CompaSSH configuration

Compassh has a really small codebase made of three files:

- **compassh**: the command used by you to start and stop Compassh pseudo-VPNs
- **compassh_proxy**: the helper used by OpenSSH
- **compassh.utils**: the library used by compassh_proxy and compassh

To install use the provided script **install.sh**. The main command **compassh** will be installed in **/usr/bin**, while **compassh_proxy** and **compassh.utils** will go in **/usr/libexec/compassh/**.

The whole configuration of **Compassh** is held in *~/.compassh.conf*. This file contains two sections. The first, **%VPN**, defines the pseudo-VPNs, while the second, **%patterns**, describe the routing policies. Let's see the first:

```
our %VPN = (
   strumentiresistenti => {
      proxy => 'root @ proxy.strumentiresistenti.org.',
      local_port => "1080"
   },
)
```

Here we have the definition of a VPN. It's called *strumentiresistenti* and is routed through *proxy.strumentiresistenti.org* using the *root* account (of course you can use any SSH account). The **local_port** is the local port used for the SOCKS proxy. Each VPN must have a separate local port. So for example we can expand this configuration with another VPN:

```
our %VPN = (
   strumentiresistenti => {
      proxy => 'root @ proxy.strumentiresistenti.org.',
      local_port => "1080"
   },
   office => {
      proxy => 'jondoe @ proxy.bigcompany.biz.',
      local_port => "1081"
   },
);
```

Here we have a second VPN, called *office, using proxy.bigcompany.biz* as gateway and *jondoe* as login account. As we said before, the local port must be different, so we choose 1081 here.

Next we setup the **%pattern** section where we define the regexp patterns to be matched against host names to let Compassh choose where the connection should be routed.

```
our %patterns = (
    'strumentiresistenti.org$' => 'strumentiresistenti',
    'bigcompany.biz$' => 'office',
);
```

It should be quite self explanatory: any host matching regexp pattern *strumentiresistenti.org$* should be routed through the first VPN called *strumentiresistenti* while any host matching *bigcompany.biz$* should be routed through the second, *office. Any other destination will not be routed and will instead start a direct connection, as usually.*

Now Compassh is configured, but how do we tell OpenSSH that we would use it? The answer lays in the *~/.ssh/config* file.

Once CompaSSH has been installed and configured it must be connected to OpenSSH. The configuration is very simple and involves just adding one single line in the *~/.ssh/config* file:

```
Host *
    ProxyCommand /usr/libexec/compassh/compassh_proxy %h %p
```

If your config doesn't have a *Host ** section, just define one and add this line. I suggest adding some more directives. Mine for example is:

```
Host *
    ServerAliveCountMax 20
    ServerAliveInterval 15
    TCPKeepAlive yes
    ProxyCommand /usr/libexec/compassh/compassh_proxy %h %p
```

Now OpenSSH will call compassh_proxy for any connection you request. compassh_proxy will setup a permanent VPN, create the SOCKS proxy at specified **local_port** and setup any port forwarding you desire.

Right, what about port forwarding? We just set up the SOCKS proxy. What if I need to forward remote port 1234 on my host? Well, this kind of setup must me configured in a file called *~/.ssh/config.VPN_name*. So, for example, we must create two files called *~/.ssh/config.strumentiresistenti* and *~/.ssh/config.office* and define any port forwarding as shown here:

```
#
# ~/.ssh/config.strumentiresistenti
#
Host *
        LocalForward 10001 localhost:10001
        LocalForward 1234 192.168.1.20:1234
        LocalForward 4001 192.168.1.22:4001
```

When the VPN is fired up, the local ports 10001, 1234 and 4001 are forwarded.

Now we are ready to do some test and see if everything is working properly. First of all, please add *~/bin/Compassh* to your $PATH environment. Don't forget to source your *~/.bashrc* or to logout and login again. You should be able to call **compassh** without adding any path.

So let's see how it works.

```
$ compassh
   VPN name               SSH connection                       Port  PID
----------------------------------------------------------------------------
   strumentiresistenti    root @ proxy.strumentiresistenti.org  1081  -
   office                 root @ proxy.bigcompany.biz           1082  -
```

compassh is listing our two VPNs, with the SSH connection profile (user and host), the SOCKS port and the PID of the process holding the VPN. Since no VPN has been enabled, the PID is null for both. So, let's start a VPN:

```
$ compassh start strumentiresistenti
$ compassh
   VPN name                SSH connection                      Port  PID
----------------------------------------------------------------------------
 + strumentiresistenti    root @ proxy.strumentiresistenti.org  1080  14932
   office                  root @ proxy.bigcompany.biz           1081  -
```

The *strumentiresistenti* VPN now has a plus sign and a PID associated. It's running. If you use **netstat** you'll see an OpenSSH listening on port 1080: that's the SOCKS proxy waiting for incoming connections. You can for example configure your browser to use localhost:1080 as SOCKS proxy and be able to connect to a private IP reachable by *proxy.strumentiresistenti.org*, let's say 192.168.1.20, even if you are on a totally separate network.

You can of course connect to localhost on ports 10001, 1234 and 4001 to access remote services forwarded by OpenSSH. And finally you can SSH on *whatever.strumentiresistenti.org* and get routed through *proxy.strumentiresistenti.org*:

```
$ ssh -l root mail.strumentiresistenti.org
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:45:ec:66 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.20/24 brd 217.70.191.255 scope global eth0
#
```

VERY IMPORTANT: To be able to use names instead of IP addresses, the names must be resolved by one of those:

- Your */etc/hosts* file
- Your DNS name server
- The remote proxy */etc/hosts* file
- The remote proxy DNA name server

Usually the best position is the remote proxy DNS name server, since *something.internal.domain* makes most sense inside a network than outside it. But if the remote proxy can't resolve the name, just add it to your */etc/hosts* file or the remote */etc/hosts* file, if you have the required permissions.

Since VPNs and patterns are separate concepts, you can define more than one pattern pointing to the same VPN. So if you have a corporate VPN concentrator tha connects to customers networks, you can just define one single pseudo-VPN to the concentrator, as we did in the *office* VPN, and then add as many pattern as you need, like in:

```
our %patterns = (
    'bigcompany.biz$' => 'office',
    'customer1.com$' => 'office',
    'customer2.com$' => 'office',
    'anothercustomer.net$' => 'office',
)
```

compassh_proxy will route any connection to all those customers through the **office** VPN.

Now use the **ps** command to see how **Compassh** is managing all of this:

```
$ ps aux | grep ssh
tx0    8838  0.0  0.0   6148  2340 pts/4    S    10:51   0:00 /usr/bin/ssh
   -F /home/tx0/.ssh/config.strumentiresistenti -N -D 1080
   root @ proxy.strumentiresistenti.org
```

compassh has started a backgrounded ssh session to *root @ proxy.strumentiresistenti.org*, using the configuration file *~/.ssh/config.strumentiresistenti* and creating with -D a SOCKS proxy on port *1080*. That's how the VPN is kept up and running

As you have seen, **Compassh** is very useful with **ssh** itself and all the software that support SOCKS protocol, like web browsers. But what if the software you're using doesn't support SOCKS? Well, don't dispair: UNIX comes in rescue as usual.

There's a wonderful feature of UNIX linker called **LD_PRELOAD**. Using library preloading, the linker can replace some standard function with substitutes coming from additional libraries. Exploiting this feature, many software allows you to *socksify* software otherwise unable to make a SOCKS connection. One great example is **tsocks**, available at http://tsocks.sourceforge.net. Download it and edit its file */etc/tsocks.conf* by adding the following lines:

```
server = 127.0.0.1
server_type = 5
server_port = 1080
```

where **server_port** is the SOCKS port of the VPN you want to route your application through. Now you can socksify any software, just by adding **tsocks** in front of the command line. Let's say you need to log in on a remote router which does not support SSH, so you need to use

telnet:

```
tsocks telnet 192.168.1.254
```

That's it.