Linux Fu: Stupid SSH Tricks

Originalartikel

Backup

<html> If you connect to remote computers over the Internet, it is a pretty good chance you use some form of SSH or secure shell. On Linux or Unix you’ll use the

```
ssh
```

command. Same goes for Linux-like environments on Windows like Cygwin or WSL. For native Windows, you might be using Putty. In its simplest form,

```
ssh
```

is just a terminal program that talks to a server using an encrypted connection. We think it is very hard to eavesdrop on anyone communicating with a remote computer via

```
ssh
```

. There are several tricks for using

```
ssh
```

— some are pretty straightforward and some are things you might not think of as being in the domain of a terminal program. You probably know that

```
ssh
```

can copy files securely, and there are easy and hard ways to set up logging in with no password. However, you can also mount a remote filesystem via

```
ssh
```

(actually, there are several ways to do that). You can use

```
ssh
```

to securely browse the web in your favorite browser, or even use it to tunnel specific traffic by port or even use it as a makeshift VPN. In fact, there’s so much ground to cover that this won’t be the last Linux Fu to talk about

```
ssh
```

. But enough setup, let’s get to the tricks. <h2>A Few Good Options</h2> We’ll assume you know the basics:

```
scp
```

and

```
sftp
```

for file copies and

```
ssh-copy-id
```

for setting up password-less login. (If not, you have ten minutes to do a quick web search.) But one of the things

```
ssh
```

is great at is manipulating the network. Keep in mind, though, that the server has to have certain options set to do some of the most interesting things, so if you don’t control the

```
ssh
```

server, some of these tricks might not work for you. There are a lot of options to remember on the

```
ssh
```

command line, but luckily you don’t have to. You don’t even have to remember your hostname or user name. In the

```
~/.ssh/config file
```

you can create an alias. For example, suppose you want to connect to your home server: <pre class=„brush: plain; title: ; notranslate“ title=„“> Host homeserver

```
HostName mih0me.dyndns.org
Port 1234
User TheAl
IdentityFile ~/.ssh/home_id
ForwardX11 yes
Compression yes
TCPKeepAlive yes
```

</pre> You can have as many aliases as you like. Just keep repeating the Host line and then follow it with options. You can also add more than one alias to a single Host statement. The subsequent options will then apply to any of the aliases. Now to connect just issues

```
ssh homeserver
```

and you are in with all the right options. Of course, if you are using Putty, your options will mainly be in the host profile and on the

```
ssh
```

panel of the options screen. You might not get as many options, but there are some you can try. <h2>Be Persistent</h2> One really nice set of options to include set the master control file. For example: <pre class=„brush: plain; title: ; notranslate“ title=„“> ControlMaster auto ControlPath ~/.ssh/master-%r@%h:%p </pre> This lets multiple sessions to the same host share a single TCP socket. Because it takes a some time to set up a secure socket, this can be speed things up if you keep several sessions going between two hosts. You can set it for all hosts by using the

```
Host *
```

line in the config file. You can use that for any global options you might have. On the other hand, if you shove a lot of data over multiple connections, turning on

```
ControlMaster
```

might not be a great idea. You can add

```
-S none
```

to override the global setting. One other thing to note is that your first

```
ssh
```

session might appear to hang if you try to exit it before all the other connections are closed. Some people deliberately run a hidden

```
ssh
```

session on login to a host they often connect to which avoids that problem. However, a better way is to set

```
ControlPersist
```

to yes. That will cause the original session to go to the background indefinitely. If you want a little grace period you can set

```
ControlPersist
```

to a number like 180. That would cause the background session to end if there are no connections for three minutes. Another downside to this approach is that you tend to get orphaned master files. In

```
rc.local
```

I have the following line: <pre class=„brush: plain; title: ; notranslate“ title=„“> /bin/rm /home/*/.ssh/master-* || true >/dev/null </pre>

With Putty, you can click the “Share SSH connections if possible” button in the SSH options panel. <h2>Configuration</h2> There are quite a few <a

href=„http://man7.org/linux/man-pages/man5/ssh_config.5.html“ target=„_blank“>configuration options you can use in the config file. For example,

```
BatchMode
```

lets you tell

```
ssh
```

the connection is made for unattended use so don’t bother prompting the user for passwords or anything else. That’s not to say it will just let you in, of course. It just means it will fail if you don’t have things set up to login without a password. There are some other interesting items. For example, you can run a local or remote command on connection. You can also send an environment variable to the remote host or even just set one. For example, suppose you want to always keep your LS_COLORS the same on your workstation and the server, but you frequently change them and don’t want to use the same profiles. You could add this to the host’s config file entry: <pre class=„brush: plain; title: ; notranslate“ title=„“> SendEnv LS_COLORS </pre> The server has to be configured to accept this, of course. Putty can handle setting an environment variable from the Data tab in its setup. On the network side, you can specify

```
TCPKeepAlive
```

to yes if you want the server and client to test their connection during idle periods. This is a two-edged sword. If the connection is idle, you won’t get disconnected. But if the network drops at the right moment for a brief period you might get disconnected where you wouldn’t have if you had stuck with the default. There’s even a way to open a layer 3 or layer 2 tunnel between the machines — a topic for a future Linux Fu. <h2>By Your Command</h2> Don’t forget that

```
ssh
```

can execute a command and send its output to you. As a practical example, I occasionally reflash my 3D printer firmware. The printer is connected to a Raspberry Pi, but I do the firmware build on my main machine. For a long time, I copied my file to the Pi (using

```
scp
```

) and then logged into the Pi to run a script I wrote called

```
flash
```

. The script disables the Reptier server software, flashes the Atmel chip on the printer control board, and then turns the server back on. Here’s the script that runs on my main computer. Note the

```
ssh
```

commands. One turns off the server. One

```
scp
```

command copies the new firmware over. Then another

```
ssh
```

does the flash and renables the server. There are many other ways to do this, of course. But don’t forget that

```
ssh
```

can run a remote command and then return. <pre class=„brush: bash; title: ; notranslate“ title=„“> #!/bin/bash if [-z „$1“] then

```
echo Usage: flash hexfile [remote_name]
echo If omitted, remote_name will have date attached
exit 1
```

fi ONAME=„$2“ EXTRA=$(date „+%Y%m%d.%H%M“) if [-z „$2“] then

```
ONAME="$1-$EXTRA.hex"
```

fi IP=192.168.1.110

```
echo Stop Server
ssh -l pi $IP "sudo systemctl stop RepetierServer"
echo Copy...
scp "$1" "pi@$IP:a8fw/$ONAME"
echo Flashing...
```

ssh pi@$IP „cd a8fw; ./flash $ONAME“ echo Restart ssh „pi@$IP“ „sudo systemctl start RepetierServer“ echo Done exit 0 </pre> If you want a more hacker-friendly example, consider using the same idea to run Wireshark locally and analyze a remote packet capture: <pre class=„brush: plain; title: ; notranslate“ title=„“> ssh root@someserver 'tcpdump -c 1000 -nn -w - not port 22' | wireshark -k -i - </pre> You can do the same trick with

```
tshark,
```

if you prefer. <h2>Speed Test</h2> What to know how fast your

```
ssh
```

connection is? Make sure you have

```
pv
```

installed and try this: <pre class=„brush: bash; title: ; notranslate“ title=„“> yes | pv | ssh remote_host „cat >/dev/null“ </pre> <img data-attachment-id=„384165“

data-permalink=„https://hackaday.com/2019/12/17/linux-fu-stupid-ssh-tricks/speed-2/“ data-orig-file=„https://hackaday.com/wp-content/uploads/2019/10/speed.png“ data-orig-size=„800,84“ data-comments-opened=„1“ data-image-meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"0"}“ data-image-title=„speed“ data-image-description=„“ data-medium-file=„https://hackaday.com/wp-content/uploads/2019/10/speed.png?w=400“ data-large-file=„https://hackaday.com/wp-content/uploads/2019/10/speed.png?w=800“ class=„aligncenter wp-image-384165 size-large“ src=„https://hackaday.com/wp-content/uploads/2019/10/speed.png?w=800“ alt=„“ width=„800“ height=„84“ srcset=„https://hackaday.com/wp-content/uploads/2019/10/speed.png 800w, https://hackaday.com/wp-content/uploads/2019/10/speed.png?resize=250,26 250w, https://hackaday.com/wp-content/uploads/2019/10/speed.png?resize=400,42 400w“ sizes=„(max-width: 800px) 100vw, 800px“/> Pretty cool! If you have good speeds — or even if you don’t — you can try mounting a remote file system using

```
sshfs
```

. This is a FUSE filesystem — that is, a filesystem that lives as a regular user program, not part of the kernel. With nothing on the remote side but the

```
ssh
```

server and standard tools, you can make any host you can connect to look like a local file system. <h2>But Wait…</h2> There’s a lot more you can do with

```
ssh
```

, and I’ll cover more shortly. But for now, hopefully you found at least one

```
ssh
```

trick you can use that was, if not new, at least a reminder for you. </html>