HOW-TO Use Public Keys for Secure Conections (PuTTY->Linux, X-Win->Linux, Linux->Linux)

Created by Peter Jones, last modified on Oct 21, 2016

Purpose

These are instructions I figure out over time to improve my efficiency in connection from my Windows PC to Linux-based testers, and also for tester-to-tester connections. I make use of public-key pairs for secure connections between machines.

Table of Contents

Use a Public Key from Windows::PuTTY → Linux

- Use PuTTY's puttygen to create a keypair, if a keypair doesn't already exist
  - Parameters: (√) SSH-2 RSA, bits = 4096
  - Generate → move mouse as instructed
  - Add comment to describe the key
  - Make sure you use a non-guessable passphrase, separate from your SSO, Universal Login, Linux password, etc.
    - Use PuTTY's "pageant" key-manager or equivalent (like KeePass's KeeAgent plugin) to manage the key(s)
    - If you have KeePass (http://keepass.info) & KeeAgent (http://keepass.info/plugins.html#keeagent) (*cf* KeeAgent QuickStart Guide (http://www.lechnology.com/software/keeagent/usage/quick-start/)), use KeePass to generate a hugely random password. And just don't lose your KeePass master password.
      1. KeePass
         a. Create a new password entry (example: "PUTTY KEY X")
         b. Autogenerate password with 128bit or 256bit (to make it HUGE – since you're never going to manually type this password, it's irrelevant)
         c. OK out of entry
      2. KeePass→PuttyGen = Copy this password from KeePass and paste it into puttygen's password field(s)
      3. PuttyGen
         a. Edit comment to be meaningful
         b. Copy comment
         c. Click "Save public key" → puttydir/<PASTE>.pub → SAVE
         d. Click "Save private key" → puttydir/<PASTE> → SAVE (private key will automatically end with .ppk)
      4. KeePass
         a. Edit the new entry "PUTTY KEY X"
         b. Advanced tab
            i. ATTACH, Attach File(s)...
            ii. Browse to your private-key .ppk
            iii. OPEN
         c. KeeAgent tab
            i. "Allow KeeAgent to use this entry" (aka "Entry has SSH Key")
            ii. The Attachment (aka Location) field should auto-fill to your new key. If you have more than one attachment, you may need to select the correct one from the dropdown.
            iii. OK
         d. Save KeePass database
    - Using pageant or KeeAgent will also allow FileZilla sFTP (and other compatible software) to make use of your public keys for accessing the account, so you don't need to store your passwords for those accounts in FileZilla
- Puttygen: load keypair, and copy the public key from the main window
- Paste public key into linux:~/.ssh/authorized_keys2
  - and then run
    ```
    ln ~/.ssh/authorized_keys2 ~/.ssh/authorized_keys
    ```
    so that both SSH1 and SSH2 see it
- Cf: "Key Based Authentication" (http://www.starnet.com/xwin32kb/Key_based_authentication) ...
- Cf: "X-Win32 2012 User Guide" (http://www.starnet.com/xwin32kb/sites/default/files/X-Win32_2012_User_Guide.pdf) ...

Use a Public Key from Windows::X-Win32 → Linux

- X-Config: :Security:
  - Allow By Xauth Cookie=set a path to the PuTTY Xauth file (it will create file if it doesn't exist)
  - Allow By Address=all
  - Allow By Prompt=yes
- X-Config: :Connections: TUNNEL
  - General = setup name, host, login, command=xterm
  - Advanced = Key File to the PuTTY private key (ppk); select "Send XAuth"; deselect "Disable X11 Forwarding"
- When you execute this connection, it should properly connect with XAuth, and it *should* accept

Use a Public Key from Linux → Linux

- Resources:
  - Cf: "SSH login without password" (http://www.linuxproblem.org/art_9.html)
  - Cf: "Managing Multiple SSH Keys" (http://www.robotgoblin.co.uk/blog/2012/07/24/managing-multiple-ssh-keys/)
    → for creating multiple keys and managing/configuring (something I haven't done yet)
  - Cf: "What is a SSH key fingerpint and how is it generated" (https://superuser.com/questions/421997/what-is-a-ssh-key-fingerprint-and-how-is-it-generated)
  - `foreach pub ( *.pub )`
    `ssh-keygen -lf $pub`
    `end`
    → will list each fingerprint and filename
  - `ssh-keygen -yf id_rsa`
    → will re-extract the .pub to stdout
  - `ssh-keygen -yf id_rsa > tmp ;`
    `ssh-keygen -lf tmp ;`
    `rm tmp`
    → will re-extract the fingerprint, to figure out which .pub file above matches the current id_rsa.
- **Example Procedure**: ltxmxdev3/4 share the same home directory, so they will share the same keypair; mpoc=ltxmxspc049 currently uses a separate username (my duo/quartet mpoc username) and home directory. Since you don't want to have to type a password every time, enter a blank password (hit [ENTER] when prompted).

```
Dev3% xterm -e ssh-keygen -t rsa -b 4096 -C "petercj@ltxmx.hillsboro"
 -> creates a 4096b key named "petercj@ltxmx.hillsboro" into the default id_rsa/id_rsa.pub pair
 -> used xterm so that I could get the GUI to help induce the randomness
 Dev3% cat id_rsa.pub >> authorized_keys2
 Dev3% ssh -X ltxmxdev4 echo hi
 Dev4% ssh -X ltxmxdev3 echo hi
 Mpoc% xterm -e ssh-keygen -t rsa -b 4096 -C "pjones@ltxmx-duo.mpoc"
 Mpoc% cat id_rsa.pub | ssh -X petercj@ltxmxdev3 "cat - >> ~/.ssh/authorized_keys2 "
 petercj@ltxmxdev3 password: XXXXXXXX
 Dev3% scp authorized_keys2 pjones@ltxmxspc049:/users/pjones/.ssh
 pjones@ltxmxspc049 password: XXXXXXXX
 Dev3% ssh -X pjones@ltxmxspc049 echo hi
 mpoc% ssh -X petercj@ltxmxdev3 echo hi
```

- **Example Procedure**: On a modern machine, you might be able to convert the "cat id_rsa.pub | ssh -X" into a single command – but you need to verify whether or not it exists

```
% which ssh-copy-id
/usr/bin/ssh-copy-id
% ssh-copy-id -i ~/.ssh/id_rsa.pub  username@remotemachine
```

- Permissions: after the switch to hlbontap1a for homedirectory mounting, all the permissions were messed up, and I had to do some googling to figure out which permissions are required for key-based authorization.

```
> ls -latr ~/.ssh/
```

```
total 52
rw-rr- 1 jonespet testeng 745 Jan 16 07:35 id_rsa.pub
rw------ 1 jonespet testeng 3239 Jan 16 07:35 id_rsa
rwx----- 1 jonespet testeng 10421 Jan 17 05:44 known_hosts*
rw------ 2 jonespet testeng 4121 Jan 17 06:34 authorized_keys2
rw------ 2 jonespet testeng 4121 Jan 17 06:34 authorized_keys
drwx------ 2 jonespet testeng 4096 Jan 27 10:52 ./
drwxr-xr-x 58 jonespet testeng 12288 Jan 27 11:38 ../
```

- Note that remote authorization requires go-w for ~/.ssh/and for ~/!
- No one other than user needs write permission on id_rsa, known_hosts, or authorized_keys*
- The file id_rsa.pub can have go+r, but doesn't require it

Security Concerns

- By using KeePass:KeeAgent, the Windows→Linux is just about as secure as you can get; Pageant is about as secure. You cannot get into Linux without either your KeePass:KeeAgent or Pageant password to unlock the key, or a non-key-access through your Linux password (which is no less secure than your current method)
- Once onto the secure linux network, the linux id_rsa & authorized_keys keep you from having to re-enter your password every time. However, it trusts root to not grab them
  - Make sure ~/.ssh has rwx------ permissions (chmod 700)
  - Make sure ~/.ssh/id_rsa has just u+rw,go-rwx perm (chmod 600)
  - You may set ~/.ssh/id_rsa to have global-read (chmod 644), but it's not necessary
  - Adding a password during ssh-keygen would make it secure from even root; however, it's back to adding a password.
  - You might be able to use `ssh-agent` and/or `ssh-add`
    - *cf.* "SSH Tutorial for Linux" (http://support.suso.com/supki/SSH_Tutorial_for_Linux), section "Using the ssh-agent program"
    - I see DannyW@ltxc logged in via
      `ssh-agent /usr/bin/ssh-agent /bin/sh -c exec -l /bin/csh -c "/usr/bin/dbus-launch --exit-with-session /etc/X11/xinit/Xclients"`
      so maybe change the xwin command to prefix with `/usr/bin/ssh-agent` first. That will be an experiment for another day.

howto ssh putty