Reflection for Secure IT Client and Server for UNIX Evaluation Guide

November 2020

Serious about Security?

You're ready to get serious about security, and Reflection for Secure IT can help. By replacing non-secure Telnet and FTP with a reliable encrypted alternative, administrators can access any TCP/IP-based application through a secure transmission tunnel, and safely transmit sensitive data and manage remote servers--even over untrusted networks.

Why use Secure Shell?

The Secure Shell (SSH) protocol is a flexible, dependable way to guarantee the safety of your data in motion. The following vital features provide reliable safeguards that are increasingly important in today's world:

- Server authentication ensures that your clients communicate with the correct server.
- Client authentication ensures that only authorized client users can connect to your server.
- Data encryption assures that data in transit is indecipherable -- the client and server establish a unique key for each Secure Shell session, and this key is required to decipher the data.
- Data integrity checking verifies that your data has not been altered during transit.
- Port forwarding protects TCP/IP communications sent over an untrusted network.

Why use Reflection for Secure IT?

To build a security solution you can trust, you need to work with software and people you can trust. With Reflection for Secure IT and Micro Focus you can count on:

- Rock solid, supported software.

  Our developers use the latest techniques in secure software design to ensure that Reflection for Secure IT products are optimized for stability and security.

- Cross-platform support.

  Reflection for Secure IT clients and servers are available for Windows and UNIX operating systems. and run on both 32-bit and 64-bit hardware.

- Responsive technical support.

  Our technical support experts work closely with you and with our development team to make sure that your questions and concerns are answered quickly and correctly.

- Security updates.

  Our security specialists watch for potential security vulnerabilities. In the event that we learn of a new vulnerability, we keep you informed and make it our top priority to resolve your security concerns.

- Comprehensive documentation.

  Our documentation team is committed to providing you with complete, technically accurate information about all facets of our products.

See for yourself!

The evaluation software is fully-functional, time-limited copy of Reflection for Secure IT. The evaluation package installs both the Reflection for Secure IT server and the client.

If you haven't yet downloaded the evaluation software, go to https://www.microfocus.com/en-us/products/reflection-secure-it/overview, select the product you want to evaluate, and the fill out the evaluation request form. Shortly after you submit the form, you'll receive an email message with a link to the evaluation download page.

1. From the download library page, select the package link that's appropriate for your UNIX platform.
2. Locate the User Guide, which is available in both HTML and PDF format on the support site: https://support.microfocus.com/manuals/rsit_unix.html
3. If you're installing on a system that is already running a Secure Shell client or server, you must uninstall the earlier, existing version, before you install Reflection for Secure IT. (For details, see the "Installation" chapter in the user guide.)
4. Download the package from the File Information and Download page, and then follow the procedures in the user guide to install Reflection for Secure IT on your system.

Getting started with Reflection for Secure IT

This guide walks you through a few key tasks that are familiar to typical Reflection or Secure IT users. After you try the procedures, you can review the ideas under "Do more..." There you'll find suggestions that can help you use the rich feature set in Reflection to maximize the highest levels of security, while saving time and money.

After you've installed Reflection for Secure IT, you can use the **ssh** command with **-V** to confirm a successful installation.

NOTE: A script is installed that you can use to start, stop, and restart the server. The name and location of the script varies, depending on your operating system. For details, see the user guide.

Displaying the host key fingerprint

The Reflection for Secure IT server uses public key cryptography to authenticate the server host. A host key pair is created when you install the server. (Or, if you uninstalled an existing Secure Shell server, the server uses your existing host key. If your key is already in `/etc/ssh2`, the server uses that key. If an OpenSSH host key is found in `/etc/ssh`, that key is migrated to the correct format and location, and the server uses the migrated key).

You can confirm the identity of your host key by displaying its unique fingerprint. This information is useful to have when client computers connect to your server for the first time. To display your host key fingerprint, use the **ssh-keygen** command as shown here:

```
$ ssh-keygen -F /etc/ssh2/hostkey.pub
Fingerprint for key:
xokin-lunob-megec-biguc-pizyh-tahuv-cebup-ricum-maves-nydam-zyxax
```

You'll see this same fingerprint again when you make your first test connection in the next exercise.

Getting Connected

For a first test, let's make a connection from the Secure Shell client to the server running on the same computer.

Connect to localhost

By using *localhost* for the server name, the following command connects your Secure Shell client to the server running on the same computer. Because no user name is specified, the client connection uses your current login name.

1. To initiate the test connection, enter the following command:

   ```
   $ ssh localhost
   ```

   The client doesn't yet have a copy of your host key, so you'll see an unknown host key prompt like this one:

   ```
   Host key not found in hostkeys database.
   Key fingerprint:
   xokin-lunob-megec-biguc-pizyh-tahuv-cebup-ricum-maves-nydam-zyxax
   You can get a public key's fingerprint by running
   $ ssh-keygen -F publickey.pub
   on the keyfile.
   Are you sure you want to continue connecting (yes/no)
   [Enter=no]?
   ```

   Notice that the fingerprint in the prompt matches the host key fingerprint you saw earlier, confirming that the server you're connecting to is, in fact, the server running on your computer, and not an imposter.

2. Enter *yes* to the unknown host prompt.
3. To complete the connection, enter your password.
4. To close the test connection, type *exit*.

What happened when you accepted the unknown key?

When you respondend “yes” to the unknown host prompt, the client added your host key to your known hosts list. To locate the key, look for the following file in your home directory:

`.ssh2/hostkeys/key_22_localhost.pub`

The key name identifies the host and port. (In most cases, it will also include the host’s IP address, which is missing in this localhost example.) As long as this file remains in place, you won’t see another host key prompt when you connect to this server from your current user account on this computer.

Connecting securely to a remote server

Now that you’ve tested a local connection, you are ready to test a connection from a client running on one computer to a server running on a remote host. (You can install the Reflection for Secure IT evaluation software on both computers, or connect to and from computers running other SSH programs.)

1. Use the following syntax to make your connection: `ssh user@hostname`. For example, `$ ssh Lee@abc.com`
2. Confirm the host key and authenticate to the server. You can now use the remote terminal session to execute commands securely on the server.

Do more...

The User Guide is available in both HTML and PDF formats on the support web site. Information on each of these following features are available in detail there or from the product’s local documentation.

- **Find additional information about ssh command line options**

  These examples demonstrate ssh without using any of the many options available for modifying the connection. To see a short summary of command line options, use `-h` as shown here: `$ ssh -h`

- **Modify the default client and server settings**

  You can modify the default client settings by editing the configuration file. The global client configuration file is `/etc/ssh2/ssh2_config`, which is installed when you install the product.

  You can modify the default server settings by editing the configuration file. The global server configuration file is `/etc/ssh2/sshd2_config`, which is installed when you install the product.

- **Configure public key authentication**

  With public key user authentication, the Secure Shell server uses a unique digital signature to authenticate the client user. To configure this, you create a key pair on your workstation using the ssh-keygen utility, upload the public key to `$HOME/.ssh2` on the server, and edit your`$HOME/.ssh2/authorization` file.

- **Use OpenSSH format public keys**

  The Reflection for Secure IT server can use public keys created by OpenSSH clients. To use an OpenSSH public key for authentication, simply copy the public key to `$HOME/.ssh2` on the server, and edit your `$HOME/.ssh2/authorization` file. You don’t need to modify the key format or make any changes to the client.

- **Install known host keys on client computers**

  To simplify initial connections, and eliminate the risk created by allowing users to accept unknown host keys, you can install host keys on client computers

- **Reuse an existing connection**

You can use the client **ConnectionReuse** keyword in the `ssh2_config` file to configure reuse of existing connections. Enabling this feature allows you to start new ssh, scp, and sftp sessions without having to reauthenticate. When ConnectionReuse is set to "yes," a new session reuses an existing tunnel if the target host, port, and user are all identical to those used for the established connection.

- **Use the man pages to get detailed command and configuration help**

  UNIX manual pages are installed for all supported commands and configuration files. To view these, type *man* followed by the command for filename. For example, `$man ssh2_config`.

Secure File Transfer

Reflection for Secure IT supports two file transfer commands: sftp and scp. Both commands can help you manage file transfers efficiently and securely.

Transferring files using scp

Each scp command established a Secure Shell connection and transfers one or more files. For each transfer, specify the source followed by the destination, as shown below. Both source and destination file specifications can include host and user information to indicate that files are to be copied to or from a remote host. All transferred data is securely encrypted.

Here's the basic syntax:

```
scp [[user@]host:]source [[user@]host:]destination
```

Here are some examples to get you started.

- The following example copies a local file (fxl.htm) to the specified remote directory (demo/path/). The client user (Lee) needs to authenticate to the server host (abc.com) before the transfer occurs:

  ```
  $scp fxl.htm Lee@abc.com:demo/path/
  ```

- The next example copies multiple remote files from the default remote directory to the local computer. (In this case, the destination is designated by the single period representing the current local directory.)

  ```
  $ scp Lee@abc.com:*.htm .
  ```

- Finally, here's an example that transfers a file between two remote hosts. Before this transfer can occur, two client authentications are required - one for each host.

  ```
  $ scp Lee@abc.com:source/file1
  Lee@xyz.com:destination/
  ```

If your company's security policies allow the use of passphraseless public keys, you can configure public key user authentication with keys that don't require user input for authentication. With this setup, you won't need to enter a password for each scp connection. You might take this approach if you want to create batch files to automate scp transfers.

Tranferring files using sftp

You can use an interactive sftp session to transfer files securely between the client and a remote server, and also to execute file management commands securely on the remote server.

1. To open an interactive sftp session, first connect to a remote host as shown in this example:

```
$ sftp Lee@abc.com
```

After a successful connection is established, the following prompt appears: `sftp>`

2. For a quick list of available commands, use the following: `sftp>help`
3. To get additional help on any individual command, type help followed by the command name. For example, to see details about the get command, use the following: `sftp>help get`.
4. Use any of the available sftp commands to transfer and manage files.
5. Use exit to end the sftp session: `sftp>exit`.

In the following sample session, Lee uses sftp to connect to the remote server abc.com. After authenticating successfully, she change the remote directory (cd), lists the remote files (ls), transfers a group of files from the remote server to the current local director (get), lists the local files to confirm the transfer (lls), and then ends the session (exit).

```
$ sftp Lee@abc.com
Lee's password:
Authentication successful.
sftp> cd demo
sftp> ls

.:
f1.txt  f2.txt f3.txt  fx1.htm  fx2.htm fx3.htm
sftp> get *.htm
/demo/fx1.htm                     4  0.0KB/s
00:00  100%
/demo/fx2.htm                     4  0.0KB/s
00:00  100%
/demo/fx3.htm                     4  0.0KB/s
00:00  100%
sftp> lls *.htm
fx1.htm fx2.htm fx3.htm
sftp> exit
$
```

Do more...

- **Find additional information about scp and sftp command line options**

  To see a short summary of command line options, use -h as shown here: `$ scp -h` or `$ sftp -h`. There is more information in the appendix of the user guide.

- **Control overwrite behavior in scrp transfers**

  By default, existing files are overwritten when you transfer files using scp. To control overwrite behavior, use the --overwrite flag on the command line. Options include yes, no, and ask. For example, to query the user before overwriting a file: `$ scp --overwrite ask fxl.htm Lee@abc.com`.

- **Create sftp batch files**

  To create and use sftp batch files, you can configure the client and server to support a non-interactive client authentication method, such as GSSAPI, or public keys without passphrase protection. Then, on the client, create a batch file that contains the sftp commands you want to automate. Finally, use the sftp with -B to connect to the remote host and run the batch file, as shown here: `sftp -B batch_file user@host.com`.

- **Limit access to server files**

  Two server configuration keywords are available to restrict users to their home directory for file transfers: ChrootSftpGroups and ChrootSftpUsers. These settings affect both sftp and scp connections from all Reflection for Secure IT clients. For more information, refer to the Server Configuration Keywords in the Appendix of the user guide.

Secure Communications using Port Forwarding

Port forwarding, also known as tunneling, is a powerful feature you can use to redirect communications through the Secure Shell channel of an active session. When configured, this capability allows you to route all your traffic through one secured port in the firewall (similar to the way SOCKS works, except with encryption).

You can use port forwarding to secure the data exchanged between any client and server applications that use the TCP/IP protocol.

Configuring a local port to forward HTTP communications

The following example demonstrates forwarding of HTTP communications between a Web server and a Web browser. You can use the same basic approach to secure communications sent to and from any TCP/IP client and server applications.

1. From a computer running the Secure Shell client, create a forwarded port by entering the following command (replace user@host.com with your user name and Secure Shell server host): `$ ssh -L 8080:support.microfocus.com:80 user@host.com`.
2. Enter your password to authenticate to the Secure Shell server.
3. Open a web browser and enter the following URL: `http:\\localhost:8080`

   Your browser should now display the Micro Focus support web site.

What's going on?

The ssh -L option can be used to configure the client to forward data from a specified local port through the Secure Shell tunnel to the Secure Shell server. From there, the data can be redirected to a port on the same server, or to a port on a different host.

Here's what happens when you execute the command shown in the previous example:

1. After you authenticate to the Secure Shell server (host.com in the example), a secure tunnel is established between your client computer and host.com. Because local port forwarding is configured, the client also starts listening on port 8080.
2. When you point your browser to the forwarded port on the client computer (localhost:8080 in the example), the data sent to this port is forwarded through the secure tunnel to the Secure Shell port on host.com.
3. The server forwards this data (in the clear) to the specified destination (port 80 on support.microfocus.com in this example).
4. Data from the web server is returned to the client using the same pathway in reverse.

NOTE: In this somewhat artificial example, the HTTP data is encrpted between your Secure Shell client and the Secure Shell server. It then passes in the clear over the Internet between the Secure Shell server and the web server. In a real-world example, both servers would typically be on the same side of your firewall. For more information on port forwarding, see Port Forwarding in the user guide.

Do more...

- **Forward data exchanged between any TCP/IP client and server**

  You can secure the data exchanged between any client and server applications that use the TCP/IP protocol. This means that you can securely forward Telnet, HTTP, SMTP, POP, and IMAP communications over an untrusted network. Once you've configured a forwarded port, set your TCP/IP application to connect to the forwarded port.

- **Configure forwarding using the command line or the configuration file**

  Reflection for Secure IT offers flexible options for configuring forwarding. In addition to configuring port forwarding from the command line, you can configure forwarding using the LocalForward and RemoteForward keywords in the client configuration file.

  By using these keywords within host stanzas, you can configure different port forwarding behavior for connections to different hosts

Centralized Public Key Infrastructure (PKI) Support

Reflection PKI Services Manager uses PKI to validate the authenticity of certificates presented by communicating parties. Using PKI Manager you can centrally administer PKI functions, such as specifying trust anchor certificates, certificate stores, certificate revocation checking, certificate-to-user- ID mapping, and audit logging for multiple servers.

PKI Manager is included as a component of Reflection for Secure IT, at no additional cost. It is a separate download and installation. For detailed information about installing and configuring PKI support, see the Reflection PKI Services Manager user guide on the documentation page: https://support.microfocus.com/manuals/reflection.html?prod=PKID.

Legal Notice