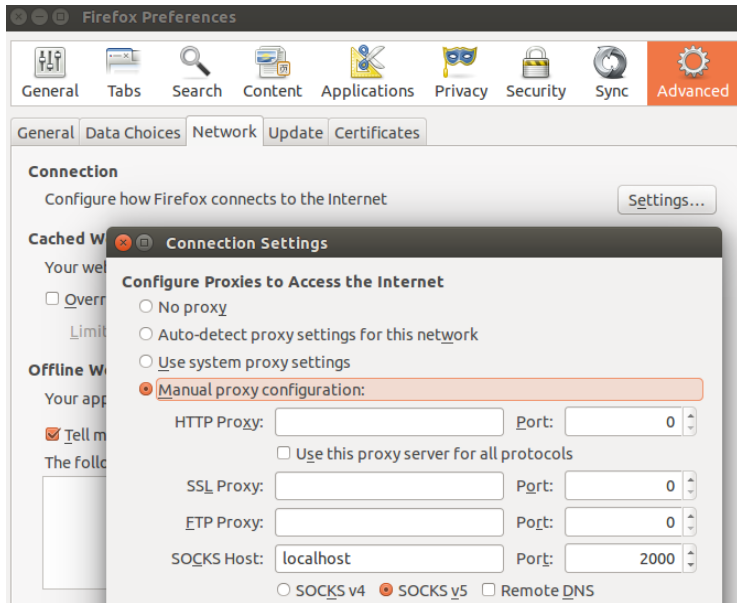
- `$ ssh -D2000 ulrik.uio.no`

You must configure your client to use port 2000 on localhost as SOCKS5 proxy.

Google Chrome:

```
$ google-chrome -proxy-server="socks5://localhost:2000"\  
-host-resolver-rules="MAP * 0.0.0.0 , EXCLUDE localhost"
```

Firefox SOCKS5 config



- The manual page, `ssh(1)`
- `.ssh/config`, `ssh_config(5)`
- Control the usage of keys in `~/.ssh/authorized_keys`
- `ssh ControlMaster`, multiple ssh logins in one connection.

Thank you!

- Questions?
- Discussion?
- Other topics?