Using OpenSSH in Windows

Duncan Murdoch

October 30, 2006

1 Introduction

Hao Yu's tutorial concentrates on the ssh client from SSH Communications Security, for which UWO has a site license. I prefer to use the OpenSSH client, so I'll give versions of his instructions specific to it.

Why do I prefer OpenSSH?

- It is essentially identical on Windows, our Linux server, my MacOSX laptop, and a system using FreeBSD. I don't need to use different commands on each.
- It works well with other Cygwin tools.
- It is free of charge, whether on UWO computers or not.
- It is open source software.

The main disadvantage of OpenSSH is that it's an old-fashioned command line interface: none of the nice menus that the commercial ssh offers.

2 Getting and installing OpenSSH

You will need to be able to run the Cygwin `setup.exe` utility. This probably requires administrative access to your machine; most students won't have that.

Run `setup.exe`, choose an online mirror, and choose to install `openssh`. This will automatically install some other packages, such as `openssl`. Tell it to go ahead, and you're done!

3 Configuring the OpenSSH client

Within Cygwin or a Windows CMD line shell, type

```
ssh fisher.stats.uwo.ca
```

This will connect you to `fisher`, using the same login name as your current name in Windows. (You can login under a different name, e.g. by

```
ssh murdoch@fisher.stats.uwo.ca
```

if you happen to know the password.) Various other command line options are possible; I strongly recommend you **do not** use the option to include your password on the command line.

4 Generate a public/private keypair on the client

Use the command

```
ssh-keygen -b 1024 -t rsa
```

You will be prompted for where to save it (accept the default) and for a passphrase. I recommend that you *do* use a passphrase: otherwise someone breaking into this computer will be able to break into every other computer you use.

This will produce public key `id_rsa.pub` and private key `id_rsa` in your `~/.ssh` directory. You should rename the public key to something to indicate which machine you used to generate it (e.g. `office.pub`), send it to the server servers you want to use, then on the servers add it to your `~/.ssh/authorized_keys` file. Because our server also uses OpenSSH, there's no need to convert the format of the public key. This sequence of commands will do all of that for me; alter it in the obvious ways.

```
mv .ssh/id_rsa.pub .ssh/office.pub
scp .ssh/office.pub murdoch@fisher.stats.uwo.ca:.ssh
ssh murdoch@fisher.stats.uwo.ca
cat .ssh/office.pub >>.ssh/authorized_keys
```

5 Using ssh-agent to avoid typing your passphrase

It gets quite irritating to type the passphrase all the time, which is why many people don't use one. However, this is risky: if your computer is stolen, or your account is broken into, then someone could pretend to be you, and have all the access that you have.

The `ssh-agent` program is a program that runs continuously, and remembers your passphrase. When `ssh` needs a passphrase, it checks there first. You only need to type it once per session, instead of every time you use `ssh`.

I have the following command in my `~/.bash_profile` attached to the alias `sshstart`:

```
ssh-agent >~/ssh-agent-pid;source ~/ssh-agent-pid; ssh-add
```

This starts `ssh-agent`, loads information about where to find it, then loads my private key. I'll need to enter my passphrase at this point, but not afterwards.