

This Free E–Book is brought to you by Natural–Aging.com.

100% Effective Natural Hormone Treatment
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!

Public–Key SSH Login

By Bryce Whitty

SSH is a popular system allowing a remote shell (command interpreter) to be used over a secure connection. By secure, here, I mean that the connection is encrypted, authenticated and integrity checked. The encryption prevents attackers reading the contents of the data being transmitted, the authentication allows both the client and the server to be sure that they are connected to the other, and not to some intermediate system in a man–in–the–middle attack, and the integrity checking ensures that the data is not being changed during transit. Together, these three features provide a secure connection.

Even so, the password based login feature transmits your password through this link, to the remote server, where it is hashed and compared with the stored value in the password file. To many, even though the connection is encrypted, this is not satisfactory. SSH allows the use of public key authentication to login to a server. Here, you upload your public key to the server, and keep your private key on the client machine, optionally password protected so that no one can steal your private key file and use it to gain access without a password.

Now, when the SSH connection is established, the server will need to check the authentication of the client; that is, make sure it is you logging in. This was previously done by requesting your password, and comparing it against the stored password hash. Now, the server encrypts a randomly generated token against your public key, and sends this to you. The private key associated with your public key, stored in a file to which only you have access, either by password protection, filesystem permissions or other means, is the only key able to decrypt this message. Now, your SSH client will decrypt the message and send it back to the server, which compares it against the original value. In reality, the authentication is often also checked in the opposite direction, using the server's public key, which may be stored by the client. Once the server knows you hold the private key which corresponds to the public key, it grants you access.

So, you may ask, what is the security benefit here? Well, no secret information is being transmitted. You are no longer transmitting a password, nor are you transmitting any of your private key file. You are using the keys to encrypt and decrypt a piece of random data, which works one time only. Anyone who did somehow manage to listen in on this data stream would not be able to regain access by

Public–Key SSH Login

playing back your password, or even by playing back the same data transaction, as a different value would be encrypted the next time you login, and only the private key itself can decrypt that.

Public Key authentication is supported in OpenSSH, and also in PuTTY and many other SSH systems. Check your systems documentation for details on how to use public–key based logins.

Bryce Whitty owns and runs

computer repair website

called

Technibble.com

. A website that provides

technical how–to's for repairing your computer. Technibble also has many guides for getting into the computer business

or managing your existing one. We also cover other side topics such as Security

and Software.

Running a Program on a Remote Server Using SSH

By C.S. Deam

How do you run a program on a remote server using ssh?

For this example we'll have two servers, one named Johnny and another named Cash. Both are running openssh. Our goal is to have a program on Johnny login to Cash and run a program on Cash. To make the task a little more complex we'll be using different users on each machine.

The first thing we'll need to do is generate public and private keys on Johnny. So, logged into Johnny as user 'boy' we create public and private keys by creating them in the .ssh directory as follows:

```
Johnny$> pwd
/home/boy/.ssh
Johnny$> ssh-keygen -t rsa -f sue
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in sue.
Your public key has been saved in sue.pub.
The key fingerprint is:
```

Public-Key SSH Login

```
8d:e9:c0:g1:c7:1f:e3:b3:2f:38:12:aa:b5:3b:2e:b3 boy@Johnny  
Johnny$>
```

In the example above we picked an arbitrary name, sue, to identify the files that hold the generated keys. When prompted to enter a passphrase (twice) we simply hit enter twice.

As the output of ssh-keygen indicates, the public key has been saved in a file named sue.pub.

The output of ssh-keygen implies, but doesn't indicate directly, that the private key is in a file named sue (Yes, a user named boy created a file named sue.)

```
Johnny$>ls -l  
-rwx----- 1 sue suegrp 887 Oct 17 14:27 sue  
-rwx----- 1 sue suegrp 223 Oct 17 14:27 sue.pub
```

The private key file, sue, will remain on Johnny for the remainder of this exercise, but the public key must be moved to the remote server, Cash. Note that the .ssh directory itself, as well as the sue and sue.pub files should have permissions of 700.

Now you'll need to ftp the sue.pub file from Johnny to Cash. The user on Cash that we'll login as is user 'named'.

```
Johnny$> ftp Cash  
Connected to Cash  
220 Cash - Property of Xyz. - Authorized users only  
Name (Cash:boy): named
```

```
331 Password required for named.
```

```
Password:
```

```
230-Last unsuccessful login: Fri Oct 17 13:12:55 2003 on ftp from Johnny 230-Last login: Fri Oct 17  
16:02:11 2003 on /dev/pts/1 from Johnny
```

```
230 User named logged in.
```

```
ftp> cd .ssh
```

```
ftp> mput sue.pub
```

```
mput sue.pub? y
```

```
200 PORT command successful.
```

```
150 Opening data connection for sue.pub.
```

```
226 Transfer complete.
```

```
224 bytes sent in 0.000781 seconds (280.1 Kbytes/s)
```

```
local: sue.pub remote: sue.pub
```

```
ftp>bye
```

```
221 Goodbye.
```

```
Johnny$>
```

Now we'll telnet into Cash and concatenate the sue.pub file into /home/named/.ssh/authorized_keys file.

```
Cash$> pwd
```

Public-Key SSH Login

```
/home/named/.ssh
Cash$> cat sue.pub >> authorized_keys
Cash$> exit
Connection closed.
Johnny$>
```

Let's recap what we've done so far.

- 1) We've created public and private keys on Johnny.
- 2) We've ftp'd the public key file, sue.pub, from Johnny to Cash.
- 3) We've telnetted to Cash, and concatenated the contents of sue.pub into authorized_keys

We're now ready to manually login from Johnny to Cash using ssh.

```
Johnny$> ssh -i /home/boy/.ssh/sue named@Cash
The authenticity of host 'Cash (xxx.yyy.zzz.aaa)' can't be established.
RSA key fingerprint is 65:11:7d:ef:ed:a3:cc:34:d1:b5:ba:c9:16:22:31:23.
Are you sure you want to continue connecting (yes/no)? yes
```

```
=====
*** NOTICE TO ALL USERS ***
=====
```

```
Cash$>exit
Connection to Cash closed.
Johnny$>
```

Now on Johnny, create a shell script called 'boynamedsue.sh' with the following one line of contents and chmod the script to 777. ssh -i /home/boy/.ssh/sue named@Cash /usr/bin/ls -l

Next, execute the script on Johnny.

```
Johnny$> boynamedsue.sh
```

```
=====
*** NOTICE TO ALL USERS ***
=====
```

```
-rwxrwxr-x 1 named namedgrp 10020 Oct 17 14:35 namedfile1.txt
-rw-r--r-- 1 named namedgrp 680 Aug 14 16:18 namedfile.html
-rw----- 1 named namedgrp 1148 Aug 18 09:51 mbox
drwxr-xr-x 2 named namedgrp 512 Jun 17 13:38 old
Johnny$>
```

You just executed a program on Johnny, that logged into Cash and ran a program (unix 'ls -l').

The next step you'll want to take is to replace the '/usr/bin/ls -l' command in the boynamedsue.sh program with the path and name of the program that you want to run.

Public-Key SSH Login

C.S. Deam is a small business owner. His eBook Computer Nuggets: Non-Techie Internet Tips For In-Laws, Out-Laws, and the Rest of Society is a great gift for non-techie family members and is available at

www.LinkertonPublishing.com

where you can sign up for FREE E-Courses & Newsletters

to help you on your path to self-employment.

Running a Program on a Remote Server Using SSH
Save \$100 in 5 Minutes Backing Up Your Web Site?
7 Tips for Building a Successful Downline Team
Secure E-Mail With Google GMail
Linux Runlevels

Net Spy Tracer
Super Charged Linking
Key Secrets to Setting Up Your Own Automatic \$ Making Machine!
The Toaster's Handbook
Home Vegetable Garden



This Free E-Book has been brought to you by Natural-Aging.com.

100% Effective Natural Hormone Treatment
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!