OpenSSH is a **FREE** version of the SSH connectivity tools that technical users of the Internet rely on. Users of telnet, rlogin, and ftp may not realize that their password is transmitted across the Internet unencrypted, but it is. OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other attacks. The encryption that OpenSSH provides has been strong enough to earn the trust ofTrend Micro and other providers of cloud computing.Additionally, OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions.

SSH is an awesome powerful tool, there are unlimited possibility when it comes to SSH, heres the top Voted SSH commands

1) Copy ssh keys to user@host to enable password-less ssh logins.

ssh-copy-id user@host

To generate the keys use the command ssh-keygen

2) Start a tunnel from some machine's port 80 to your local post 2001

ssh -N -L2001:localhost:80 somemachine

Now you can acces the website by going to http://localhost:2001/

3) Output your microphone to a remote computer's speaker

dd if=/dev/dsp | ssh -c arcfour -C username@host dd of=/dev/dsp

This will output the sound from your microphone port to the ssh target computer's speaker port. The sound quality is very bad, so you will hear a lot of hissing.

4) Compare a remote file with a local file

ssh user@host cat /path/to/remotefile | diff /path/to/localfile –

Useful for checking if there are differences between local and remote files.

5) Mount folder/filesystem through SSH

sshfs name@server:/path/to/folder /path/to/mount/point

Install SSHFS from http://fuse.sourceforge.net/sshfs.html
Will allow you to mount a folder security over a network.

6) SSH connection through host in the middle

ssh -t reachable_host ssh unreachable_host

Unreachable_host is unavailable from local network, but it's available from reachable_host's network. This command creates a connection to unreachable_host through "hidden" connection to reachable_host.

7) Copy from host1 to host2, through your host

ssh root@host1 "cd /somedir/tocopy/ && tar -cf – ." | ssh root@host2 "cd /samedir/tocopyto/ && tar -xf -"
Good if only you have access to host1 and host2, but they have no access to your host (so ncat won't work) and they have no direct access to each other.

8) Run any GUI program remotely

ssh -fX <user>@<host> <program>
The SSH server configuration requires:

X11Forwarding yes # this is default in Debian

And it’s convenient too:

Compression delayed

9) Create a persistent connection to a machine

ssh -MNf <user>@<host>

Create a persistent SSH connection to the host in the background. Combine this with settings in your ~/.ssh/config:
Host host
ControlPath ~/.ssh/master-%r@%h:%p
ControlMaster no
All the SSH connections to the machine will then go through the persisten SSH socket. This is very useful if you are using SSH to synchronize files (using rsync/sftp/cvs/svn) on a regular basis because it won’t create a new socket each time to open an ssh connection.

10) Attach screen over ssh

ssh -t remote_host screen -r

Directly attach a remote screen session (saves a useless parent bash process)

11) Port Knocking!

knock <host> 3000 4000 5000 && ssh -p <port> user@host && knock <host> 5000 4000 3000

Knock on ports to open a port to a service (ssh for example) and knock again to close the port. You have to install knockd.
See example config file below.

```
[options]
logfile = /var/log/knockd.log
[openSSH]
sequence = 3000,4000,5000
seq_timeout = 5
command = /sbin/iptables -A INPUT -i eth0 -s %IP% -p tcp –dport 22 -j ACCEPT
tcpflags = syn
[closeSSH]
sequence = 5000,4000,3000
seq_timeout = 5
command = /sbin/iptables -D INPUT -i eth0 -s %IP% -p tcp –dport 22 -j ACCEPT
tcpflags = syn
```

12) Remove a line in a text file. Useful to fix

ssh-keygen -R <the_offending_host>

In this case it’s better do to use the dedicated tool

13) Run complex remote shell cmds over ssh, without escaping quotes

ssh host -l user $(<cmd.txt)

Much simpler method. More portable version: ssh host -l user “`cat cmd.txt`”

14) Copy a MySQL Database to a new Server via SSH with one command

mysqldump –add-drop-table –extended-insert –force –log-error=error.log -uUSER -pPASS OLD_DB_NAME | ssh -C user@newhost “mysql -uUSER -pPASS NEW_DB_NAME”

Dumps a MySQL database over a compressed SSH tunnel and uses it as input to mysql – i think that is the fastest and best way to migrate a DB to a new server!

15) Remove a line in a text file. Useful to fix “ssh host key change” warnings

sed -i 8d ~/.ssh/known_hosts

16) Copy your ssh public key to a server from a machine that doesn't have ssh-copy-id

cat ~/.ssh/id_rsa.pub | ssh user@machine "mkdir ~/.ssh; cat >> ~/.ssh/authorized_keys"

If you use Mac OS X or some other *nix variant that doesn't come with ssh-copy-id, this one-liner will allow you to add your public key to a remote machine so you can subsequently ssh to that machine without a password.

17) Live ssh network throughput test

yes | pv | ssh $host "cat > /dev/null"

connects to host via ssh and displays the live transfer speed, directing all transferred data to /dev/null
needs pv installed
Debian: 'apt-get install pv'
Fedora: 'yum install pv' (may need the 'extras' repository enabled)

18) How to establish a remote Gnu screen session that you can re-connect to

ssh -t user@some.domain.com /usr/bin/screen -xRR

Long before tabbed terminals existed, people have been using Gnu screen to open many shells in a single text terminal. Combined with ssh, it gives you the ability to have many open shells with a single remote connection using the above options. If you detach with "Ctrl-a d" or if the ssh session is accidentally terminated, all processes running in your remote shells remain undisturbed, ready for you to reconnect. Other useful screen commands are "Ctrl-a c" (open new shell) and "Ctrl-a a" (alternate between shells). Read this quick reference for more screen commands: http://aperiodic.net/screen/quick_reference

19) Resume scp of a big file

rsync –partial –progress –rsh=ssh $file_source $user@$host:$destination_file

It can resume a failed secure copy (usefull when you transfer big files like db dumps through vpn) using rsync.
It requires rsync installed in both hosts.
rsync –partial –progress –rsh=ssh $file_source $user@$host:$destination_file local -> remote
or
rsync –partial –progress –rsh=ssh $user@$host:$remote_file $destination_file remote -> local

20) Analyze traffic remotely over ssh w/ wireshark

ssh root@server.com 'tshark -f "port !22" -w -' | wireshark -k -i –

This captures traffic on a remote machine with tshark, sends the raw pcap data over the ssh link, and displays it in wireshark. Hitting ctrl+C will stop the capture and unfortunately close your wireshark window. This can be worked-around by passing -c # to tshark to only capture a certain # of packets, or redirecting the data through a named pipe rather than piping directly from ssh to wireshark. I recommend filtering as much as you can in the tshark command to conserve bandwidth. tshark can be replaced with tcpdump thusly:
ssh root@example.com tcpdump -w – 'port !22' | wireshark -k -i –

21) Have an ssh session open forever

autossh -M50000 -t server.example.com 'screen -raAd mysession'

Open a ssh session opened forever, great on laptops losing Internet connectivity when switching WIFI spots.

22) Harder, Faster, Stronger SSH clients

ssh -4 -C -c blowfish-cbc

We force IPv4, compress the stream, specify the cypher stream to be Blowfish. I suppose you could use aes256-ctr as well for cypher spec. I'm of course leaving out things like master control sessions and such as that may not be available on your shell although that would speed things up as well.

23) Throttle bandwidth with cstream

tar -cj /backup | cstream -t 777k | ssh host 'tar -xj -C /backup'

this bzips a folder and transfers it over the network to “host” at 777k bit/s.
cstream can do a lot more, have a look http://www.cons.org/cracauer/cstream.html#usage for example:
echo w00t, i’m 733+ | cstream -b1 -t2

24) Transfer SSH public key to another machine in one step

ssh-keygen; ssh-copy-id user@host; ssh user@host

This command sequence allows simple setup of (gasp!) password-less SSH logins. Be careful, as if you already have an SSH keypair in your ~/.ssh directory on the local machine, there is a possibility ssh-keygen may overwrite them. ssh-copy-id copies the public key to the remote host and appends it to the remote account’s ~/.ssh/authorized_keys file. When trying ssh, if you used no passphrase for your key, the remote shell appears soon after invoking ssh user@host.

25) Copy stdin to your X11 buffer

ssh user@host cat /path/to/some/file | xclip

Have you ever had to scp a file to your work machine in order to copy its contents to a mail? xclip can help you with that. It copies its stdin to the X11 buffer, so all you have to do is middle-click to paste the content of that looong file :)