Private and Public key setup for automated Secure Copy (scp)

Network Evaluation

We're Looking For Trouble

Tutorials and Guides to help enhance the integrity and security of your systems

This document is provided free of charge by Network Evaluation. The contents were contributed by members of the snort-users mailing list at sourceforge.net. Special thanks to Barry Basselgia, Andreas Östling, Eric Hines, J-H Johansen, and Demetri Mouratis. Layout, testing and verification by Network Evaluation. Please report errors, enhancements, etc. to info@networkeval.com.

Assumptions: These instructions were developed and tested on Red Hat Linux, versions 8 and 9, and Fedora 3. They may also work on other operating systems. Since the original purpose was to allow stripped-down systems (remote sensors) to use these, all instructions are based on command-line utilities. No GUI components are used.

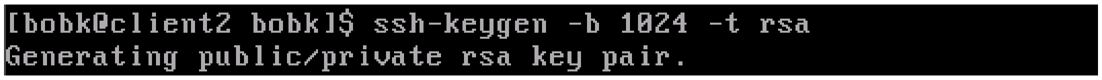
The Problem: Secure copying (using the scp utility) normally requires the use of passwords entered interactively. Neither file i/o redirection, nor Expect scripts usually work well for this purpose.

Purpose: This document is intended to aid system administrators in setting up Linux systems to allow for the automated operation of Secure Copy (scp). This would usually be to keep remote systems up to date by copying key files (you decide what those might be) over an encrypted connection to avoid tampering, spying, or interception of data.

1) Perform these instructions with a userid other than root. It may be safe enough to use root in some environments, but prudence dictates that root should only be used for those operations that actually *require* its use.

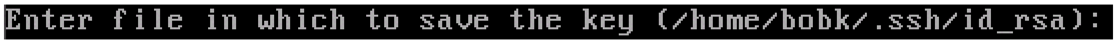
2) Make sure that each participating system has common userids other than "**root**" for this operation. We'll use "**bobk**" for our example.

3) Generate an RSA key pair on each system using the command:
   `ssh-keygen –b 1024 –t rsa`



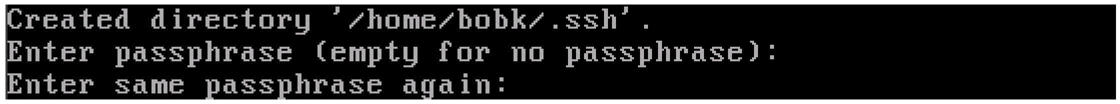
```
[bobk@client2 bobk]$ ssh-keygen -b 1024 -t rsa
Generating public/private rsa key pair.
```

4) Press <Enter> to accept the default location for the key file (**/home/bobk/.ssh/id_rsa** in this case)



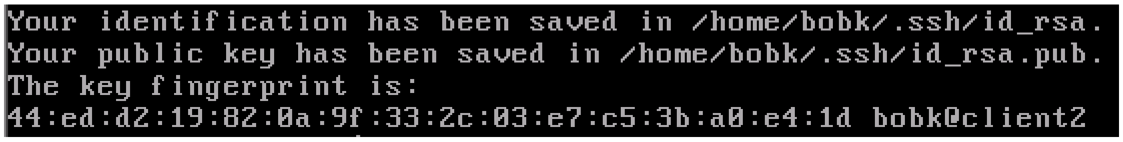
```
Enter file in which to save the key (/home/bobk/.ssh/id_rsa):
```

5) Press <Enter> for both passphrase prompts to enter an "empty" passphrase. (Specifying a passphrase makes the automation of the secure copy process more difficult, if not impossible)



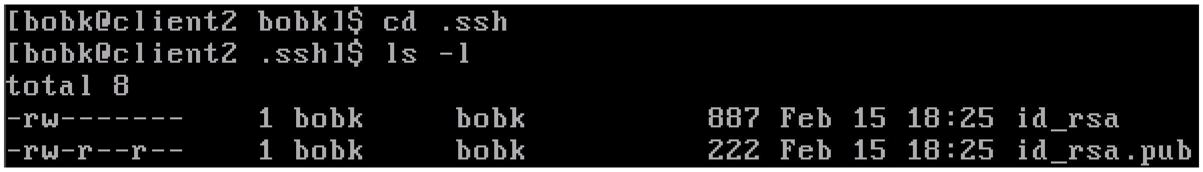
```
Created directory '/home/bobk/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

6) The ssh-keygen utility will finish the operation by informing you of the results:



```
Your identification has been saved in /home/bobk/.ssh/id_rsa.
Your public key has been saved in /home/bobk/.ssh/id_rsa.pub.
The key fingerprint is:
44:ed:d2:19:82:0a:9f:33:2c:03:e7:c5:3b:a0:e4:1d bobk@client2
```

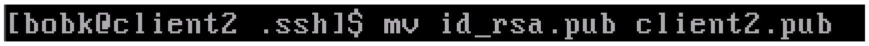
7) Change to the identification directory (/home/bobk/.ssh in this case) and verify that the files have been created.



```
[bobk@client2 bobk]$ cd .ssh
[bobk@client2 .ssh]$ ls -l
total 8
-rw-------    1 bobk     bobk          887 Feb 15 18:25 id_rsa
-rw-r--r--    1 bobk     bobk          222 Feb 15 18:25 id_rsa.pub
```

The file **id_rsa** is the private half of the private/public key set. The file **id_rsa.pub** is the public portion of this key set.

8) Rename the public key to a name that describes the system from which it originates by using the mv command: Since this system is named **client2**, then the public key file is named **client2.pub**.



```
[bobk@client2 .ssh]$ mv id_rsa.pub client2.pub
```

9) Perform all of the above steps for each computer that will be participating in the automated secure copying of files, with the relative file names for each.

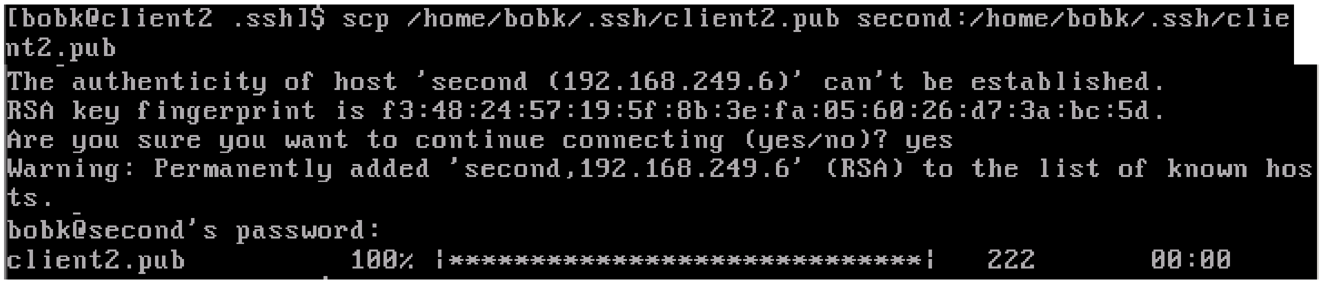
In the case of this example, there will three systems participating; One central “server” from which the new configuration files will come, and two remote systems which will perform scheduled copying of configuration files.

The server is named “**second**”, and the two remote systems are named respectively “**client1**” and “**client2**”.

Once the creation and renaming of public keys for all participating systems has been performed, then we’ll proceed to exchanging keys on relevant systems.

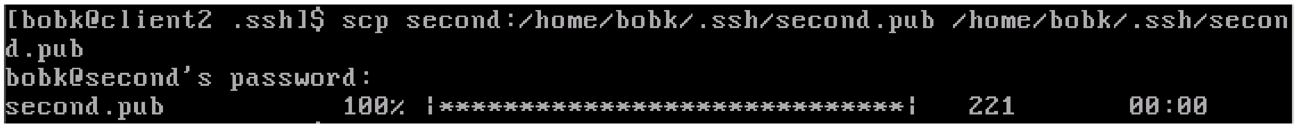
1) Manually perform a secure copy to and from each of the client systems to the server (**second**). Do not copy the client1 and client2 system’s keys to one another. While this can be done if desired, it’s not necessary for our intended operation.

2) From the command prompt, enter the following command as one line:
   ```
   scp /home/bobk/.ssh/client2.pub
   second:/home/bobk/.ssh/client2.pub
   ```
   (During the process, you will be prompted for user bobk’s password, since the keys are not yet in place.)



```
[bobk@client2 .ssh]$ scp /home/bobk/.ssh/client2.pub second:/home/bobk/.ssh/clie
nt2.pub
The authenticity of host 'second (192.168.249.6)' can't be established.
RSA key fingerprint is f3:48:24:57:19:5f:8b:3e:fa:05:60:26:d7:3a:bc:5d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'second,192.168.249.6' (RSA) to the list of known hos
ts.
bobk@second's password:
client2.pub          100% |*****************************|   222       00:00
```

3) Now copy the public key from the server (**second**) back to **client2** (in this example) and repeat steps 2 and 3 from **client1** as well.



```
[bobk@client2 .ssh]$ scp second:/home/bobk/.ssh/second.pub /home/bobk/.ssh/secon
d.pub
bobk@second's password:
second.pub           100% |*****************************|   221       00:00
```

4) At this point, each of the systems has the other's public keys for the user bobk as follows:

System Name	second	client1	client2
Second	second. pub	second. pub	second. pub
client1	client1.pub	client1.pub	**Not Used**
client2	client2.pub	**Not Used**	client2.pub

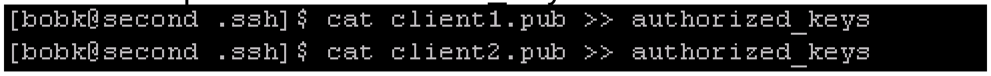
Note: Different versions of SSH may use "**authorized_keys**" or "**authorized_keys2**". We'll use "**authorized_keys**" for this example.

5) Now, the public keys on each system need to be added to a file named "**authorized_keys**" in the relative **.ssh** directories.

Note: Where there is only a single authorized system name, simply renaming the other system's public key file to "**authorized_keys**" would suffice. But to make the same procedure work across the board, we'll take a slightly different tack.

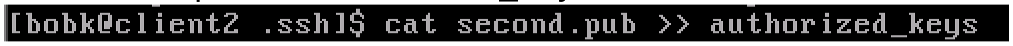
6) Change to the relative .ssh directory on each system in turn. We'll be using the server "second" for the following example:

7) Perform the following command on the server:
cat client1.pub >> authorized_keys
cat client2.pub >> authorized_keys



```
[bobk@second .ssh]$ cat client1.pub >> authorized_keys
[bobk@second .ssh]$ cat client2.pub >> authorized_keys
```

8) Perform the same relative command on each of the client systems:
Cat second.pub >> authorized_keys



```
[bobk@client2 .ssh]$ cat second.pub >> authorized_keys
```

9) To test now, perform an scp from each of the systems, copying one of the known files to a "test" filename. You should not get prompted for passwords at any time.

10) **Note:** Make sure that you have connected to each system from each system for which the automated copying will take place, since the first time the connection is made, the known_hosts file has not yet been populated, and this will prevent the automatic copying.