

Securing Windows Environments with SSH

CONTENTS

SSH Basics	1
Running SSH on Windows Servers	1
Using SSH for File Transfer	3
Using SSH for Remote Administration of Windows Desktops	3
Securing RDP Through an SSH Tunnel	3
SSH and Windows: A Secure Partnership	4
SSH Products from Attachmate	4

Securing Windows Environments with SSH

Secure communication is taken for granted in the UNIX world. Thanks to the powerful SSH (Secure Shell) protocol, UNIX administrators have long been able to manage their servers remotely, automate tasks with remote scripting, run batch jobs, control configuration settings, and copy data or files across untrusted networks—all without jeopardizing security.

But until recently, the same could not be said about the Windows world. That's because Windows lacks a scriptable remote secure shell capability and instead has offered nonsecure tools, such as RDP, which transmit data in the clear. And remote management packages, which are typically bulky to install and run, seem excessive when all that's really needed is a simple, secure command line prompt.

As Windows servers become more prevalent and are used for more of the data center's heavy lifting, running SSH on those servers makes more and more sense. For one thing, SSH gives administrators a secure, light-footprint way to perform routine maintenance tasks, such as moving a file or restarting a service—no matter where they happen to be. SSH on Windows servers also allows an organization with a mixed environment to establish a uniform security policy.

This white paper outlines the primary functions of SSH and describes how it works with Windows servers. The Attachmate® SSH solution, Reflection® for Secure IT, is referenced throughout the paper for demonstration purposes. By the time you're done reading, you'll know the most common uses of SSH and how to creatively use it in Windows environments.

SSH Basics

SSH is a program built to log in to another computer over a network, execute commands on that remote machine, and move files from one machine to another—while providing strong authentication over a nonsecure channel. SSH was originally designed to replace standard UNIX communications protocol applications, such as Telnet, rlogin, rsh, and rcp. As SSH has evolved and been ported to platforms other than UNIX, it has become the standard tool for the following critical tasks:

- **Secure remote logins**

Traditional protocols such as Telnet and FTP, transmit user names and password data in clear text over the network, making them easy prey for third-party interceptions. By running SSH instead, you authenticate yourself to the remote computer's

SSH server through an encrypted connection. The entire session is secure, and the authenticated user cannot tell the difference because the encryption is transparent.

- **Secure remote administration**

Telnet is a widely used protocol for the remote access and administration of UNIX servers. It has been used on Windows as well. Telnet offers no security against eavesdropping or password theft and provides only plain-text password authentication. On the other hand, SSH offers secure remote access and administration of servers, including various user authentication methods.

- **Secure file transfers**

FTP is commonly used for file transfers in heterogeneous environments. As a standard component of every operating system, including Windows, FTP is the tool most commonly used to transmit files larger than 5 MB that do not fit into e-mail. But FTP is problematic because it doesn't encrypt data (including user names and passwords) sent from one computer to another. As a result, packets can be easily intercepted en route and read. Passwords also need to be programmed into scripts that automate file transfers.

With SSH, you can easily and securely automate file transfers. SSH encrypts files prior to transmission and then decrypts them after they arrive at their destination. In addition to providing ironclad user authentication, SSH doesn't require scripted passwords.

- **Secure remote command execution**

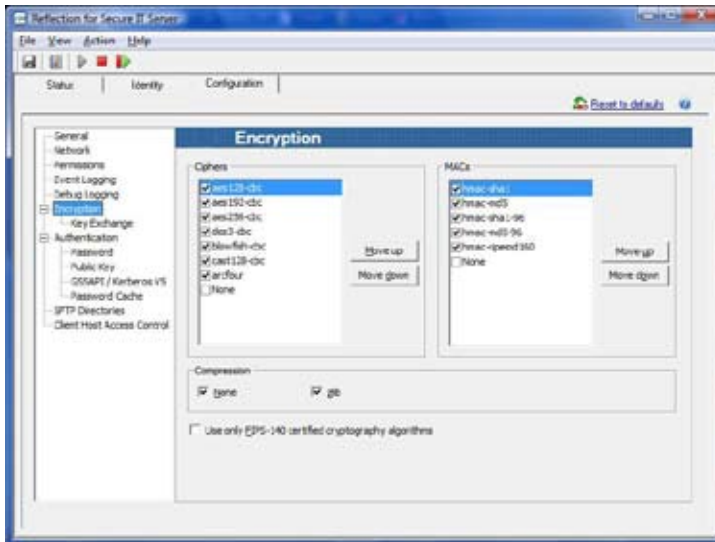
System administrators frequently need to run the same command on multiple computers. But in many situations they do not want these commands to be transmitted in plain text. SSH encrypts all commands for travel across the network.

The SSH protocol also lets you establish secure connections with other programs or protocols through the use of port forwarding. This means that protocols such as RDP (Remote Desktop Protocol) and applications such as printing can be run through a secure channel.

Running SSH on Windows Servers

To safeguard remote commands and file transfers in the Windows environment, SSH must be installed on the Windows server. You can then connect securely

to your Windows server with an SSH client utility. Reflection for Secure IT Windows Server is an SSH server product designed for Windows servers; Reflection for Secure IT Windows Client provides the client utilities that allow you to remotely administer and transfer files to your Windows server. Once SSH is installed on the Windows server, you can configure the server using the graphical configuration program provided by most SSH products (see screen shot below) or by editing the configuration file (sshd2_config) with a text editor.



Using the graphical configuration program provided by Reflection for Secure IT, you can easily configure your encryption settings.

Here are some configuration options to consider:

Authentication

Your choice of authentication options will vary by SSH server. For example, Reflection for Secure IT offers password (local or Windows domain), user public key, GSSAPI, and keyboard interactive. Reflection for Secure IT also lets you choose between giving users access to local or domain accounts, or limiting their access to specific folders after authenticating.

Some SSH servers, including Reflection for Secure IT, allow you to strengthen security by requiring multiple types of authentications, for example, adding the user public key method to password authentication.

Once you've chosen your configuration options, the server advertises what it can support and the client attempts to authenticate with methods that have been configured in a particular order of priority.

Encryption

A number of encryption methods can be used by SSH clients and servers. Depending on your security and performance requirements, you may wish to configure different algorithms.

* On a Windows 2000 server, it is best to have installed Windows 2000 with the support tools for this command.

Command line functions

Once the server is started and the client connects, the user will be at the command prompt in Windows. From there, the user will have secure access to any of the command line functions that Windows supports:

- dir, del, copy, move, type, mkdir: common file management commands.
- ps, pstat, netstat, ping, ipconfig, shutdown, net view, passwd, debug: common administrative commands.
- cacls: displays or modifies access control file lists.
- net start/net stop: starts/stops a service.
- net use g: \\server\share /USER:DOMAIN\hessu: maps a drive to a share.
- net user add: adds a new user account to the account database.
- whoami: shows who you are*.
- tlist: shows the list of processes (tlist -t gives more detail and shows the parent- child hierarchy of the processes)*.

- kill: kills a process*.

Command shell (cmd.exe)

In most Windows 2003 configurations, access to command shell is restricted to administrators, members of TelnetClients group, and fully authenticated users (users who are logged on with a local password). This is an operating system restriction, not an SSH server restriction.

It is the administrator's responsibility to define a security policy for accessing the command prompt through SSH. For example, the administrator might define the following security policies in the Reflection for Secure IT Windows server configuration tool (group restriction and user restriction sections):

- Allow only members of the TelnetClients group to access the server through SSH (use AllowGroups TelnetClients).
- Add temporary membership in the TelnetClients group to all SSH users who do not authenticate by password (use AddGroupsToToken TelnetClients).
- Allow only administrators to access the command shell (PermitUserTerminal admin).

DID YOU KNOW?

There are two versions of the SSH protocol. The older SSH-1 protocol has some technical limitations that make it less secure than the newer SSH-2 protocol. SSH-2 has become the de facto standard and is going through the RFC process in the Internet Engineering Task Force.

- Create a group, SSHUsers, to control access to SSH, but allow SSHUsers a .temporary membership. to the TelnetClients group (AllowUsers SSHUsers, AddGroupsToToken TelnetClients).

The administrator may also want to configure Windows 2003 security settings in the Security tab on the Properties dialog for cmd.exe:

- Keep default settings. SSH server does not do any additional processing, meaning that only members of the TelnetClients group, administrators, and users who offered a password can use the command shell.
- Remove all restrictions by manually granting access to cmd.exe to Users (or Everyone, or SSHUsers).

When the SSH server software is started, it begins to listen on a port for a socket. The default port is 22, now a well-known port for SSH. This port can be changed to suit any custom environment.

When the server is listening for a socket, it waits until a client initiates a socket connection. Once connected, the server creates a child process. The child process handles the actual connection with the client, including authentication, supported cipher negotiation, and termination of the connection. In many SSH servers, an additional child process, running at the privileges of the authenticated user, is created for the command sessions, data file transfers, and forwarded TCP/IP ports.

After the connection terminates, the child process terminates as well. The parent process remains listening for other connections until explicitly stopped.

Using SSH for File Transfer

Native Windows file sharing may be sufficient for some internal file copying needs, but when you need to copy files to a remote web server, to a third-party supplier or vendor, or between Windows and UNIX platforms, a more robust solution is required.

Not only does SSH give Windows administrators a secure way to execute remote commands for server management, but it also provides two standards-based secure file transfer protocols: SFTP and SCP. Although both protocols are cross-platform mechanisms that can move files securely across untrusted networks, they do have definite differences.

SCP is an older protocol, available with both SSH-1 and SSH-2. It is a useful command line tool, meant strictly for copying files or running batch files and

other automated processes. It has no file management capabilities and no graphical user interface. However, many commercial SSH clients and servers have updated the SCP protocol to run through the SFTP protocol to improve security.

The newer SFTP is available only with the SSH-2 version of the protocol. It can operate either via the command line or with a graphical front end. SFTP provides file management capabilities—such as copy, move, delete, rename and change permissions—and runs as a subsystem under SSH.

Using SSH for Remote Administration of Windows Desktops

Sometimes administrators want simple command line access to remote desktops. For example, perhaps there are many remote locations with low-bandwidth connections, so that bringing up remote control or a GUI is not efficient. Or maybe the administrator just wants an unobtrusive way in which to accomplish a quick task in the background.

The problem is that most command line tools for Windows—for example, Telnet, rcmd, and rclient—are not secure. They send passwords in plaintext, and all interaction between client and server is transmitted across the network in the clear.

SSH is an ideal replacement for these command line tools. The SSH server is lightweight and can be easily loaded onto Windows desktops. Administrators can then connect using an SSH client and do their work securely.

In many organizations, the policy is this: When going into any system through the command line, use SSH.

Securing RDP Through an SSH Tunnel

SSH performs port forwarding so that other protocols can be run securely through an SSH tunnel. This SSH capability allows you to route all your traffic through one secured port in the firewall (similar to the way SOCKS works, except with encryption).

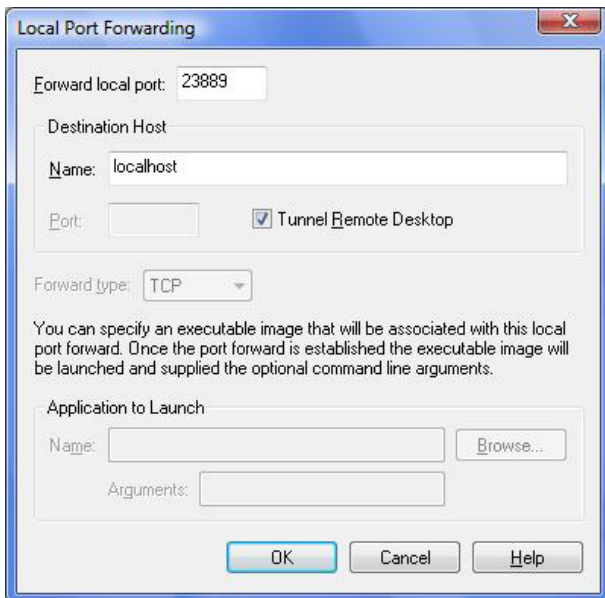
One of the protocols that can be run through an SSH tunnel is RDP (Remote Desktop Protocol). RDP is the protocol underlying what Microsoft calls Remote Desktop in Windows Server 2003 and Terminal Services in Windows 2000 Server. It lets you remotely log on to a Windows machine and work as if you were seated at the local console.

Remote Desktop can protect against passive attacks, but not active attacks. It also requires opening another port in the firewall—usually port 3389—which most

security-conscious organizations are loath to do. Running RDP through an SSH tunnel can mitigate these security problems.

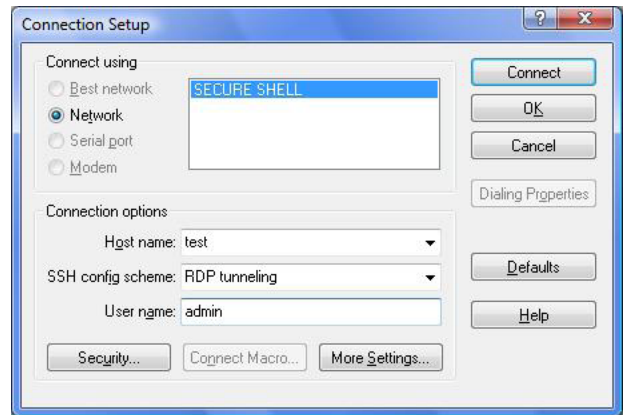
To get started, install your SSH server on the target server and run your SSH client on the administrator's workstation. Tunneling RDP is easy to set up through the Reflection for Secure IT Windows client. Just follow the steps below:

1. Enable tunneling on your SSH server.
2. Enable RDP on the target machine.
3. Open the SSH client.
4. Go to the Connection menu, select Connection Setup, and click the Security button.
5. Click the Tunneling tab.
6. Under Local Forwarding, click the Add button. The following screen appears:



7. Under Destination Host, select the Tunnel Remote Desktop check box.
8. Enter an arbitrary port (with a number greater than 1024) on your local machine through which any data sent will be tunneled.
9. Under Destination Host, enter localhost in the Name field.

10. Click OK. The Connection Setup screen appears:



11. Click Connect.
12. When prompted, enter your credentials to authenticate to the tunnel and then to the remote host.
13. Your Remote Desktop/Terminal Services session will launch automatically.

That's all there is to setting up an SSH-secured link for RDP.

SSH and Windows: A Secure Partnership

With SSH, Windows servers get the benefit of secure communication, robust authentication, better access control, and easier firewall configuration. Administration of Windows servers can be done securely, with minimal impact on the operating system. File transfers between Windows servers or from Windows servers to other types of servers in a heterogeneous environment can be protected with a proven security protocol. Best of all, SSH eliminates the vulnerabilities exposed by tools like Telnet and FTP.

SSH Products from Attachmate

The Attachmate Reflection for Secure IT product line consists of SSH clients and servers for a variety of platforms. In addition to running on Windows servers and Windows desktops, Reflection for Secure IT supports Sun Solaris, IBM AIX, HP-UX, Red Hat Enterprise Linux, and Novell SUSE Linux—with clients and servers for all of these systems.

Reflection for Secure IT has received FIPS 140-2 validation from an independent third party for the quality of its cryptography.

About Attachmate

Attachmate delivers advanced software for terminal emulation, application integration, and secure communications. Our NetIQ business provides solutions for automating IT processes and managing performance, security, and compliance of distributed IT. With our technologies, more than 65,000 businesses worldwide are putting their IT assets to work in new and meaningful ways. www.attachmate.com



Corporate Headquarters
1500 Dexter Avenue North
Seattle, Washington 98109
TEL 206 217 7500
800 872 2829
FAX 206 217 7515

EMEA Headquarters
The Netherlands
TEL +31 172 50 55 55
FAX +31 172 50 55 51

Asia Pacific Headquarters
Australia
TEL +61 3 9825 2300
FAX +61 3 9825 2399

Latin America Headquarters
Mexico
TEL +52 55 9178 4970
FAX +52 55 5540 4886

WEB attachmate.com
E-MAIL info@attachmate.com

For regional office information, visit www.attachmate.com.