



Remote Administration of Windows Systems with SSH

Published by the Open Source Software Lab at Microsoft. April 2008.

Special thanks to Chris Travers, Contributing Author to the Open Source Software Lab. Most current version will be maintained at <http://port25.technet.com>.



Abstract:

SSH has largely replaced Telnet for remote administration of UNIX and Linux systems, but has not yet been used much on Windows. SSH is generally considered to be more secure than Telnet and the Berkeley remote commands (rlogin, etc). This paper uses SSHWindows, a minimal package of Cygwin and OpenSSH. It is available from <http://sshwindows.sourceforge.net>. The paper is written such that an average Windows system administrator can get an SSH server up while understanding how to make use of security features.

Information in this document, including URL and other Internet Web site references, is subject to change without notice and is provided for informational purposes only. The entire risk of the use or results from the use of this document remains with the user, and Microsoft Corporation makes no warranties, either express or implied. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

© 2008 Microsoft Corporation. This work is licensed under the Microsoft Public License. The Microsoft Public License is [available here](#).

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Microsoft, Windows, Windows XP, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are property of their respective owners.

1.1.1 A Note about Cygwin

SSHWindows includes a minimal installation of Cygwin. Although Cygwin performs a similar function to the Subsystem for UNIX Applications (SUA) in Windows, these technologies are not the same. Cygwin essentially emulates a POSIX environment on top of the Win32 subsystem, while SUA provides a native POSIX environment which runs in parallel to (rather than on top of) the Win32 subsystem. For this reason, OpenSSH may not perform as well as it might if it was compiled on SUA. It can also be installed on platforms that SUA does not support.

1.1.2 SSH vs telnet and rlogin

The original telnet and rlogin protocols were designed before network eavesdropping was a significant concern. As a result, these protocols tend to pass potentially sensitive information (such as passwords and session data) across the network in plain text.

The MIT Kerberos project added encryption for these protocols. However, the decision was made to use strong authentication, mutual authentication of the server and the client, and session encryption through policies in the server configuration files. Improperly configured servers, thus, were insecure.

Kerberos authentication is somewhat easier to manage over a large network than native SSH means. Because many SSH implementations did not support Kerberos authentication properly, it was common to choose between Kerberized telnet and SSH for remote authentication. However, today, OpenSSH and many other implementations support GSSAPI (Kerberos ticket) authentication. As a result, there is little reason not to use SSH as a standard protocol for remote command-line access.

1.1.3 Understanding SSH Security

SSH provides a number of security features designed to protect the authentication process. None of these measures is perfect, but they form a multilayered system of protection.

Remote Host Authentication: When a connection is requested, and before user authentication is sent, a public key exchange is used to initiate both a key exchange and a host authentication with the SSH server. This state is somewhat vulnerable to man-in-the-middle attacks on the first attempt to access a specific server, but is reasonably safe for subsequent connections since the server's public key is cached on the client.

User Authentication: Users may be authenticated using any number of methods including passwords, public key authentication, and (in many implementations) GSSAPI/Kerberos tickets. Due to the way that password-based authentication occurs, the password must be sent using a method where decryption is possible. OpenSSH uses triple-DES by default for this function but rotates keys periodically to prevent eavesdropping.

Session Encryption: The entire session is encrypted using a symmetric cypher with keys regenerated at specific intervals. OpenSSH defaults to triple-DES.

1.1.4 Key Management Basics

Note that actual use of user-based authentication keys is beyond the scope of this paper in part because different implementations have different formats for storing keys, and because different versions of the same software may store the authorized keys in different locations. However, although the use of these keys for user authentication is not required for basic use, the theory is covered because it is the source of a great deal of confusion among SSH administrators.

SSH uses public key authentication of the server and supports similar authentication for the client. Most SSH-related security compromises (aside from brute force attacks via passwords) I am aware of occur because of a lack of attention to key management. Hence this topic is extremely important for SSH administrators on Windows and Linux to understand.

Public key encryption uses two keys with the assumption that one cannot reasonably derive one key from the other. One key is kept secret and is called the "private key." The other can be freely shared (and is called the "public key").

If we have a public key that we know should be associated with a given user or computer, we can use this encryption technology to generate digital signatures which verify the identity of the other party (whether a computer or a user). Note that the important assumptions are that we have the public key and that we trust that it is associated with a given user or computer. Also note that no information relating to the private key is passed between the parties.

As mentioned previously, SSH performs a host key verification before initiating authentication. This exists to prevent a server from misrepresenting its identity. If the server's public key is compromised, or if the client does not have the key for the server cached, it is possible for an attacker to impersonate a server. With password authentication, this allows an attacker to intercept passwords.

If the host key is compromised, this has a similar effect to compromising a password, but if the key is not known previously, the client will usually accept the key from the host and trust it on the first connection. This means that the host key merely verifies that the server appears to be the same system on each connection. Once again, if the server can be impersonated and password authentication is allowed, passwords can be harvested for later unauthorized use.

Similarly, a user key, if compromised can provide a means of unauthorized access to a server, as it functions essentially like a stored password. Some degree of protection exists by encrypting all user keys with passphrases, but the server has no way of enforcing this because no essential information to the key is transferred to the server.

No SSH implementations that I know of support standard trust-based public key infrastructures (such as X.509) for credentials used from outside a controlled network (Kerberos is designed only to be used inside such a network). However, when Kerberos is used, most of the above considerations are substantially mitigated. Setting up SSH with Kerberos is beyond the scope of this paper, however.

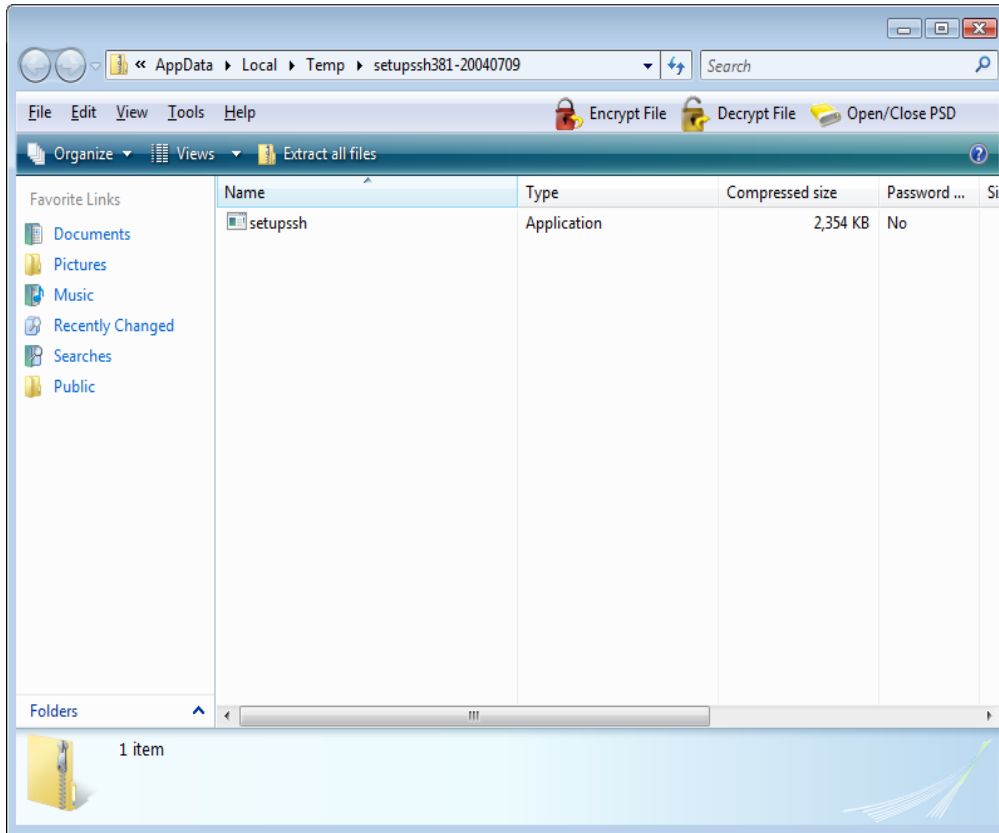
1.1.5 A Note about Vista

I was unable to get this software to run on Windows Vista. I believe that the issue may be related to the version of Cygwin bundled with the software. The software does install and the configuration commands do work but the service will not start. I would expect this to be fixed in a later release. For this reason, I have documented what is necessary to get the software to install (though not run) on Vista. As new versions are released, your mileage may vary.

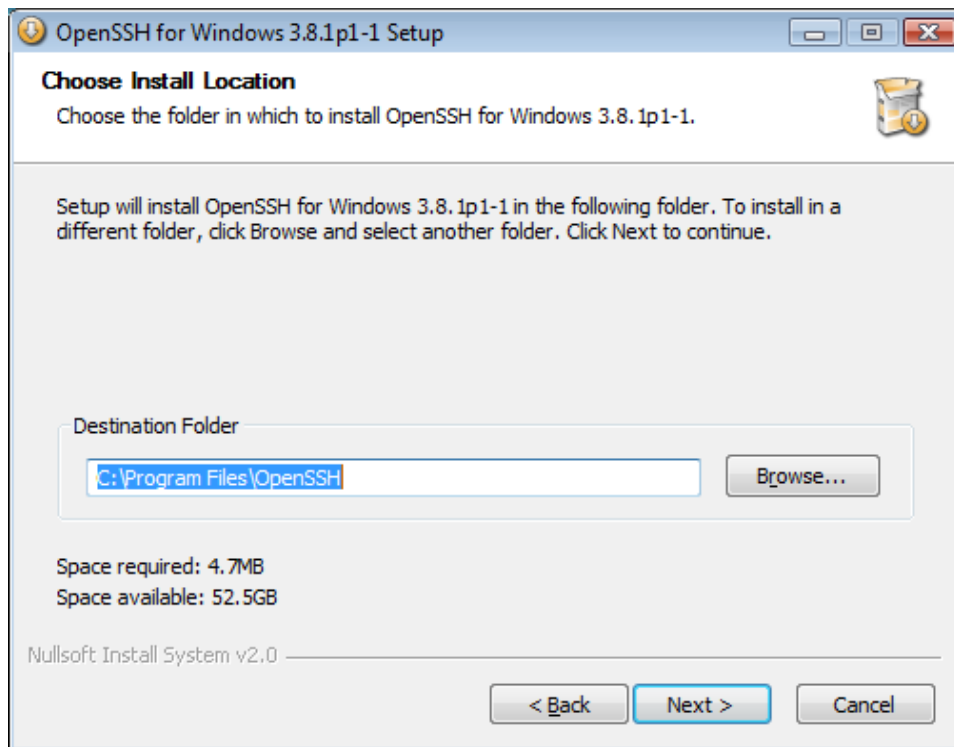
Other versions of Windows work properly with these steps. The installation on Server 2003 was painless, and I was able to connect from a Linux system quite easily.

1.2 Installing the Software

SSHWindows is available of a .zip file containing an installer, which is somewhat unconventionally named setupssh. This file can be run directly from the archive.



Most of the installation is pretty standard. By default, the software is installed to C:\Program Files\OpenSSH and takes up 4.7 MB disk space. The location can be changed in the screen below.



1.3 Configuring the Software

Once the software is installed, it must be configured so that it can run. This step may seem somewhat foreign or awkward to a Windows administrator because this is a minimal port from Linux/UNIX to Windows and thus requires certain files be in place.

1.3.1 Creating User and Group Maps

Open a command prompt window. Use the following command to get into the proper directory (assuming you have installed SSHWindows into the default location and your HOME directory is on the system drive):

```
C:\Documents and Settings\Administrator>cd "%Program Files\OpenSSH\bin"
```

Note that the commands below use the >> operator instead of the > operator.

Then create the group mappings (for local groups):

```
C:\Program Files\OpenSSH\bin>mkgroup -l >> ..\etc\group
```

Optionally, you may want to create domain group maps too:

```
C:\Program Files\OpenSSH\bin>mkgroup -d >> ..\etc\group
```

Next create your user mappings (for local users):

```
C:\Program Files\OpenSSH\bin>mkpasswd -l >> ..\passwd
```

If you want to create a mapping for domain users, change the -l to -d:

```
C:\Program Files\OpenSSH\bin>mkpasswd -d >> ..\etc\passwd
```

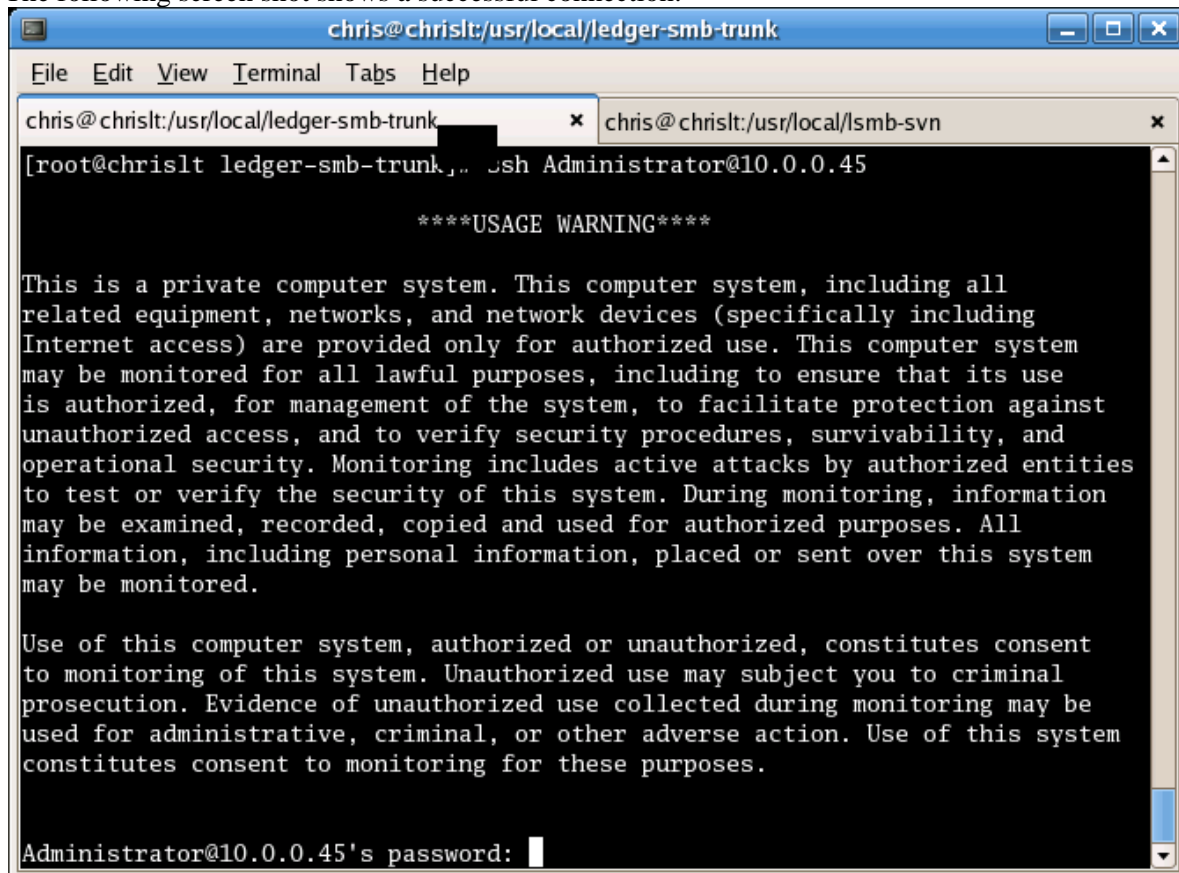
Next, start the service:

```
C:\Program Files\OpenSSH\bin>net start opensshd
```

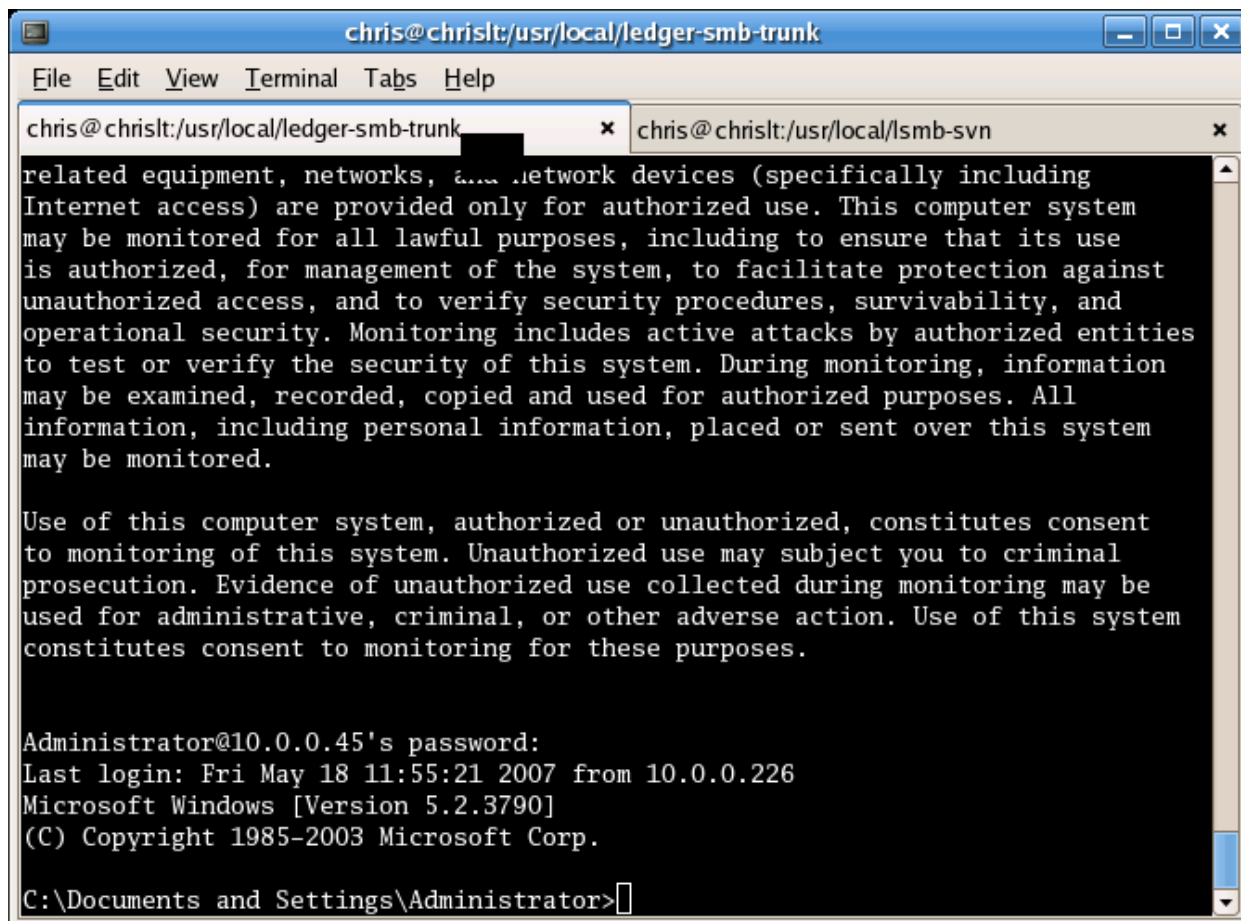
Finally, you can test this from a Linux system:

```
bash$ ssh Administrator@10.0.0.45
```

The following screen shot shows a successful connection:



Once you enter the appropriate password, you should see a DOS prompt:



```
chris@chrislt:/usr/local/ledger-smb-trunk
related equipment, networks, and network devices (specifically including
Internet access) are provided only for authorized use. This computer system
may be monitored for all lawful purposes, including to ensure that its use
is authorized, for management of the system, to facilitate protection against
unauthorized access, and to verify security procedures, survivability, and
operational security. Monitoring includes active attacks by authorized entities
to test or verify the security of this system. During monitoring, information
may be examined, recorded, copied and used for authorized purposes. All
information, including personal information, placed or sent over this system
may be monitored.

Use of this computer system, authorized or unauthorized, constitutes consent
to monitoring of this system. Unauthorized use may subject you to criminal
prosecution. Evidence of unauthorized use collected during monitoring may be
used for administrative, criminal, or other adverse action. Use of this system
constitutes consent to monitoring for these purposes.

Administrator@10.0.0.45's password:
Last login: Fri May 18 11:55:21 2007 from 10.0.0.226
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>
```

1.4 An Overview of Windows SSH Clients

Windows also has a number of ssh clients which are available for use. The SSHWindows package comes with a command-line ssh tool ported, using Cygwin, to Windows. Typically, it is invoked as:

```
C:\Program Files\OpenSSH\bin>ssh user@host
```

Other commonly used tools include:

PuTTY: This is a standard client with a graphical front-end for setting up sessions. It also includes command-line scp and sftp clients. It has its own format for user keys.

WinSCP: A graphical front-end for securely transferring files to and from the remote system using the SCP extension of the SSH protocol. It uses PuTTY's key storage format.

Cygwin with OpenSSH: Cygwin is a distribution which allows one to run applications written for POSIX environments on nearly any Windows platform. It is similar to the system documented here except that it includes a greater number of UNIX-like components. Cygwin does not perform as well as SUA though they fill similar functions.

Any of the above clients can be used to connect to Windows or Linux SSH servers with equal ease. The interfaces vary but in general, all that is required is entering a hostname or address, a username, and a password. Note that the command line tools (and also PuTTY) generally ask for the password separately in order to prevent eavesdropping on the program.

1.5 Final Thoughts

SSH is not as useful on Windows as it is on Linux, in part due the differences between how remote access to graphical applications is handled, and in part due to the fact that Windows is not generally as command-line oriented as Linux. However, it is nonetheless a very important tool with regard to remote management. Advanced topics, not covered in this paper include:

- Use of public keys instead of passwords for user authentication.
- Creation of SSH-protected network tunnels for allowing remote access to specific network services through an encrypted tunnel.
- Use of Kerberos as an authentication method for SSH.
- Leveraging other Public Key Infrastructures (such as X.509) from SSH.

Each of these areas provides an opportunity to leverage SSH in more diverse environments as one tool in a network security toolkit. Thus this paper, while providing a reasonable introduction to the installation and basic use of SSH on Windows, is nothing more than a beginning.

1.6 About the Author

Chris Travers is the owner of Metatron Technology Consulting, a firm devoted to helping customers leverage open source solutions on any platform. Over the past ten years, he has worked with various flavors of UNIX, Linux, and Windows.