The Secure Shell Handbook

Contents

About the Author.....3

Introducing SSH - the Secure Shell.....3

Secure Shell (SSH) at a glance.....3

Using PuTTY on Windows.....6

Download and unzip the PuTTY binaries.....6
Connect to your instance using PuTTY.....6
Generate SSH Keys with PuTTYgen.....7
Convert SSH Keys to and from OpenSSH formats with PuTTYgen.....10
Upload your public key to an existing server.....11
Connect to your server via SSH using PuTTY and SSH Keys.....12
Set up SSH key management with PuTTY Agent.....12
Set up TCP tunnels over SSH using PuTTY.....12
Create a SOCKS proxy with PuTTY.....13
Transfer files to and from your instance with pscp.....13
X11 Forwarding with PuTTY and XMing.....14
Verify MD5 and SHA-1 checksum on Windows.....14
Install GnuPG on Windows.....15
Verify the PuTTY signature using GPG and RSA keys.....15
Install XMing on Windows.....16

SSH Tips and Tricks.....16

Encrypt, Decrypt and change your Private Key password.....16

Index.....17

About the Author

SSH Handbook by Mihai Criveti

Introducing SSH - the Secure Shell

Secure Shell (SSH) at a glance

Secure Shell (SSH) is an encrypted network protocol for initiating secure remote shell sessions. Users connect to a SSH server with an SSH client application and can execute remote shell commands on the remote machine over a secure channel. The PuTTY SSH Client and OpenSSH (providing both client and server components) are common packages used to administer remote systems using the SSH protocol. You can use PuTTY or OpenSSH to interact with your Linux instances. Using SSH keys allows you to secure your instances against common brute force attacks.

Using the PuTTY SSH Client on Windows

Download and Install PuTTY on Windows

Follow the instructions on the PuTTY homepage at http://www.chiark.greenend.org.uk/~sgtatham/putty/[1] to download and install PuTTY on your system.

1. Download the **ZIP** file containing all binaries and help files to get started, or download each tool individually.
2. Unzip the PuTTY archive to a local directory.
3. Verify the cryptographic hash (ex: MD5, SHA-1) and the PuTTY signature (GPG / RSA keys) as instructed in the PuTTY documentation.

Log into your instance using your username and password

You will need to know the IP address (or hostname), the username and password of the server you want to connect to.

1. Start `PuTTY.exe`.
2. In the **Session** category, enter the `Host Name (or IP address)` of your server in the format `user@host`, provide a name for your session in **Saved Sessions** then click **Save** to save your session for future re-use.

   Note: Example: `root@10.0.0.1`, where 10.0.0.1 is your server address, and root is the username.
3. Click **Open** to connect to your server.

Create an SSH key

Generate a RSA private and public key pair using `PuTTYgen`:

1. Start `PuttyGen.exe` and click **Generate**[2].
2. Click **Save private key** to save your PuTTY format (PPK) private key.
3. Copy the text from the **Public key for pasting into OpenSSH authorized_keys file** field.

   Note: You will use this public key to authorize your user for SSH access.

Allow log in using public keys for your user

Place your public key in the `~/.ssh/authorized_hosts` file on each server you want to access using SSH keys.

---

[1] 3rd party website subject to change

[2] current versions of `PuTTYGen` will create a 2048 bit RSA key by default.

1. Create the `.ssh` directory for your user, if it does not previously exist:

```
[[ -d ~/.ssh ]] || mkdir ~/.ssh
```

2. Place the **Public key for pasting into OpenSSH authorized_keys file** generated above into the `~/.ssh/authorized_keys` file. Replace `PUBLIC_KEY` in the command below with your actual public key in OpenSSH format, as generated by `PuTTYGen`:

```
# Append a long line containing the private key to authorized_hosts:
echo 'PUBLIC_KEY' >> ~/.ssh/authorized_hosts
```

3. Set restrictive permissions on the `.ssh` directory and files[3]:

```
chown -R $USER: ~/.ssh
chmod 700 ~/.ssh
chmod 644 ~/.ssh/authorized_keys
```

Log into your instance using SSH keys

Load an existing session (or provide your server `username` and **Host Name** in the form: `user@hostname`) and perform the following steps to use SSH keys when connecting with `PuTTY`:

1. Under Category, navigate to **Connection** > **SSH** > **Auth**.
2. In the **Authentication parameters** section, click **Browse** to select a **Private key file for authentication** and locate your private key in PuTTY PPK format.
3. You can now navigate to **Session** if you want to **Save** your session, then click **Open** to connect using an SSH key.

Copy files using SCP

You can copy files to and from your server using PuTTY `pscp.exe`.

To receive files from a remote server: `pscp [options] [user@]host:source target`.

For example, to copy `/etc/hosts` from your server to `c:\temp\hosts.txt`:

```
pscp user@host:/etc/hosts c:\temp\hosts.txt
```

To send files to a remote server: `pscp [options] source [source...] [user@]host:target`.

For example, to copy `c:\temp\hosts.txt` to your server `/tmp/hosts`:

```
pscp c:\temp\hosts.txt user@host:/tmp/hosts
```

Using OpenSSH SSH Client on Linux and Mac

If you're running Linux, Mac or another UNIX-like operating system, chances are your system already comes with an SSH client. Type `ssh` in a terminal. If the command is not recognized, you may need to install OpenSSH client packages.

Download and Install OpenSSH on Linux

Red Hat / CentOS / Fedora (yum based distributions):

```
# Search for the OpenSSH client package
yum search openssh

# Install the corresponding package on your distribution
yum install openssh-client
```

[3] At this point, only the `authorized_keys` file may exist on your server.

Debian / Ubuntu (apt based distributions):

```
# Search for the OpenSSH client package
apt-get update && apt-cache search openssh

# Install the corresponding package on your distribution
apt-get install openssh-client
```

Other distributions / operating systems: check the instructions listed on http://www.openssh.com and / or your distribution specific documentation.

Log into your instance using a username and password

```
ssh user@host
```

Create a SSH key

Create a `4096` bit `RSA` key in `~/.ssh/id_rsa` and provide a label `you@example.com`:

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -C "you@example.com"
```

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cmihai/.ssh/id_rsa.
Your public key has been saved in /home/cmihai/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:d8Mh5ffUFg1eAho7C4eg0IOFKih7AWDpVCIfPkS6Hl0 you@example.com
The key's randomart image is:
+---[RSA 4096]----+
|++B*. .    . o.ooo|
|+B+oo. . . * . o+|
|=o+ .E  o * o o +|
|=oo..    o = o + |
|+o o    S o +   .|
|o o      . . .   |
| o               |
|                 |
|                 |
+----[SHA256]-----+
```

Allow log in using public keys for your user

```
ssh-copy-id user@host
```

Log into your instance using SSH keys

```
ssh -i ~/.ssh/id_rsa user@host
```

Copy files using SCP

```
# Copy /remote/file on a remote host to /local/file:
scp -i ~/.ssh/id_rsa user@host:/remote/file /local/file

# Copy a directory recursively from /remote/dir to /local/dir
scp -i ~/.ssh/id_rsa -r user@host:/remote/dir /local/dir

# "Upload" a local directory to a remote host:
scp -i ~/.ssh/id_rsa -r /local/dir user@host:/remote/dir
```

Using PuTTY on Windows

PuTTY is a free implementation of SSH for Windows and UNIX and a xterm emulator written and maintianed primarily by Simon Tatham.

The following tools from the PuTTY family are described in this section:

- PuTTY (the Telnet and SSH client itself).
- PSCP (an SCP client, i.e. command-line secure file copy).
- PSFTP (an SFTP client, i.e. general file transfer sessions much like FTP).
- PuTTYtel (a Telnet-only client).
- Plink (a command-line interface to the PuTTY back ends).
- Pageant (an SSH authentication agent for PuTTY, PSCP, PSFTP, and Plink).
- PuTTYgen (an RSA and DSA key generation utility).

Additional supporting tools are also described:

- XMing - a X Server for Windows (used to show running GUI applications over SSH).
- GNU Privacy Guard (GnuPG) - free implementation of the OpenPGP standard (used to verify PuTTY download keys and signatures).

Download and unzip the PuTTY binaries

1. Open your Web Browser and navigate to the PuTTY homepage at http://www.chiark.greenend.org.uk/~sgtatham/putty/

   You will be presented with the PuTTY home page (at the time of this writing).

   Warning: The link provided above is for a 3rd party website that hosts the PuTTY binaries. The link or availability of PuTTY at a later date may differer.

2. Click on the **Download** link and download the latest version of PuTTY:

   You can also download a **ZIP** file containing all binaries and help files.

3. Unzip or place the downloaded files in a local directory.
4. Verify MD5 and SHA-1 checksum on Windows on page 14.
5. Verify the PuTTY signature using GPG and RSA keys on page 15.

   The PuTTY website provides RSA keys compatible with both GnuPG and PGP2. This document provides instructions for installing GnuPG from the Gpg4win distribution on Windows, here: Install GnuPG on Windows on page 15.

You now have PuTTY on your machine.

You can now Generate SSH Keys with PuTTYgen on page 7.

Related information

PuTTY Manual - Appendix E: PuTTY download keys and signatures

IETF Request for Comments: 3174 - US Secure Hash Algorithm 1 (SHA1)

Connect to your instance using PuTTY

Connect to your instance using SSH with a Username and Password.

Obtain the Username and Password associated with your server (or log in from the local console).

1. Start `PuTTY.exe` or your platform equivalent to begin connecting to your server.
2. In the Session category, set the basic options for your PuTTY session:
   a) Enter the Host Name (or IP address) of your server.
3. Navigate to **Connection** > **Data** and provide a **Auto-login username**. You can also provide the username by prefixing the server address with `username@`.
   `root`
   This name will be automatically sent to the server.

   Important: You should log in with a regular user account once you complete the steps Create a regular user account.
4. Navigate to the **Session** category and save your session by providing a name in the **Saved Sessions** field and clicking on **Save**.
   You can restore your connection options at a later date by selecting the desired session from the list and clicking **Load**.
5. Click **Open** to initiate your PuTTY session.

You are now connected to your instance.

Generate SSH Keys with PuTTYgen on page 7.

Generate SSH Keys with PuTTYgen

Generate PuTTY-format key (*.PPK) files used to permit SSH key based Secure Shell access with tools such as `pagent`, `pscp`, `psftp` and `putty`.

You will need to install PuTTYgen and PuTTY as described in Download and unzip the PuTTY binaries on page 6.

This task normally applies to Windows systems using PuTTY, through the same steps apply also if you are using PuTTY on Linux or another platform[4].

Attention: You only need to perform this task once, as the private key can be used to access multiple systems that are configured to permit access to you via your public key.

Note: By default, PuTTY uses keys in the PPK format, but PuTTYgen can be used to convert keys to the more common OpenSSH format, as described in Convert SSH Keys to and from OpenSSH formats with PuTTYgen on page 10.

[4] While PuTTY is also available on Linux and other platforms, the OpenSSH client `ssh` is more commonly used.

1. Start `PuttyGen.exe` or your platform equivalent to begin generating your SSH key pair.

Key generation options:

2. In the **Parameters** section, select your key generation options:
   a) Select the **type of key you wish to generate**.
   b) Select the **number of bits in a generated key**[5]

   The SSH-1 protocol only supports RSA. SSH-2[6] supports both RSA and DSA keys. Using SSH-2 RSA is strongly recommended for security reasons.

   Select SSH-2 RSA and `4096` bits.

Generate your key:

3. Click **Generate** to create a public/private key pair.
4. Move your mouse randomly over the progress bar to generate entropy.

   Moving your mouse around in the blank area is used to generate additional randomness to ensure that strong keys are generated.

Configure your generated key:

5. Provide a **Key comment** to be associated with your key. A common convention is to use your name and the systems it will be used on.
   name@site

[5] For RSA, minimum bit size is 768 and default is 2048.
[6] SSH-2 is a revised version of the SSH protocol and features security and feature improvements over SSH-1.

6. Optional: Provide and confirm the **Key passphrase** that will be used to unlock your key every time it used, or when you add it to an SSH agent[7].

   It is highly recommended that you set a passphrase on your SSH key. `PuTTY Agent` or `ssh-agent` can be used to unlock your key and still permit password-less SSH for automation purposes as long as the agent is running.

   Provide a strong alphanumeric passphrase of considerable length, composed of lowercase, uppercase and special characters.

Save your generated key:

7. Click **Save private key** and provide the file path to the location you wish to save your key in.

   This is your **private** key. Do not share it with anyone else. If another person (or service) requires access to the server they should use their own private / public key pair.

   Restriction: Permissions on your key should be set as such they don't permit other users from copying your key. For example, on Linux you would set permissions to owner read only, like this: `chmod 400 my_key`.

   Your PuTTY-format key (*.PPK) has been saved to the specified location.

8. Click **Save public key** and save your public key.

   This will save the public key in RFC 4716 standard SSH-2 format, as used by SSH.com. This is not generally required for OpenSSH (as found on most Linux systems. Instead you should use the public key format described in the next step.

9. Copy the text from the **Public key for pasting into OpenSSH authorized_keys file** field.

   This key is normally placed in the user directory `~/.sshd/authorized_hosts` for each of the servers you wish to access using SSH. As the name suggests this key is public and can be used to provide access to resources (such as servers) to anyone who has your private key (ideally, only you).

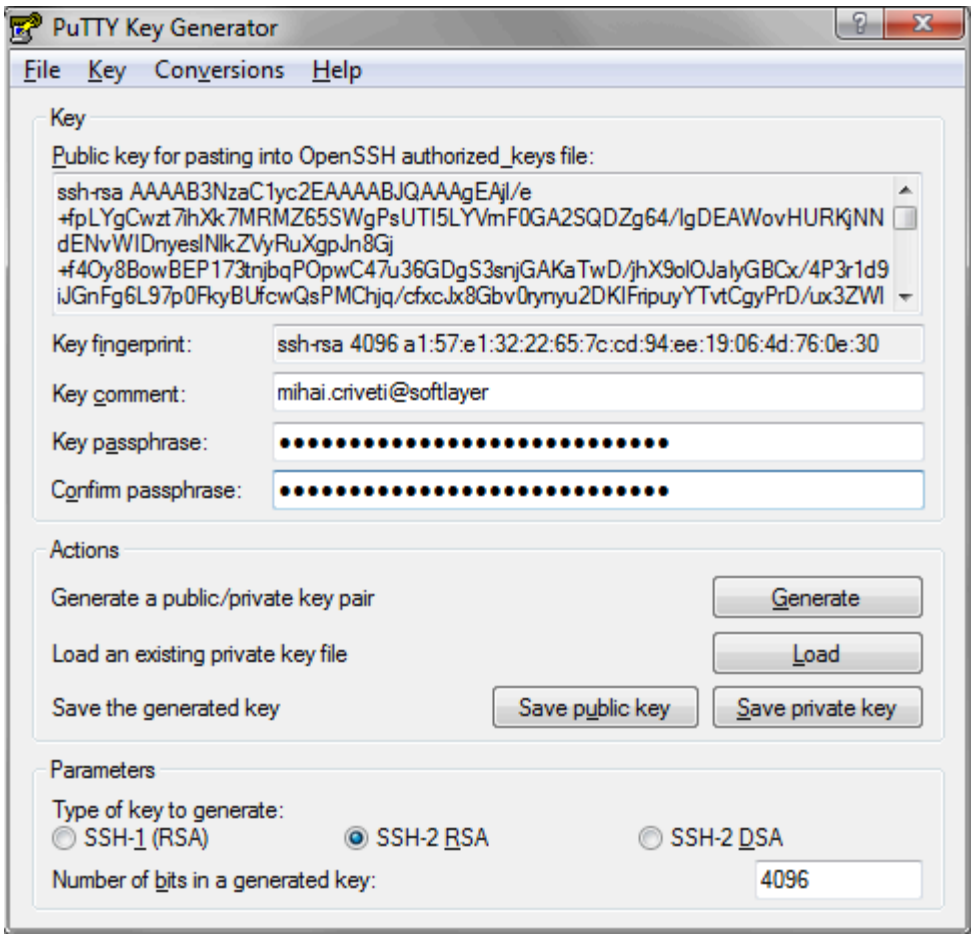
   Tip: You can re-generate your public key from your private key, if needed. You can do so by loading an existing private key in PuTTYgen by clicking the Load button. If the key is in a different format than PuTTY PPK, you will use the **Conversions** > **Import** menu instead.

10. Exit the PuTTY Key Generator by clicking on **File** > **Exit**.

You have generated a private and public key pair for use with PuTTY tools. Notice also that a **Key fingerprint**[8] was generated for your public key. This can be used to quickly identify your key, without having to compare the entire public key.

[7] An SSH agent runs for the duration of your local operating system session, and stores one or more (unlocked) keys in memory, enabling SSH access without having to unlock your key each time it is used. For example, keys loaded in `Pageant` can be used automatically by `PuTTY`, `PSCP`, `PSFTP`, `Plink`, `WinSCP`, `FileZilla` and other tools that support the protocol, without the need for you to type a key passphrase each time.

[8] Public Key Fingerprint - short cryptographic hash (like SHA-256) of your public key used to identify it visually.



You can now Upload your public key to an existing server on page 11 to allow SSH key access by editing your system user's `authorized_hosts` file, Connect to your server via SSH using PuTTY and SSH Keys on page 12, Transfer files to and from your instance with pscp on page 13, Copy files with WinSCP, Convert SSH Keys to and from OpenSSH formats with PuTTYgen on page 10 or Set up SSH key management with PuTTY Agent on page 12.

Related information

PuTTY Documentation - Chapter 8: Using public keys for SSH authentication

NIST: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths

Convert SSH Keys to and from OpenSSH formats with PuTTYgen

Convert your private key to and from other formats for use with PuTTY or OpenSSH.

You will need to install PuTTYgen - as described in Download and unzip the PuTTY binaries on page 6.

This task normally applies to Windows systems using PuTTY - through the same steps apply if you are using PuTTY on Linux or another platform.

1. Start `PuttyGen.exe` or your platform equivalent to begin generating your SSH key pair.
2. Click on **Conversions** > **Import** and select your Private key in OpenSSH format.
   `~/.ssh/id_rsa`
   If your key is protected by a passphrase, you will need to enter the passphrase to unlock your key.
3. Click **Save private key** and save your private key in a safe location.

You have converted your private key to the PPK format used by PuTTY tools.

You can now Upload your public key to an existing server on page 11 to allow SSH key access by editing your system user's `authorized_hosts` file, Connect to your server via SSH using PuTTY and SSH Keys on page 12, Transfer files to and from your instance with pscp on page 13, Copy files with WinSCP, or Set up SSH key management with PuTTY Agent on page 12.

Upload your public key to an existing server

You need a private key, a public key in OpenSSH format, an existing Linux server instance that you can access with a username and password.

Note: This is the password usually set during OS installation.

Important: As a best practice, you should first create a non-privileged username and password, as described in Create a regular user account, and Configure sudo for your user.

1. Log into your instance as described in Connect to your instance using PuTTY on page 6.
2. Add your OpenSSH format public key in the `~/.ssh/authorized_key` file.

   Note: We are adding a SSH-2 public key, thus we are using the `authorized_keys2` file.

   ```
   [[ -d ~/.ssh ]] || mkdir -p ~/.ssh # Create the ~/.ssh directory if it
    does not exist
   $EDITOR ~/.ssh/authorized_keys # Use the configured editor to create or
    edit the authorized_hosts file
   ```

   Place your public key in this file. This looks similar to:

   ```
   ssh-rsa AAAAB3NzaC1yc2EAAAAD[more text here] cmihai@ADMINIB-QTA7GRE
   ```
3. Set restrictive permissions on the `authorized_keys` file and the ~/.ssh directory.

   ```
   chown -R $USER: ~/.ssh
   chmod 700 ~/.ssh
   chmod 400 ~/.ssh/id_rsa
   chmod 644 ~/.ssh/id_rsa.pub
   chmod 644 ~/.ssh/authorized_keys ~/.ssh/known_hosts
   ```

   Note: Permissions should normally be:

   - SSH Directory: ~/.ssh -> 700 (drwx------)
   - Public keys: ~/.ssh/id_rsa.pub -> 644 (-rw-r--r--)
   - Private keys: ~/.ssh/id_rsa -> 400 (-r--------)
   - known_hosts: 644
   - Owner and group: your user / your group
   - Permissions on your home folder should not allow rw by group / other

   If you see a screen like this, then the permissions are too open:

   ```
   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
   @         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
   Permissions 0644 for '/home/cmihai/.ssh/id_rsa' are too open.
   It is recommended that your private key files are NOT accessible by others.
   This private key will be ignored.
   ```

You have uploaded your public key associated with your private key to your instance.

You can now Connect to your server via SSH using PuTTY and SSH Keys on page 12.

Related information

SuperUser - Permissions on private key in .ssh folder

Connect to your server via SSH using PuTTY and SSH Keys

You will need a SSH private key, as described in Generate SSH Keys with PuTTYgen on page 7 and a public key configured on the server you want to connect to, as described in Upload your public key to an existing server on page 11.

1. Start `PuTTY.exe` or your platform equivalent to begin connecting to your server.
2. In the Session category, set the basic options for your PuTTY session:
   a) Enter the Host Name (or IP address) of your server.
3. Navigate to **Connection** > **Data** and provide a **Auto-login username**. You can also provide the username by prefixing the server address with `username@`.
   `root`
   This name will be automatically sent to the server.

   Important: You should log in with a regular user account once you complete the steps Create a regular user account.
4. Navigate to the **Session** category and save your session by providing a name in the **Saved Sessions** field and clicking on **Save**.
   You can restore your connection options at a later date by selecting the desired session from the list and clicking **Load**.
5. Navigate to the **Connection** > **SSH** > **Auth** category and click on **Browse** under **Private key for authentication**, then locate your private SSH key.
   This will associate a private key with your PuTTY session to be sent automatically during login.
6. Click **Open** to initiate your PuTTY session.

You are now connected to your server.

Set up SSH key management with PuTTY Agent

1. Start `PAGEANT.EXE` or your platform equivalent to start using a SSH agent.
   PuTTY Agent will start minimized to the system tray.
2. Add one or more SSH keys to PuTTY Agent:
   a) Right click the PuTTY Agent icon in the system tray and click Add Key.
   b) If your key is passphrase protected, you will be prompted to enter the passphrase associated with your key. The key comment associated with this key will display above the passphrase prompt to aid in identifying the key.

SSH Agent will now keep a unlocked key in memory during the duration of your local system session, and present it automatically to supported tools like PuTTY whenever you connect to a SSH enabled server.

Connect to your server via SSH using PuTTY and SSH Keys on page 12. You do not need to specify a key in the PuTTY configuration, as this will be taken automatically from SSH Agent.

Related information

PuTTY Documentation - Chapter 9: Using Pageant for authentication

Set up TCP tunnels over SSH using PuTTY

SSH can be used to perform TCP port forwarding. One use case for this would be securely connecting to a VNC or RDP server over an encrypted SSH tunnel.

We assume that you have setup SSH keys as described in Generate SSH Keys with PuTTYgen on page 7.

You can tunnel other types of network traffic through an encrypted SSH tunnel using TCP port forwarding over your SSH session. The following steps describe how to forward a single port (such as 3390), commonly used for RDP from your instance to your local machine.

1. Start PuTTY. In the **hostname** field, type the Public IP address of your Linux instance.
2. In Category box, click SSH, select the Compression checkbox.
3. Expand SSH and click Auth. Load your encrypted key (.PPK format) by clicking Browse and navigating to the location where you .PPK key is stored.
4. Click Tunnel. Then, select a Source port to be used on your local machine (for example, 3390 in case you want to use RDP. You cannot use 3389 as default port of RDP because it is used by your local machine in case your machine is running on Windows) and a Destination to be forwarded (for example, 10.10.10.23:3389).
5. Click Session from Category box. Add a name to your session, save your settings and click Open.

   Note: You may forward multiple ports on any one connection. You can have multiple static and dynamic port forwarding on the same session.
6. Connect to RDP through 127.0.0.2:3390 (the port which was configured in step 4).

For additional details and uses additional uses (such as forwarding a port from your machine or using dynamic port forwarding) see The Tunnels panel section of the PuTTY User Manual.

Create a SOCKS proxy with PuTTY

Perform the steps in Connect to your instance using PuTTY on page 6 and save your session.

1. In the PuTTY configuration - navigate to **Connection** > **SSH** > **Tunnels**.
2. **Add a new forwarded port**
   a) Provide a available Source port (on your machine), such as 8888.
   b) Select the **Dynamic** radio button.
   c) Click **Add**.
   d) Navigate to Session and save your PuTTY session.
3. Click **Open** to connect to your system.
4. Configure your Web Browser to use the port configured above as a SOCKS5 proxy.

Your Web Browser connections will now flow through your SSH host.

Transfer files to and from your instance with pscp

You can copy files to and from your server using PuTTY `pscp.exe`.

Receive files from a remote server:

To receive files from a remote server: `pscp [options] [user@]host:source target`.

For example, to copy `/etc/hosts` from your server to `c:\temp\hosts.txt`:

```
pscp fred@example.com:/etc/hosts c:\temp\hosts.txt
```

Send files to a remote server:

To send files to a remote server: `pscp [options] source [source...] [user@]host:target`.

For example, to copy `c:\temp\hosts.txt` to your server `/tmp/hosts`:

```
pscp c:\temp\hosts.txt fred@example.com:/tmp/hosts
```

You can now upload and download files.

X11 Forwarding with PuTTY and XMing

Install XMing:

1. In the PuTTY configuration - navigate to **Connection** > **SSH** > **X11**.
2. Check the **Enable X11 forwarding** checkbox.
3. Type in the X display location.
   This is the location displayed by XMing (or another X11 server), such as `localhost:0.0`

You can now tunnel X11 applications to your local X11 server using SSH.

Run a GUI application and test that it functions correctly. Please note that X11 forwarding does not preserve your graphical session after closing SSH. Look at VNC in GUI Access of SSH with VNC and RDP if you wish to preserve your X11 session.

Related information

PuTTY Manual - Using X11 forwarding in SSH

Verify MD5 and SHA-1 checksum on Windows

Using the Microsoft File Checksum Integrity Verifier to verify checksums on the Windows platform.

1. Navigate to http://www.microsoft.com/en-us/download/details.aspx?id=11533 and download the Microsoft File Checksum Integrity Verifier from Microsoft.
2. Execute the downloaded package and proceed to extract the tool. Example: `c:\tools\`
3. Compute both MD5 and SHA1 for all files in the `PuTTY` directory:

```
PS C:\Tools\PuTTY> C:\Tools\fciv.exe -both -wp C:\Tools\PuTTY
```

```
//
// File Checksum Integrity Verifier version 2.05.
//
                MD5                                 SHA-1
--------------------------------------------------------------------------
51bd1bf6d054a01c8e556ecfa705b537 bea6fedf753178170aa45708e902c2c48b9d5bc8
 PAGEANT.EXE
07d07cc89c7b25229b3b999724bd3e5b e79298341d580033c6011ee0eef51fd5c9693c6b
 PLINK.EXE
b9735750b270236fb5228f4a344b22ef ec11523186a2cbba7bc3f68cba1a5ae36065fd4a
 PSCP.EXE
d83d91061fbd0a270883cee9ac6129a9 ac37ac13032672e59da953cabf14ceff9f2082a4
 PSFTP.EXE
a51cb0942adc0c78ae3d1ee6e0ebfa08 653f7e757207a81d48ec9a46811b32d51a8a7d54
 PUTTY.CHM
608ba22d585805f56ed6682c8a42392e 10a8d18c78a41b45a27caec959283b98dac1147c
 PUTTY.CNT
354d9abefa0ed67a08bd056324284d6e 91b21fffe934d856c43e35a388c78fccce7471ea
 PUTTY.EXE
c2bcb4d56fd986b1ad94b595817ae996 99990e52bce8b05d73f31a88baa67a8e94f5f4ef
 PUTTY.HLP
```

```
476992da1f015286c4fc70e4a31f5bfc 7fe800fdd10136adabb2ed5df80fe6a62759755c
 putty.zip
be49685e80862a4a49c06bb2545a6f4e 54c8601236dfe90b2e3e62f74155709d0231dcb2
 PUTTYGEN.EXE
```

4. Compare these to the checksums listed on the PuTTY site:
   MD5:

```
a51cb0942adc0c78ae3d1ee6e0ebfa08  putty.chm
608ba22d585805f56ed6682c8a42392e  putty.cnt
c2bcb4d56fd986b1ad94b595817ae996  putty.hlp
07d07cc89c7b25229b3b999724bd3e5b  x86/plink.exe
51bd1bf6d054a01c8e556ecfa705b537  x86/pageant.exe
be49685e80862a4a49c06bb2545a6f4e  x86/puttygen.exe
476992da1f015286c4fc70e4a31f5bfc  x86/putty.zip
b9735750b270236fb5228f4a344b22ef  x86/pscp.exe
354d9abefa0ed67a08bd056324284d6e  x86/putty.exe
d83d91061fbd0a270883cee9ac6129a9  x86/psftp.exe
```

   SHA1:

```
653f7e757207a81d48ec9a46811b32d51a8a7d54  putty.chm
10a8d18c78a41b45a27caec959283b98dac1147c  putty.cnt
99990e52bce8b05d73f31a88baa67a8e94f5f4ef  putty.hlp
e79298341d580033c6011ee0eef51fd5c9693c6b  x86/plink.exe
bea6fedf753178170aa45708e902c2c48b9d5bc8  x86/pageant.exe
54c8601236dfe90b2e3e62f74155709d0231dcb2  x86/puttygen.exe
7fe800fdd10136adabb2ed5df80fe6a62759755c  x86/putty.zip
ec11523186a2cbba7bc3f68cba1a5ae36065fd4a  x86/pscp.exe
91b21fffe934d856c43e35a388c78fccce7471ea  x86/putty.exe
ac37ac13032672e59da953cabf14ceff9f2082a4  x86/psftp.exe
```

The binaries are intact and match the checksum.

Verify the PuTTY signature using GPG and RSA keys on page 15 to ensure they come from the author.

Install GnuPG on Windows

The GNU Privacy Guard (GnuPG) is a free replacement for PGP.

1. Go to http://gpg4win.org/download.html and download Gpg4win.
2. Install Gpg4win on your system.
   a) Run the Gpg4win package you have downloaded.
   b) Chose the installer language.
   c) Click **Next** to continue, then review the **License Agreement** and click **Next**.
   d) **Select the components to install** (leave default options as selected) and click Next.
   e) Select a **Destination Folder** and click **Next**.
   f) Check Start Menu under **Install Options** and click **Next**.
   g) Use Gpg4win under **Chose Start Menu Folder** and click **Next**.
   h) When prompted for **Installation Complete** click **Next**, then **Finish** to complete your installation.

You now have GnuPG on your system.

Verify the PuTTY signature using GPG and RSA keys on page 15

Verify the PuTTY signature using GPG and RSA keys

Verify that the PuTTY binaries are signed with the author's key.

Verify the signature:

```
gpg --verify putty.zip.RSA putty.zip
```

You have verified that the putty bundle is signed by the author.

Use PuTTY.

Install XMing on Windows

SSH Tips and Tricks

Encrypt, Decrypt and change your Private Key password

Should you need to change your password:

Decrypt your private key:

Using `openssl`:

```
openssl rsa -in ~/.ssh/id_rsa -out ~/.ssh/id_rsa
```

Using `ssh-keygen`, just provide an empty password:

```
ssh-keygen.exe -p
```

Encrypt your private key:

Using `ssh-keygen`, you can encrypt your password using AES-128-CBC. This can also be used to change your private key password:

```
ssh-keygen -p
```

You can encrypt your private key using different cipher using `openssl`:

```
openssl rsa -aes-256-cbc -in ~/.ssh/id_rsa -out id_rsa
```

Note: Older versions of ssh used 3DES as a default cipher.[9].

[9] See: http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdfr

Index

A

authorized_keys 11

D

DSA 10

P

pagent 12
PPK 7, 10, 12
pscp 13
PuTTY 6, 6
PuTTY Website 6
PuTTYgen 7, 10

R

RSA 10

S

SOCKS 13
SSH 3, 3

T

Tips 16
Tunnel 12

X

XMing 14