Hentzenwerke Whitepaper Series

Remote Access Via SSH

By Whil Hentzen

SSH is one of those typical "Linux mysteries" for the uninitiated. SSH provides a secure mechanism to connect to another machine over a network. This allows you to control a remote computer (such as through the command window) over the Internet without exposing your connection to other people. Here's what SSH does, why you'd use it, how it works and, most importantly, how to use it.

Hentzenwerke Publishing, Inc. • books@hentzenwerke.com • www.hentzenwerke.com

1. Preface

1.1 Copyright

1.2 Revisions

1.2.1 History

Version	Date	Synopsis	Author
1.0.0	2004/8/19	Original	WH

1.2.2 New version

The newest version of this document will be found at www.hentzenwerke.com.

1.2.3 Feedback and corrections

If you have questions, comments, or corrections about this document, please feel free to email me at 'books@hentzenwerke.com'. I also welcome suggestions for passages you find unclear.

1.3 Acknowledgments

Thanks to MLUG member Joe Baker for a great talk on SSH in 2003, the MLUG regulars at the August meeting, Ted Roche for his critical eyeballing and nudging me to include scp, and Steve Suehring for some background material.

1.4 Disclaimer

No warranty! This material is provided as is, with no warranty of fitness for any particular purpose. Use the concepts, examples and other content at your own risk. There may be errors and inaccuracies that in some configurations may be damaging to your system. The author(s) disavows all liability for the contents of this document.

Before making any changes to your system, ensure that you have backups and other resources to restore the system to its state before making those changes.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

1.5 Prerequisites

This document was written using Fedora Core 2.0 on the server and clients running Fedora Core 1.0, and assumes a beginner's familiarity with use of Linux via the GUI and the Command Window.

2. SSH definition - what is it and what does it do?

Connecting to another machine on the Internet is a risky proposition. Some tools, like ftp and telnet, that provide access to another machine transmit the username and password in clear text (meaning that they're not encrypted, and thus readable by anyone who captures them.) In some cases, this isn't a problem. For example, FTP transfers are fine for communication that doesn't have to be secure, such as if you're just downloading a set of RPMs whose authenticity can be verified separately. But this isn't acceptable if you are trying to transfer things securely.

On the other hand, telneting over a public network into a machine to set a configuration, while a common practice years ago, is now disavowed as a practice by all savvy users. This is because there are all sorts of bad guys out there with tools to capture these clear text usernames and passwords for their own unscrupulous purposes. As a result, the safe way to connect is to use a program that encrypts the communications between the two computers.

SSH is one such program. It is a software utility included with every mainstream Linux distribution, that provides a method of using a command window (or shell) on a remote machine. It's name is an acronym for “Secure SHell”. SSH has to run both on the machine you're using (known as the local machine or the client, running client software) and the machine you're

connecting to (known as the remote machine, the host, or the server, running server software.) SSH runs on the remote machine just like any other Linux service, and is initiated by you on the client in order to connect to the remote machine.

The internal behavior of SSH is to establish a secure tunnel and then pass back and forth the contents of a shell session. In fact, it gets better. SSH also will also let you share that secure tunnel with other applications that would not be secure on their own.

3. The configuration used for this discussion

I used the following components for this document.

1. A Web server running Fedora Core 2.0 with an IP address of 169.207.151.113 that is tied to the domain of www.hidbigo.com. This Web server had port 22 open in order to be able to accept SSH connections.
2. A client running Fedora Core 1.0, running on a separate network and Internet connection.

4. Using SSH

For this HOWTO, I'm going to assume you've got access to both the server for configuration purposes and then will use the client to connect to the server.

First, SSH (the server) must be running on the remote machine, and you need to know the IP address or URL of the remote machine. The remote machine needs to be able to accept requests on port 22, the port used by SSH.

4.1 Configuring the server

On the server, the first thing you should do is change a setting in /etc/ssh/sshd_config the SSH config file. The default value for

```
PermitRootLogin
```

is

```
yes
```

With this default, anyone can try to log in to the server as root, and once that's accomplished (either by knowing the root password or by guessing), they own the machine.

Change this setting to no, like so:

```
PermitRootLogin no
```

Some distributions have this setting commented out, which is equivalent to having the setting set to no. Nonetheless, it's probably a good idea to explicitly set it to no. This 'no' setting prevents someone from SSH'ing into the machine as root. Instead, they would have to SSH into the server using a different user account, and then, as that user, 'su' to become root. This means they have to know not only the root account's password, but also the username and password of another account on the machine, so you're made it harder for them to break in and gain control.

Once you made the change, you'll need to restart the SSH server. This can be done via a command in the command window or through the GUI.

In the command window, switch to root, and then issue the command

```
/etc/rc.d/init.d/sshd restart
```

In the GUI, open the Services dialog via the System Settings | Server Settings |Services menu option (you'll need to enter the root password), select the sshd service, and then click the Restart button as shown in **Figure 1**.

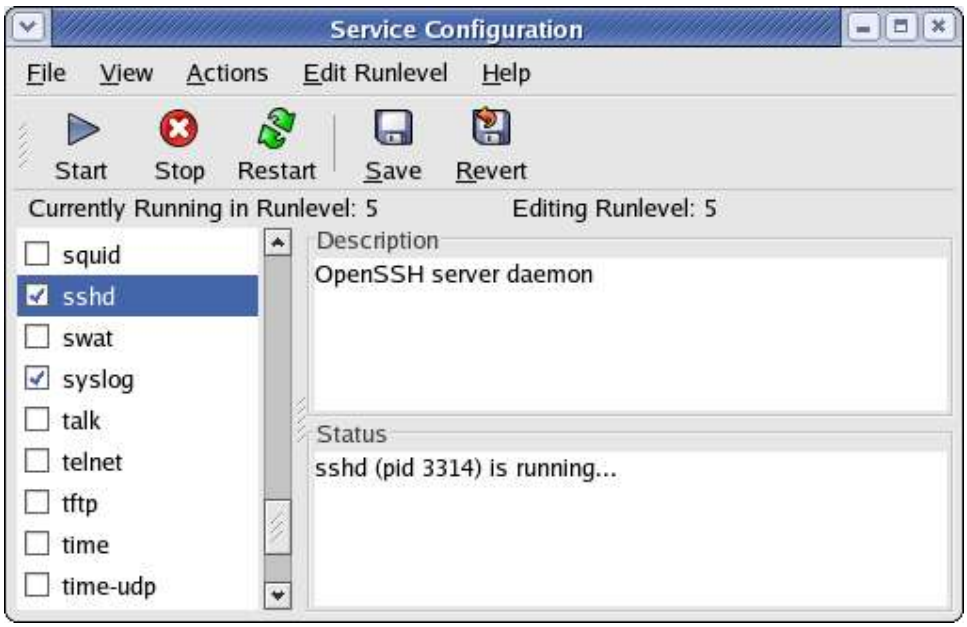

***Figure 1**. The Service Configuration dialog allows you to restart the SSH service.*

4.2 Using the client to connect to the server

Now let's go to the Linux client - the machine you're working on. Open a command window and issue the command

```
ssh
```

followed by the IP address or URL of the remote machine, like so

```
ssh 169.207.151.113
```

or

```
ssh www.hidbigo.com
```

Upon receipt of a connection attempt from a client, the two machines will undertake a handshaking process that includes the server sending it's fingerprint to the client so that the client can verify that the server is indeed who it says it is. The client will compare its own copy of the server's key (the copy is stored on the client) with the copy that the server has sent.

If this is the first time you've connected to the remote machine, you'll see a response displaying the fingerprint of the remote machine, a warning that the client doesn't know if that's the right fingerprint, and you'll be prompted to verify that the fingerprint is correct, as shown in **Figure 2**.

***Figure 2**. Upon first connection, you're asked to confirm this is the machine you want to connect to.*

The line

```
The authenticity of host 'www.hidbigo.com (169.207.151.113)' can't be established.
```

means that the client doesn't yet have a copy, and thus the client doesn't know if the fingerprint is valid or not. . The line

```
RSA key fingerprint is 8a:f8:6f:e4:1f:9a:ca:50:44:f8:83:3e:11:9e:a1:5e.
```

shows the key that the server purportedly sent to the client. It is up to the client to confirm that this key is indeed the actual key of the server. I say 'purportedly' because it is technically possible that the fingerprint that the server sent was intercepted and modified en route before its display on the client's machine.

If this connection is between machines containing highly sensitive data, you may wish to verify the validity of the key through a second channel, such as the telephone. In other words, for this first contact, you (the person in front of the client) would communicate with the owner of the server via another mechanism to find out what the real fingerprint of the server is, and compare that value with the value that was displayed on your computer screen. If they're the same, you can be sure that the fingerprint was not intercepted and modified en route.

To be pragmatic, in most situations you'll simply assume that the initial value of the fingerprint received from the server is authentic.

In both cases, assuming that the the key is good, you'll say 'yes' to the "Are you sure you want to continue connecting?" prompt, and a copy of the key will be saved on the client computer, as indicated by the "Permanently added 'www.hidbigo.com, 169.207.151.113' (RSA) to the list of known hosts." message in **Figure 3**.

***Figure 3**. SSH asks you to verify that the key the server sent to you, the client, is correct; you'll either want to independently verify the authenticity of the key, or blindly trust on the first connection.*

The list of known hosts referred to is in the file

```
/home/{username}/.ssh/known_hosts
```

and looks like this:

```
www.hidbigo.com,169.207.151.113 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAnAUoD6Jhn4KbJyAiX3oX/nR+u4BIP03ZWxll+PKLDju1n6mYAiUI6m2LK/I0fPDydWClX/w13Ip+Y/
udAN/KmFmPrrAuJIJgVn/8lSBRHLY2AsW7gKXGNnwvPrQ7O3v6+tmSTeJrvnUwSr2022rncQ+gf1Tc1rE3L/d9G+5GnXM=
```

The known_hosts file will contain a separate entry for every host (server) that the client connects to.

Note that there is a known_hosts file for each user on the client, so that the hosts that Al connects to aren't confused with the hosts that Barb connects to.

Once the value has been added to the known_hosts file, during subsequent connections, the value received from the server will be compared with the value stored on the client. If the values differ, it could mean that the server has been modified, such as through an update of the ssh program, or that the server has been attacked and compromised. Regardless, in the event of the value has changed, you'll want to investigate why before continuing with the connection.

Upon successful entry of the password, the connection will be closed, as shown in Figure 3, but the client will now have a record of the server. Issue the 'ssh' command again with the IP address or URL, and you'll immediately be prompted for the password this time, as shown in **Figure 4**.

***Figure 4**. After successfully supplying the password to the remote user account, you'll see a command prompt that confirms you're connected remotely.*

Upon successful entry of the password this time, you'll get a command prompt for the user on the server, and you can now issue commands to the server just as if you were sitting in front of the machine itself.

At this point, it's appropriate to discuss the user accounts and passwords more explicitly. When you're logging on to the remote box, you're logging onto an account on that machine, and thus, you're being prompted for the password of the client's currently logged on user – but on the server machine. In other words, if you're logged in as 'alvin' on the client machine, there will need to be an 'alvin' account on the server as well. Enter the password for the 'alvin' account on the server.

If you, the reader, try to log in to the sample domain here, www.hidbigo.com, you'll be stymied unless you know a user account and the password associated with that account.

Once you're done with your session on the remote machine, type

```
exit
```

in the command window, as shown in **Figure 5**.

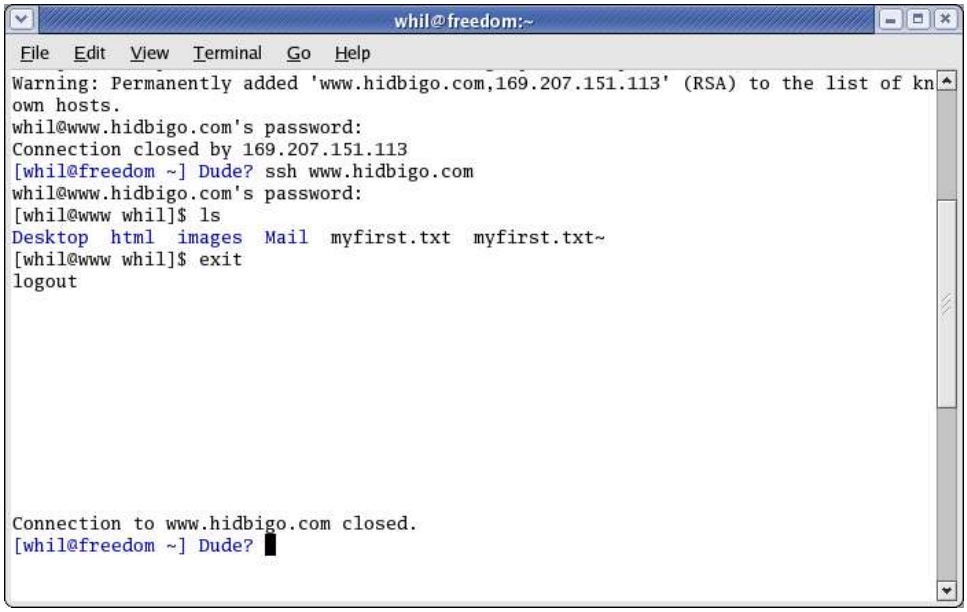

***Figure 5**. Logging out of the SSH session restores your command prompt to its previous value.*

After a few moments, the server will respond with the logout command and then the client will confirm the closure of the connection as well. The command window on the client will be returned to the currently logged in user on the client.

5. Advanced Functions

After you've used SSH a bit, you'll want to do a few more things.

5.1 Become root on the remote machine

You can become root on the remote machine by issuing the 'su' command in the command window once you're connected to the remote machine. You'll need the password of the root user on the remote machine, of course. When you're done with your root work on the remote machine, issue the

```
exit
```

command and you'll be returned to the regular user who was logged onto the remote machine. For example, if alvin had logged onto the server, the prompt in the command window would say

```
[alvin@remote alvin]$
```

After alvin then su'd to become root, the prompt would say

```
[root@remote alvin]$
```

or if avlin did 'su -', the prompt would say

```
[root@remote ~]$
```

where the ~ denotes root's home directory, because the '-' incorporates the root user's environment and switches the current directory to the user's home directory, which would be /root in this case.

5.2 Logging in as a different account

If you want to log in as a specific user on the remote machine in a minimum number of keystrokes, you can do it in one of two ways. If you wanted to log in as the user bob on the remote machine, for example, you could

```
ssh -l bob www.hidbigo.com
```

or

```
ssh bob@www.hidbigo.com
```

where '-l' is the letter "ell", and where you could use the IP address instead of the URL if you wished.

5.3 Testing SSH on a single machine

If you don't have a pair of machines to experiment with, you can log in to the local machine, using '127.0.0.1' or 'localhost' as the IP address or URL, like so:

```
ssh localhost
```

or

```
ssh 127.0.0.1
```

5.4 Restarting SSH remotely

You can restart SSH remotely by logging onto the remote machine via SSH?, changing to the root account, and issuing the restart command. If you forget to switch to root first, you'll get a pair of errors, as shown in **Figure 6**.

***Figure 6**. You can't restart SSH on the remote machine unless you're logged in as root.*

Figure 7 shows a successful restart.

***Figure 7**. When you are logged in as root on the remote machine, you can restart SSH.*

5.5 Using scp (secure copy)

So you can make a secure connection to another machine, and then run programs on that remote machine from the comfort of your own easy chair. What else might you want to be able to do? Copy files back and forth, just as if that remote machine was just another locally accessible share. Why wouldn't you just use the 'cp' command? Because when you're on the remote box, 'cp' no longer knows anything about the client machine. When you're connected to a remote machine via ssh, the command window you're using is for all practical purposes a command window on that remote box, just as if you were sitting in front of it. Only you aren't. And so, if you were in front of the remote machine, you wouldn't be able to use the cp command to copy files back to the client box back on your desk, right?

Enter scp, the secure copy command. scp uses the SSH secure tunnel to transmit files in both directions.

scp, however, knows about both the client and the server – about both the local machine and the remote box. Here's how it works. Suppose you are on a local box, and you want to transfer a file from your local box, say, a new HTML page named itemlist.txt, to the remote machine, and keep the same name. Issue the scp command:

```
[whil@freedom ~] Dude? scp ./itemlist.txt whil@www.hidbigo.com:/home/somedir
whil@www.hidbigo.com's password:
itemlist.txt                                  100%  191     3.4MB/s   00:00
[whil@freedom ~] Dude?
```

Note that the syntax of scp is similar to cp: command, source, and then target. When you are describing the remote machine, however, you need to include more than just the filename – you need to include the name of the host and the user you're connecting through (just like an ssh command) as well.

You'll be asked for whil's password on the remote machine, and assuming success in that area, voila, the file will be copied through the secure SSH tunnel.

6. Where to go for more information

The ssh command has both man and info pages (type “man ssh” and “info ssh” in a command window) as part of its inclusion in a Linux distribution. In addition, you may find the following Web sites useful.

http://www.suso.org/linux/tutorials/ssh.phtml

http://wks.uts.ohio-state.edu/sysadm_course/html/sysadm-558.html

http://www.openssh.com/

7. About the author

Whil Hentzen started out life in the early '80's as a custom software developer using dBASE II (he still has the original 8 1/2 x 11 grey binder of documentation, much to the chagrin of his wife), and switched to FoxPro in 1990. Besides billing 15,000 hours in the 90's, he presented more than 70 papers at conferences throughout North America and Europe, edited FoxTalk, Pinnacle Publishing's high end technical journal for 7 years, hosted the Great Lakes Great Database Workshop since 1994. He's written 7 books and published 30 more on a variety of software development topics. He was a Microsoft Most Valuable Professional from 1995 through 2003 for his contributions to the FoxPro development community, and received the first Microsoft Lifetime Achievement Award for Visual FoxPro in 2001.

Whil began using Linux on the desktop when OpenOffice.org became a standard in the mainstream distributions, as it spelled potential for custom application development in the future, and has been a Linux user, developer, and evangelist ever since. His first book on Linux, Linux Transfer for Windows Power Users, was published in early 2004.

He is available for new and legacy Visual FoxPro application development as well as Web and desktop development on Linux.

8. A word from our sponsor

This free whitepaper is published and distributed by Hentzenwerke Publishing, Inc. We have the largest lists of "Moving to Linux", OpenOffice.org, and Visual FoxPro books on the planet.

We also have oodles of free whitepapers on our website and more are being added regularly. Our Preferred Customer mailing list gets bi-monthly announcements of new whitepapers (and gets discounts on our books, first crack at special deals, and other stuff as we think of it.)

Click on "Your Account" at www.hentzenwerke.com to get on our Preferred Customer list.

If you found this whitepaper helpful, check out these Hentzenwerke Publishing books as well:

Linux Transfer for Windows® Network Admins:
A roadmap for building a Linux file and print server
Michael Jang

Linux Transfer for Windows® Power Users:
Getting started with Linux for the desktop
Whil Hentzen