Using SSH tunnels

In some cases you may need to access remote hosts that have private IP addresses or are protected behind a firewall. These hosts might be only accessible through an intermediate login host. The SSH tunnels (SSH port forwarding) facilitate access to multiple hosts protected behind a single public host exposed to the Internet. Once the tunnels are established the hosts behind the firewall can be directly accessed from your machine. Examples where tunnels can be beneficial:

- When using the INCD cloud service you may have multiple VMs with private IP addresses acessible through a single front-end VM having a single public IP address. This enables you to access remote VMs with private addresses and save scarse IPv4 address space.
- Some INCD services are not directly exposed to the Internet and may require passing through an intermediate login host for added security.

In the following examples the hostnames *public.ncg.ingrid.pt* and *internal.ncg.ingrid.pt* as well as the port numbers are placeholders that should be replaced by actual real hostnames and port numbers according to your scenario.

jump over SSH hosts easily from the command line

- If you just need to connect via SSH by jumping over a reachable host you can use the jump host functionality of SSH whose syntax is described below and where:
    - **internal-remote-host** is the host or IP address behind the firewall that you want to reach via SSH.
    - **public-remote-host** the remote host or IP address of the intermediate host that is exposed to the Internet through which you will connect to the **internal-remote-host**.
    - The username is optional and can be different for the public and internal host.

Basic jump host syntax:

```
ssh -J username@public-remote-host   username@internal-remote-host
```

Example of using the SSH jump host functionality to access the SSH port (port number 22) of the remote host *internal.ncg.ingrid.pt* through a publicly reachable host *public.ncg.ingrid.pt*. Most of the examples in this page are valid both for ssh and its related commands such as scp and sftp.

```
ssh -J public.ncg.ingrid.pt  username@internal.ncg.ingrid.pt

scp -J public.ncg.ingrid.pt  mylocalfile  username@internal.ncg.ingrid.pt:

sftp -J public.ncg.ingrid.pt username@internal.ncg.ingrid.pt


ssh -J username@public.ncg.ingrid.pt  username@internal.ncg.ingrid.pt
```

forwarding of X11 windows

- SSH enables the forwarding of X windows.
- This requires that your local machine has an X11 display server. If you are on Windows you can use **MobaXterm** that works both as ssh client and X server.
- For more information on this topic see **here**

port forwarding from local to remote in detail

- Port forwarding enables multiple tunnels over a single SSH connection thus exposing multiple remote hosts and ports to be directly accessed by SSH-related and SSH-unrelated applications (such as forwarding HTTP/HTTPS ports over SSH).
- The port forwarding is usually defined upon the connection to the intermediate host that is directly acessible. An SSH session to the intermediate host can simultaneously define one or more port forwardings using the **-L** option.
- The basic syntax for SSH port forwarding is described below where:
  - **local-port** is a TCP port of your local machine (your desktop, laptop, etc). Just pick one port that is not being used. This port should be above the privileged port range (above 1024) and preferrably outside of the IP ephemeral port range to avoid collisions with dynamically allocated ports (for many Linux systems select a port outside of the range 32768-60999).
  - **internal-remote-host** is an IP address or hostname of a remote host behind the firewall.
  - **internal-remote-port** is the TCP port of the internal-remote-host that you want to access.
  - **public-remote-host** the remote host or IP address of the intermediate host that is exposed to the Internet.

Basic port forwarding syntax:

```
ssh -L local-port:internal-remote-host:internal-remote-port   public-remote-host
```

Example of establishing a tunnel to access the SSH port (port number 22) of the remote host *internal.ncg.ingrid.pt* through a publicly reachable host *public.ncg.ingrid.pt*. We pick a random *local-port* (e.g. 31732) that is then mapped to the port 22 of *internal.ncg.ingrid.pt*.

```
ssh -L 31732:internal.ncg.ingrid.pt:22   public.ncg.ingrid.pt
```

Once the SSH connection to *public.ncg.ingrid.pt* is established the host *internal.ncg.ingrid.pt* can be directly accessed from your local host by connecting to the *local-port* on the loopback address. Just create a second command line terminal and try:

```
ssh -p 31732 username@127.0.0.1
scp -P 31732 mylocalfile  username@127.0.0.1:
sftp -P 31732 127.0.0.1
```

Once the SSH connection to the intermediate host is closed the tunnels will stop working. Therefore the initial SSH connection must be kept alive while needed and the connections to the remote internals hosts must be performed from a separate local terminal window.

Another example where both port 22 (SSH) and port 80 (HTTP) are both mapped to local ports. With this both ports are forwarded over a single SSH connection. The access to the web server from the local host can be tested using curl.

```
ssh -L 31732:internal.ncg.ingrid.pt:22 -L 8080:internal.ncg.ingrid.pt:80  public.ncg.ingrid.pt


curl http://127.0.0.1:8080
```

port forwarding from remote to local in detail

- This enables a remote host behind the firewall to access services in your local network. Basically this enables port forwarding in the reverse direction, from the remote network to the local network.
- This can be potentially dangerous for you as it enables users in the remote machine or remote network to access your local machines.
- The basic syntax for port forwarding from the remote side to the local side is described below where:

  - **remote-port-on-public-host** is a port number from the *public-remote-host* Just pick one port that is not being used above the privileged port range (above 1024) and preferrably outside of the IP ephemeral port range to avoid collisions with dynamically allocated ports (for many Linux systems outside the range of 32768-60999).
  - **local-host** a host in the local network of the user to be accessed from the *public-remote-host*.
  - **local-port** the port number of the *localhost* to be accessed from the *public-remote-host*.
  - **public-remote-host** the remote host or IP address of the intermediate host that is exposed to the Internet.

Basic remote port forwarding syntax:

```
ssh -R remote-port-on-public-host:local-host:local-port   public-remote-host
```

Example of forwarding the *remote-port* number 65532 on *public.ncg.ingrid.pt* to the port 22 of the local host named *myotherlocalhost*.

```
ssh -R 65532:myotherlocalhost:22   public.ncg.ingrid.pt
```

Then from *public.ncg.ingrid.pt* you can access myotherlocalhost in your local network with:

```
ssh -p 65532 127.0.0.1
```

jump host via config file

- Configuring SSH via jump host in your $HOME/.ssh/config enables the configuration to be stored. The configuration syntax is below where:
  - **Host** is a name choosen by you to identify this mapping, this name may be different from the hostname and will be recognized as a hostname only by SSH.
  - **Hostname** the real hostname of the internal host that you want to access behind the firewall.
  - **User** the username for the internal host identified by *Hostname*.
  - **ProxyJump** the remote host or IP address of the intermediate host that is exposed to the Internet.

The basic configuration for remote access via a jump host

```
Host name-for-the-internal-remote-host
Hostname actual-internal-remote-host
User username-in-actual-internal-remote-host
```

```
ProxyJump username@public-remote-host
HostKeyAlias name-for-the-internal-remote-host
```

Example of configuration:

```
Host ncg-internal
Hostname internal.ncg.ingrid.pt
User username-for-internal
ProxyJump username@public.ncg.ingrid.pt
HostKeyAlias ncg-internal
```

Accessing the configured host:

```
ssh ncg-internal
```

port forwarding via config file

- Configuring port forwarding in your $HOME/.ssh/config requires configuring two hosts. The first will be the intermediate host accessible through the Internet and the second the internal host. The fields are as follows:
    - **public-remote-host** the remote host or IP address of the intermediate host that is exposed to the Internet.
    - **LocalForward** the mapping of a *local-port* to a remote internal hostname or IP address and port number that are behind the firewall.
    - **name-for-the-internal-remote-host** is a name choosen by you to identify this mapping, this name may be different from the actual hostname and will be recognized as a hostname only by SSH.
    - **local-port** must be the same across the two host definitions and is a local TCP port number that will be mapped to the *internal-remote-host* and *internal-remote-port*.

The basic configuration for port forwarding:

```
Host public-remote-host
LogLevel FATAL
LocalForward local-port internal-remote-host:internal-remote-port


Host name-for-the-internal-remote-host
```

```
Hostname 127.0.0.1
User username
Port local-port
HostKeyAlias name-for-the-internal-remote-host
```

Example of port forwarding for both SSH (port 22) and HTTP (port 80):

```
Host public.ncg.ingrid.pt
LogLevel FATAL
LocalForward 31732 internal.ncg.ingrid.pt:22
LocalForward 8080  internal.ncg.ingrid.pt:80

Host ncg-internal
Hostname 127.0.0.1
User username
Port 31732
HostKeyAlias ncg-internal
```

Accessing the configured host

```
ssh ncg-internal
curl http://127.0.0.1:8080
```

(*) notice that you cannot use the hostname *ncg-internal* with curl as this is a hostname that is only recognized by SSH related applications (ssh, scp, sftp).

using socks proxies

- SSH supports socks4 and socks5 proxying. This allows socks enabled applications to access hosts behind the firewall. This is only valid for socks enabled applications e.g. firefox.
- The syntax is below where:
  - **public-remote-host** the remote host or IP address of the intermediate host that is exposed to the Internet
  - **local-port** as in the other cases is a TCP port of your local machine (your desktop, laptop, etc). Just pick one port that is not being used. This port should be above the privileged port range (above 1024) and preferrably outside of the IP ephemeral port range to avoid collisions with dynamically allocated ports (for many Linux systems select a port outside of the range 32768-60999).

The basic command line usage to establish a socks server via SSH:

```
ssh -D local-port  username@public-remote-host
```

The basic $HOME/.ssh/config setup to establish a socks server:

```
Host public-remote-host
DynamicForward local-port
User username
```

Example of socks proxy use with Firefox:

```
ssh -D 31733 username@public.ncg.ingrid.pt
```

Then in firefox:

1. Goto *Preferences->Network Settings*:
2. Select *Manual proxy configuration*
3. Enter in *SOCKS Host*: 127.0.0.1
4. Enter in *Port*: 31733
5. If needed add local networks to be excluded from the proxy using *No proxy for*
6. If needed add the options related to DNS proxying

(*) notice that all your http request will now be forwarded through the socks proxy, to restore the previous settings choose *No proxy* in the *Preferences->Network Settings*.

(*) as in the other examples the TCP port number to be allocated might need to be adjusted to match a free port number in your local machine (desktop, laptop, etc).

using sshuttle

The tool sshuttle is available in many Linux distributions. It uses SSH in combination with iptables and other functionalities to easily establish tunnels with better performance and a behavior more similar to a VPN. Instead of establishing a direct end-to-end TCP connection to the remote internal hosts, sshutle intercepts local packets, sends their payload over SSH and establishes connections from the remote public host to the internal hosts. This prevents some of the performance issues associated with TCP end-to-end connections through SSH inner tunnels and also enables other capabilities such as accessing more transparently hosts and services behind the firewall.

Requirements:

- Python 2.3 or higher on the remote server side.
- sshuttle requires root privileges in your local machine (desktop, laptop, etc), root privileges in the remote machine are not required.

- sshuttle must be installed in the local client side, sshuttle is not required on the remote host.
- Create an sshutle configuration file for instance in $HOME/.ssh/sshutle-ncg.conf with the following content:
    - **remote networks plus prefixes** one per line (e.g. 192.168.1.0/24)
    - **--auto-nets** optional to add networks know by the remote host side.
    - **--dns** optionally enables the capture of dns requests and forwards them to the remote side, use carefully as it may disrupt your local DNS queries. Also if you have multiple sshuttle instances you can only have one of them capturing and forwarding the DNS queries.
    - **--auto-hosts** to enable discovery of hosts in the remote side
    - **--seed-hosts** followed by a line containing remote hostnames separated by commas that you want to use
    - **--remote** followed by a line with the hostname of the public remote host to act as intermediate host

Template for the sshutle configuration file:

```
ip-network-A/ip-prefix-A
ip-network-B/ip-prefix-B
ip-network-X/ip-prefix-X
--auto-nets
--auto-hosts
--seed-hosts
internal-remote-host-01,internal-remote-host-02, ...
--remote
username@public-remote-host
```

Example that needs to be adjusted to your actual needs:

```
192.168.1.0/24
192.168.2.0/24
--auto-hosts
--seed-hosts
internal.ncg.ingrid.pt, otherinternal.ncg.ingrid.pt, anotherinternal.ncg.ingrid.pt
--remote
username@public.ncg.ingrid.pt
```

Invoking sshutle from the command line:

```
sshuttle @$HOME/.ssh/sshutle-ncg.conf
```

Then you can ssh into the hosts directly:

```
ssh internal.ncg.ingrid.pt
```

If the remote internal host has other services running such as a web server they will be also accessible:

```
curl http://otherinternal.ncg.ingrid.pt
```

Revision #69
Created 11 January 2021 12:16:32 by Jorge Gomes
Updated 12 January 2021 14:11:20 by Jorge Gomes