SSH Tips & Tricks

David Schoen

2013-03-01 Fri

Contents

1 Talk - SSH tips & tricks

1.1 What is SSH?

1.1.1 History (as I see it)

- 1983 rsh for BSD as part of rlogin
- 1985 I was born
- Other stuff happens
- 1995 - I get my first computer and SSH is born (totally related)

1.2 Keys

Keys are useful for doing away with passwords.

Generate a key using ssh-keygen.

Use ssh-copy-id to install it on remote nodes (you will need some backup authentication type at this point, e.g. an old key or an available password).

Basic security - use a passphrase to encrypt your key.

1.2.1 agents

Typing a passphrase is tedious, use an agent to load it in to memory in a **relatively** secure manner.

ssh-agent is the old classic. These days Gnome etc tend to have something "built in" that may or may not work for you. PuTTy also provides Pageant for those other OSes.

1.3 mssh/cssh

mssh and cssh are both useful for managing multiple remote machines over SSH at once.

cssh provides a thin wrapper around a normal xterm and mssh provides an integrated (and better tiling) solution.

I demo'd editing authorized_keys on all hosts at once using mssh.

1.4 bash tab completion

You can add ssh tab completion to other commands, e.g. mssh by aliasing it with:

```
shopt -u hostcomplete && complete -F _ssh mssh
```

Place in .bashrc to make it persistent.

1.5 Jumping between hosts

There's a few methods for jumping between hosts, but most people seem to use agent forwarding and it's fairly dangerous, so please consider the alternatives.

1.5.1 Agent forwarding

Once you have an agent set up, it can be forwarded to an intermediate host with "ssh -A", but if it's not at least as trustworthy as the box you're forwarding from, you are compromising your security.

The ssh agent socket can be seen either as the current user or as root with:

```
ps auxe | grep SSH_
```

anyone with access can then use that to hop on to other hosts that your agent can authenticate against.

1.5.2 Manually bouncing with nc

A safer method is to bounce using nc:

```
ssh -o 'ProxyCommand ssh <user>@<bouncehost> nc %h %p' <user>
```

I've tried to explain this previously here: http://lyte.id.au/2012/03/19/ssh-agent-forwarding-is-a-bug/ (if something is unclear there, feel free to give me an email or leave a comment).

1.5.3 random arbitrary bouncing with nc

Bounce somewhere arbitrary via a known bounce host:

```
# Place in ~/.ssh/config to connect anything via "ssh0" by just adding "-via-ssh0"
# on the end of the hostname:
Host *-via-ssh0
ProxyCommand ssh root@ssh0 nc $(echo %h | sed 's/-via-ssh0$//') %p
```

Now stack them!

Use this in ~/.ssh/config to be able to just bounce via anything:

```
Host *-via-*
ProxyCommand ssh $(echo %h | sed -r 's/^.*-via-//') nc $(echo %h | sed 's/-via-.*$//'
```

Note: I haven't figured out how to make this work with arbitrary usernames yet, but if you've at least got the username configured for each host in .ssh/config then you can just start bouncing in any direction with this.

1.6 Forwarding "stuff"

ssh can:

- forward (getting a port from the remote back to local "-L ...")
- reverse forward (getting a port from local over to the remote "-R ...")
- dynamic forward (socks proxy - tsocks can help)

The demo I gave where Netflix thought I was in America was by:

- installing tsocks
- configuring tsocks to use 127.0.0.1:1080 (in /etc/tsocks.conf)
- connecting to a US host with "ssh -D 1080 <ushost>"
- then running firefox in a fresh profile under tsocks "tsocks firefox -P -no-remote"

1.7 editing over ssh

Vim - navigate to:

```
sftp://<user>@<host>/<path>
```

Emacs - navigate to:

```
//ssh:<user>@<host>:<path>
```

1.8 mosh

"Remote terminal application that allows roaming, supports intermittent connectivity, and provides intelligent local echo and line editing of user keystrokes."

http://mosh.mit.edu/

The fine print:

- drops out if the server changes IP
- requires a UDP port to establish a secondary encrypted stream
- bleeding edge
- can leave dangly hard to reestablish procs open

1.9 Security when exposing sshd to the internet

If you have a sshd daemon connected to the internet here's a few basic security strategies.

1.9.1 brute force prevention

Try denyhosts, fail2ban or anything that drops repeated failures.

1.9.2 password auth limiting

If you have a box exposed to the internet I would strongly recommend disabling password authentication (note: set up your keys first!) with this in /etc/ssh/sshd_config:

```
PasswordAuthentication no
```

If your sshd is new enough you can disable it globally (above) and then put in a match section to selectively allow password auth, e.g. just allow local addresses:

```
Match Address 192.168.0.0/16,172.16.0.0/12,10.0.0.0/8,127.0.0.1/32
PasswordAuthentication yes
```

1.9.3 AllowUsers

Try to reduce the available users down to those that actually need it with AllowUsers or "PermitRootLogin no".

2 Stuff I ran out of time to talk about

2.1 X11

Generally if you're doing X forwarding, you're opening your desktop up to attack, please watch: http://mirror.internode.on.net/pub/linux.conf.au/2013/ogv/Aint_No_Party_Like (preferably all the way through, but ssh X forwarding starts from about 16:54).

2.2 sshfs

Good if you want to navigate a remote machine as if it was local.

If you use this with svn try "sshfs -o workaround=rename user@host:path mount" to solve "permission" issues.

3 Pre talk config

Just for completeness I've included the commands I use to prepare the VMs (works on Ubuntu 12.04).

```
hosts=(ssh{0..9})

# clean up
time for h in "${hosts[@]}"; do
lxc-halt -n "$h"
lxc-destroy -n "$h" -f
rm -rf -- /var/lib/lxc/"$h"
btrfs subvolume delete /var/lib/lxc/$h/rootfs
lxc-destroy -n "$h" -f
done

# halt
time for h in "${hosts[@]}"; do
lxc-stop -n "$h"
done

# creation
time for h in "${hosts[@]}"; do
lxc-create -n "$h" -t ubuntu
done

# boot
time for h in "${hosts[@]}"; do
lxc-start -n "$h" -d
done
```

list vm ips in to /etc/hosts:

```
for i in ssh{0..9}; do echo $(dig $i @10.0.3.1 +short) $i; done >> /etc/hosts
```