

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

Part 1 of 2:
Instructions for OpenSSH version 2

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

The following illustrates how you would setup OpenSSH (version 2) to be able connect as a user from a local machine (hostA), to a user (perhaps the same or different) on a remote machine (hostB), without specifying a password, using the "publickey" authentication method. Note: ssh-agent, which is used to avoid using passphrase-less public authentications, is not discussed here, but is simple and recommended to implement.

Note: To keep things simple, we will use the default file names for the configuration, identification, and authorization files involved in this process. You should be aware, however, that these can be changed either on the relevant command line and/or within the associated configuration file for that command (e.g. ssh2, sshd2, ssh2_config, sshd2_config). Also, we will assume that the local and remote Unix users already exist. Finally, since there are multiple implementations of SSH version 2 (from different vendors, as well as from the public domain OpenSSH project), if there is any doubt as to which implementation is being used (for example on a system having multiple implementations installed), always specify the fully qualified path to the OpenSSH implementation when executing the commands below. This procedure works only for the OpenSSH v2 implementation. [[The SSH2 implementation (from www.ssh.com), for example, uses different configuration file names and directives; and therefore, using its ssh-keygen, ssh and scp commands below, will produce a result that does not work]].

Now let:

hostA = The initiating host, which does not need have to the OpenSSH daemon running (sshd), but does need to have the OpenSSH ssh and scp client programs installed.

hostA user = **userA**

hostB = The listening/receiving host, which needs to have the OpenSSH daemon (sshd) running and properly configured (explained later), as well as have the ssh/scp client programs installed.

hostB user = **userB**

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

Step1)

As (**userA**) on (**hostA**), generate a public/private key file pair using the standard name for it (i.e. **identity.pub/identity**) as follows:

```
userA@hostA$ mkdir -p ${HOME}/.ssh
userA@hostA$ chmod 755 ${HOME}
userA@hostA$ chmod 700 ${HOME}/.ssh

userA@hostA$ cd ${HOME}/.ssh
userA@hostA$ /usr/local/bin/ssh-keygen -b 2048 -t dsa \
-P "" -f identity

userA@hostA$ chmod 400 identity identity.pub
userA@hostA$ touch known_hosts
userA@hostA$ chown root:root known_hosts (do this as root)
userA@hostA$ chmod 444 known_hosts
```

The above ssh-keygen command will generate the files "**identity**" and "**identity.pub**" in **\${HOME}/.ssh**. Note that the "-P" option above notifies ssh-keygen to save the key with an empty passphrase.

Step2)

Next, copy (i.e. cut & paste) the contents of the public key file just generated above on hostA (identity.pub) into the following files on **hostB**:

```
"~userB/.ssh/authorized_keys"
"~userB/.ssh/authorized_keys2"
```

So...

```
userB@hostB$ mkdir -p ${HOME}/.ssh
userB@hostB$ chmod 755 ${HOME}
userB@hostB$ chmod 700 ${HOME}/.ssh

userB@hostB$ vi ${HOME}/.ssh/authorized_keys
<paste in the contents of userA@hostA:${HOME}/.ssh/identity.pub>

userB@hostB$ vi ${HOME}/.ssh/authorized_keys2
<paste in the contents of userA@hostA:${HOME}/.ssh/identity.pub>

userB@hostB$ chmod 400 ${HOME}/.ssh/authorized_keys
userB@hostB$ chmod 400 ${HOME}/.ssh/authorized_keys2
```

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

Step3)

Finally, ensure that on **hostB** the OpenSSH daemon configuration file, **"/usr/local/etc/sshd_config"**, has the following important directives set as shown below. After changing the configuration file (if it was necessary), be sure to restart the sshd daemon.

```
#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile      .ssh/authorized_keys
```

(and others - to be filled in later).

Step4)

You should now be able to log into userB@hostB as userA@hostA without specifying a password, by executing the command below (... note that the same applies for the initiation of SCP sessions):

```
userA@hostA$ /usr/local/bin/ssh [-v] -i ${HOME}/.ssh/identity -l userB hostB
```

Unix User Account on proxy01.prod.nymex.com (NYMEX Side)

```
longscp:x:5006:10:Longitude SCP Account:/opt/home/longscp:/usr/lib/rsh
```

Unix User Account on someserver.longitude.com (Longitude Side)

```
nymexscp:x:?????:?:NYMEX SCP Account:/foo/bar/nymex:/usr/lib/rsh
```

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

Part 2 of 2:
Instructions for SSH2 (from www.ssh.com)

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

Setting up SSH2 publickey user authentication

The following illustrates how you would setup SSH2 to be able to connect as a user from a local machine (hostA), to a user (perhaps the same or different) on a remote machine (hostB), without specifying a password, using the "publickey" authentication method. Note: ssh-agent2, which is used to avoid using passphrase-less publickey authentications, is not discussed here, but is simple and recommended to implement.

Note: To keep things simple, we will use the default file names for the config., identification, and authorization files involved in this process. You should be aware, however, that these can be changed, if you wish, either on the relevant command line and/or in the associated configuration file for that command (e.g. ssh2, sshd2, ssh2_config, sshd2_config).

Step1) Make sure that public-key authentication is enabled on both the local and remote servers, by checking that the configuration file read by the sshd2 daemon, when it was started, contained the word "publickey" in the "AllowedAuthentications" directive as shown here (or that the corresponding command line option was specified directly):

```
/opt/ssh-2.4.0/etc/ssh2/sshd2_config: AllowedAuthentications publickey
```

Note: Other (comma separated) authentication methods can be listed on the same line as well.

Step2) If the local user does not already have a public and private key pair generated for himself in \${HOME}/.ssh2, then generate one now:

```
mvega@hostA$ /opt/ssh-2.4.0/bin/ssh-keygen2 -P \  
                  -b 2048 -t dsa ${HOME}/.ssh2/identity
```

The above command will generate the files "identity" and "identity.pub" in \${HOME}/.ssh2. Note that the "-P" option above tells ssh-keygen2 to save the key with an empty passphrase.

Step3) Next, add the **private** key filename generated above to the list of identities you are known by in the "\${HOME}/.ssh2/identification" file:

```
mvega@hostA$ echo "IdKey identity" >> ${HOME}/.ssh2/identification
```

Step4) Next, copy the contents of the **public** key file on the local machine (i.e. the contents of **identity.pub**) to a file underneath \${HOME}/.ssh2 on the remote users machine. The actual filename is arbitrary but it should end in ".pub" and have a name indicative of the origins of the public key:

```
ophelia@hostB$ cd ${HOME}/.ssh2  
ophelia@hostB$ vi mvega_local.pub  
          <Paste in the contents of mvega@hostA:${HOME}/.ssh2/identity.pub>  
ophelia@hostB$ echo "Key mvega_local.pub" >> ${HOME}/.ssh2/authorization
```

Setting up SSH for publickey user authentication
(Covers both OpenSSH version 2, and SSH2 (from www.ssh.com))

Step5) You should now be able to log into ophelia@hostB as mvega@hostA without specifying a password, by executing the following command:

```
mvega@hostA$ ssh [-v] -i identification -l ophelia hostB
Sun Microsystems Inc. SunOS 5.8 Generic February 2000
No mail.
ophelia@hostB$
```