Using SSH at SCI

Category: SCI Documentation

Table of Contents

*TL;DR: *check out Putting It All Together and copy the* `~/.ssh/config` *file there, and use the UIT VPN when off campus.*

The Internet is a wild and dangerous place, as we all know, which means that we need use safe and secure methods for working with computing resources at SCI, particularly when connecting to and using remote hosts. The canonical way for working with remote hosts is SSH – the Secure Shell. SSH encrypts traffic between hosts and uses secure authentication protocols to enable safe communication between hosts, and while it is mostly used for command-line work it can also be used to tunnel X11 and other types of traffic as well. SSH is the recommended way to access remote computers (servers,

desktops, etc) both within SCI and when working with SCI machines from outside of the SCI network. SSH is installed on all SCI UNIX hosts and comes installed by default on MacOS, so you can just run `ssh` from the command line. If you are using a Windows machine, Putty and the Windows Terminal both support SSH, as do many other available applications (e.g. MobaXTerm, SecureCRT, etc).

How SCI is set up to use SSH

All Unix hosts in SCI are set up to use SSH by default, so you can easily connect to them by simply ssh'ing into them using your preferred client. SSH is unrestricted within the UofU network, which also includes any machines connected via the campus VPN. When coming from outside of the UofU network, however, access is limited to shell.sci.utah.edu only, as restricting what we expose to the outside world makes it much harder for attackers to find ways of exploiting our systems and enables us to more quickly identify, respond to, and remediate threats. This means that when you are coming from off campus, you will need to go through `shell.sci.utah.edu` using port 5522, either by connecting to it directly or using it as a jump host (see Using SSH from Off Campus below for how to make this easy/transparent). Note: the `shell.sci.utah.edu` machines are on high-speed networks to enable fast access, including large data transfers.

Tips and Tricks for using SSH

Creating Keys

When ssh'ing into a machine, you can authenticate yourself using your username and password, but most people set up an SSH key, which identifies you as being you and makes SSH an easier to use. There are tons of FAQs out there on what keys

are and how to use them (e.g. ssh.com's page on ssh-keygen). For a quick start, however, you can use this command on a linux box to create a new key for use at SCI:

```
ssh-keygen -t ed25519 -f ~/.ssh/id_sci -C "SCI Institute"
```

This will prompt you to create a passphrase to protect your SSH identity, and like all other passwords and passphrase, the longer and more diverse the better.

Distributing Keys

Note that when you create a new SSH key you actually get two files— in the example above they would be `id_sci` and `id_sci.pub`. The first one, `id_sci`, is the private key that you use to prove that you are you, and the second is the public key, which can be used to make is so you can access hosts using your key instead of your regular SCI password. All you need to do is to put the contents of the `.pub` file into your authorized_keys file (`~/.ssh/authorized_keys`) on the host you are trying to access, and since in SCI your home directory is shared across all Linux machines, you only need to do this once to use your key across all of SCI. And to make things even easier, you can use the `ssh-copy-id` command, which does all the work for you.

Using ssh-agent to Make Authentication Easier

To avoid having to type your key's passphrase every time you try to connect to another machine, you can use ssh-agent to cache your identity for you, which will make authentication transparent to you. A good practice is to do this on the box you use most frequently, such as your desktop or laptop, and connect to all other machines from there, so that you don't have authenticated processes running on a bunch of machines scattered everywhere.

Linux

The simplest way to use ssh-agent on Linux is to add the following line to your startup file (e.g. `.zshrc`, `.bashrc`, etc):

```
eval `ssh-agent`
```

Then, for each identity you want ssh-agent to cache for you, just run `ssh-add <identity file>` and you're off to the races. For more information, see How To Use Ssh-agent Safely.

MacOS

The version of SSH on MacOS is integrated with Mac's Keychain, so using it is very simple and can be achieved in one of two ways: running `ssh-add –use-apple-keychain <identity_file>` or adding UseKeychain to `~/.ssh/config`.

Windows

How you set up SSH keys on Windows is dependent on what client you use, so please reference the documentation for your client (e.g. Putty).

Keep Alive

If you read the man page for `ssh_config`, you will find that there are a ton of configuration options available. Here are two options you might find useful when ssh-ing in from outside of the SCI network:

```
# The following two lines will help prevent connections
from dropping due to inactivity
ServerAliveInterval 300
ServerAliveCountMax 3
```

This will instruct SSH to send some packets every 5 minutes so that the connection is kept alive and therefore hopefully not automatically closed.

Using SSH from Off Campus

As a security policy, all off-campus SSH traffic is blocked, except through `shell.sci.utah.edu` on a non-standard port (5522). So, if you want to connect to SCI computers from off-campus, you can either use the campus VPN or go through `shell.sci.utah.edu`.

UofU VPN

Using the campus VPN magically puts your machine on the SCI network, so from a networking standpoint you are virtually on the SCI network. As such, there isn't anything special you need to do– you can just SSH around just like if you were sitting in an office on campus.

Using shell.sci.utah.edu as a gateway

The one place that SSH is open to the world is via `shell.sci.utah.edu` on a non-standard port (5522). You can do that by specifying the port on the command line (e.g. `ssh -p 5522 shell.sci.utah.edu`) or by putting a line in your `.ssh/config` file to specifically use that port:

```
# This says to use port 5522 for shell.sci.utah.edu only
Host shell.sci.utah.edu
Port 5522
```

Using shell.sci.utah.edu as a jump host

You can also make accessing SCI hosts from off-campus simple and easy by setting up your SSH config to use shell.sci.utah.edu as a *jump host*. To do this, the first thing to do is to set up your SSH key and agent so that you can ssh into shell on port 5522 so you can do something like this without typing in your passphrase (see above about creating and managing keys):

```
[@mylaptop] ~:> ssh shell.sci.utah.edu uptime
```

```
15:47:41 up 27 days, 22:01,  8 users,  load average: 0.28,
0.17, 0.17
[@mylaptop] ~:>
```

Now you can use shell.sci.utah.edu as a jump host from the command line by using the arguments `-A -J shell.sci.utah.edu`:

```
[@mylaptop] ~:> ssh -A -J shell.sci.utah.edu -p 5522
foobar.sci.utah.edu hostname
foobar
[@mylaptop] ~:>
```

In this case, the command `hostname` was passed *through* `shell.sci.utah.edu` to the host `foobar.sci.utah.edu` where it was executed and returned that machine's hostname. If you don't want to remember how to do that, or if you don't want to have to type all that out when you are ssh'ing in from off campus, there are two ways you can automate this:

1. Set up an alias in your shell: adding something like this to your `.zshrc` or `.bashrc` will give you a new command to use for just such an occasion:

   ```
   alias sci-ssh='ssh -A -J ssh://shell.sci.utah.edu:5522
   $*'
   ```

2. You can set up your `ssh_config` with the following lines so that the jump host is automatically used with all SCI hosts. Note that this will *always* use the jump host with SCI machines, so this is best used on machines that you won't ever use with the campus VPN (i.e. not your personal laptop, but perhaps a machine at another facility you regularly use):

   ```
   # sci hosts need to go through
   Host *.sci.utah.edu !shell.sci.utah.edu
     ProxyJump shell.sci.utah.edu
     IdentityFile ~/.ssh/sci

   Host shell.sci.utah.edu
   ```

```
Port 5522
IdentityFile ~/.ssh/sci
```

Putting It All Together

Like many other systems out there, there are a slew of SSH clients and a seemingly infinite number of ways you can configure it. Here is how your humble author has his SSH configuration set up to work in SCI, using the information contained above, using a Mac laptop and a Linux desktop, plus a handy utility called `keychain` (which you can find installed on SCI systems by default). This assumes you have an SSH created and have copied it to `~/.ssh` on both the laptop and your SCI home directory.

Laptop

I assume that if I am using my (Mac) laptop I will either be on the campus network or using the VPN, so no need for a jump host configuration. I simply have this as my `~/.ssh/config` file:

```
# Disable forwarding
ForwardAgent no
ForwardX11 no

# Use sci.utah.edu as a domain by default
CanonicalizeHostname yes
CanonicalDomains sci.utah.edu
CanonicalizeMaxDots 3
CanonicalizeFallbackLocal yes

# Keep alive
ServerAliveInterval 300
ServerAliveCountMax 2

# Let's use agents
AddKeysToAgent yes
```

```
UseKeychain yes

Host *.sci.utah.edu
  IdentityFile ~/.ssh/id_sci
```

Using this I can just run `ssh` from the command line and get to any host in SCI when I am on campus or on the VPN. If, for some reason I cannot use the VPN (very rare occurrence), I do have this alias in my `~/.zshrc` file:

```
alias sci-ssh='ssh -A -J ssh://shell.sci.utah.edu:5522 $*'
```

Desktop

I do a lot of work on my desktop and often wind up ssh'ing from this computer to other hosts in the SCI domain. As such, I use the same `~/.ssh/config` file as on my laptop (with the "`UseKeychain`" line commented out, since that's only for Macs). I also run `ssh-agent` on my desktop, to make using my SCI SSH key (among others) simple and easy, and have this ZSH function defined in my ZSH environment to make it a simple one-command startup (with this defined, you just type in `kch` on the command line and it will set up the agent with all your keys for you):

```
#
# kch() - get ssh-agent running
#
# ARGUMENTS: <none>
#

function kch() {
  # kch just starts up keychain if necessary and sources
the ENVVAR file
  $( type -p keychain 2>&1 >/dev/null )  ||
     { printf "keychain executable not found\n" && return
1; }
  keychain --host ${HOST}
  if [ -f ~/.keychain/${HOST}-sh ]; then
     source ~/.keychain/${HOST}-sh;
  fi
```

```
  # parse .ssh/config for identity files
  idlist=$( ssh-add -l )
  for idfile in $( grep IdentityFile ~/.ssh/config | awk '{
print $2 }' ); do
     # ssh-add it if it isn't there already
     idfile=$( eval echo $idfile )   # get file expansion
     fp=$( ssh-keygen -l -f $idfile )
     if [[ $idlist != *"$fp"* ]]; then
        ssh-add $idfile
     fi
  done
}
```

Then I have this in my .zshrc file (for my machine called `mydesktop`):

```
# mydesktop is my main host
if [[ "$HOST" == "mydesktop" ]]; then
   # load up ssh-agent
   kch
fi
```

This way, whenever I log into my desktop box, it either connects to an already-running agent or starts up a new one, and makes sure all the keys that I use are loaded up. That way, when ssh’ing from my desktop to any other SCI machine I don’t have to type in my password/passphrase.