Software Engineering
Supplement on CVS

1 Using CVS

We covered the basics of setting up a repository, adding a project to the repository, and making and using a working copy. Please refer to the lecture notes for details. If you have problems or questions not answered in the lecture notes, more information is available online at http://ximbiot.com/.

2 Using CVS remotely

2.1 Setting up the repository

The first step is the same as before: set up the CVS repository as described above on a Unix machine accessible via ssh.

2.2 Setting up secure access with ssh

On the machine containing the CVS repository, in your home directory, run the following command:

```
ssh-keygen -t dsa
```

When prompted for the name of the file to store the key, press return. When prompted for a passphrase, press return again. This generates a public/private key pair in a sub-directory called ".ssh". Now, copy the public key to a list of authorized keys:

```
cp .ssh/id_dsa.pub .ssh/authorized_keys
```

This allows you to log in from other machines without a password, as long as your ".ssh" directory on those other machines contains the same "id_dsa.pub" file.
To enable other users to access your repository remotely without giving them your password, do the following. Ask each user to run "ssh-keygen -t dsa" as above. Then have them send you their ".ssh/id_dsa.pub" file. Add this file to the end of your ".ssh/authorized_keys" file and the other users will have access. Unfortunately, this also means that other users can log in as you and access your files. To limit their access to ONLY using the CVS repository, add the following text *immediately before* their entry in the ".ssh/authorized_keys" file:

```
no-port-forwarding,no-X11-forwarding,no-agent-forwarding,command="/usr/bin/cvsserver"
```

2.3 Accessing CVS remotely

The steps for accessing CVS remotely are similar to the standard use of CVS except for two things:

1. You must have a ".ssh" subdirectory in your home directory containing a file called "id_dsa.pub" whose contents also reside in the ".ssh/authorized_keys" file on the remote machine (see above).

2. You must set the **CVS_RSH** environment variable to **ssh**. The way this is done depends on which Unix shell you are using. You are probably using csh, tcsh, ksh, or bash. To find out, type "ps" and look for the name of the shell process. Using csh or tcsh, environment variables are set as follows:

   ```
   setenv CVS_RSH ssh
   ```

   Using ksh or bash, set the environment variable like this:

   ```
   export CVS_RSH=ssh
   ```

3. Instead of just providing the path to the repostitory, you must also provide the machine where the repository sits. This is done by using the ":ext:" tag followed by your username and the remote machine. For example, if the repository is located in the directory "/home/barrett/repository" on the remote machine "access.cims.nyu.edu" under the username is "barrett", I would do the following to check out project "myproj":

   ```
   cvs -d :ext:barrett@access.cims.nyu.edu:/home/barrett/repository co myproj
   ```

Once the project is checked out, you can update and commit seamlessly with the remote computer. The only difference is that each time you log in, you have to make sure the **CVS_RSH** environment variable is set to **ssh**. For convenience, you may want to add the command that does this to a file that gets automatically executed every time you log in. If you are using csh or tcsh, the file is".cshrc". For bash or ksh, the file is ".profile".