Contents

Using Telnet

1. Open the command prompt and type "telnet" (On Windows vista/7 you will need to install it from "programs and features").
2. connect to <Router_LAN_IP> e.g. 192.168.1.1 so in the command prompt, this would look like:

```
telnet 192.168.1.1
```

1. When asked for the username, enter *root* (even if you changed username in web interface)
2. When asked for the password, enter your router's password (default "admin")

SSH

Overview

SSH, or Secure Shell, is an encrypted protocol and associated program intended to replace telnet. It can also be used for creating secure tunnels, somewhat akin to Virtual Private Networks, and for use as a network file system (Sshfs). Unless changed, everything SSH operates on port 22.

SSH operates just as telnet with a user/password combination or on a Public/Private key infastructure. For the latter to work, a small public key is given to the server and the server gives your client its public key. Your client encrypts information to the server using the servers public key and the server encrypts information sent to you using your public key. Private keys are never exchanged, and are used to decrypt the information encrypted with the associated public key.

The DD-WRT firmware can use user/pass logon or only allows connections from clients whose public keys are manually entered via the web interface. Multiple keys can be entered by placing them on separate lines. If you want to use user/password to login using SSH use user "root" with the password you set in the webinterface

Actually you can manually set (via telnet or ssh) the sshd_authorized_keys nvram variable. ie nvram set sshd_authorized_keys=key1 key2 key3 etc

You can also manually edit /tmp/root/.ssh/authorized_keys and add keys (although these will disappear on a reboot unless you have a startup script altering them).

It is worth pointing out ssh keys are quite long strings of characters so if you paste them in you have to be careful that you don't get any line breaks (ie it is one Long continuous line). or they will not work.

Setting Up

Public key method

Public key authentication is one of the most secure methods of logging into SSH. It functions similar to HTTPS, as all transmissions are encrypted with a key that only the client and server will have. Another plus...if you use this method instead of password authentication, no one will be able to crack away at your router trying to guess the password!

To enable it, first you should generate a Public/Private key pair on your desktop machine. This can be done through the "Puttygen" utility if you're using either Putty or WinSCP as clients. Copy the **public key** to the clipboard and save the private key somewhere on your computer. There is no need to save the public key. If you forget it, you can instruct Puttygen to open your private key file rather than generating a new key pair and it will tell you your public key. Users of non-windows environments may use the ssh-keygen(1) utility:

```
user@machine:~> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
68:1c:50:0e:76:c1:d0:c7:9e:5e:5a:65:78:20:5c:fb user@machine.example.com
```

It is recommended that you don't secure your key pair with a password, as this will make things easier for you, although somewhat less secure.

1. Using the Web Interface, go to the **Administration** tab. (in v24 use **Services** tab)
2. Under the Services sub-tab, Enable **SSHd** in the Secure Shell section. If new options don't appear, Save Settings
3. Paste your **public key** in the **authorized key** of the **SSHD** section that has now expanded. You will need to generate this on your desktop if you don't have one yet.
4. Save and Apply Settings

NOTE: The format of the public key when pasted has to be "ssh-rsa", space, key, space, comment. Here is an example: (please note that there should be no line feed at the end)

```
 ssh-rsa AAAAB4NfaC3yc5 ... jZfYmBTi7Q== rsa-key-20101024
```

Alternate method:

```
Connect with ssh (login/password :0)
root@wrt54g:~# nvram set sshd_authorized_keys='ssh-rsa AAAAB4NfaC3yc5 ... jZfYmBTi7Q==  rsa-key-2
root@wrt54g:~# nvram commit
root@wrt54g:~# reboot
```

Remember to enter your key as an entire characters line (no space, tab...)

In Putty, you can enable key authentication by opening the SSH authentication configuration (Connection -> SSH -> Auth) and entering or browsing to your private key file. Also make sure your auto-login username is root (in Connection -> Data).

Password Login method

If you don't want the hassle of generating ssh keys, you may use the password logon method. However, please be aware that this method is much less secure! (passwords may be truncated to 8 characters or less)

1. Using the Web Interface, go to the **Administration** tab. (in v24 use **Services** tab)
2. Under the Services sub-tab, Enable **SSHd** in the Secure Shell section. If new options don't appear, Save Settings
3. Enable **Password Login** to enable the password login
4. Save and Apply Settings

After this you may login as user "root" with the password you set for the webinterface

Automatic Login (for shell scripts)

The Dropbear SSH client allows you to specify the password through an environment variable. This is useful when you need dd-wrt to auto-login to another host via SSH.

```
#the following requires dd-wrt v24 or later
DROPBEAR_PASSWORD='my password' ssh user@hostname
```

Security Tips

- Choose a random, non-standard port number >1024, especially if you enabled SSH access from the Internet! Most attackers will use a port scanner that only scans for common open ports by default. Scanning all 65535 ports is much slower for them, which makes it more difficult to find an attack vector and also more likely to be flagged by an Intrusion Detection System.
- Memorize, or record somewhere safe, your router's key fingerprint! In the process of logging into your router, if you see that the key fingerprint matches, you can rest assure noone is spying on your connection (i.e. via man-in-the-middle attack). If the key fingerprint does NOT match (your SSH client would likely notify you of this), something is wrong and you should consider terminating the connection immediately! (Note: the router's key fingerprint may change upon reset and/or upgrade, as it will likely generate a new key pair)
- For even more added security when using the public key method, you can password protect your private key. This way, if someone malicious happens to get ahold of it, they will still not be able to log into your router without first cracking the password of the key. Otherwise, if the keys are unprotected, anyone who stumbles upon them could likely gain immediate root access to your router and network.

SSH Shell Client

Provides a secure alternative to standard telnet.
A good Windows Client to use is Putty
Configure the client to use the Private Key you saved earlier.
Most Linux distros have telnet and SSH clients by default.

SSH Port Forwarding

SSH port forwarding is the ability to create encrypted tunnels to pass traffic through, sort of like a VPN. Below we will discuss two different approaches to SSH port forwarding; Local, and Remote

Local Port Forwarding

A real world example:

Suppose that you wish to manage your router's settings from anywhere over the Internet. You want to use a GUI interface, but you don't want to enable management via remote HTTP (insecure) or HTTPS (resource-intensive). How do you accomplish this?

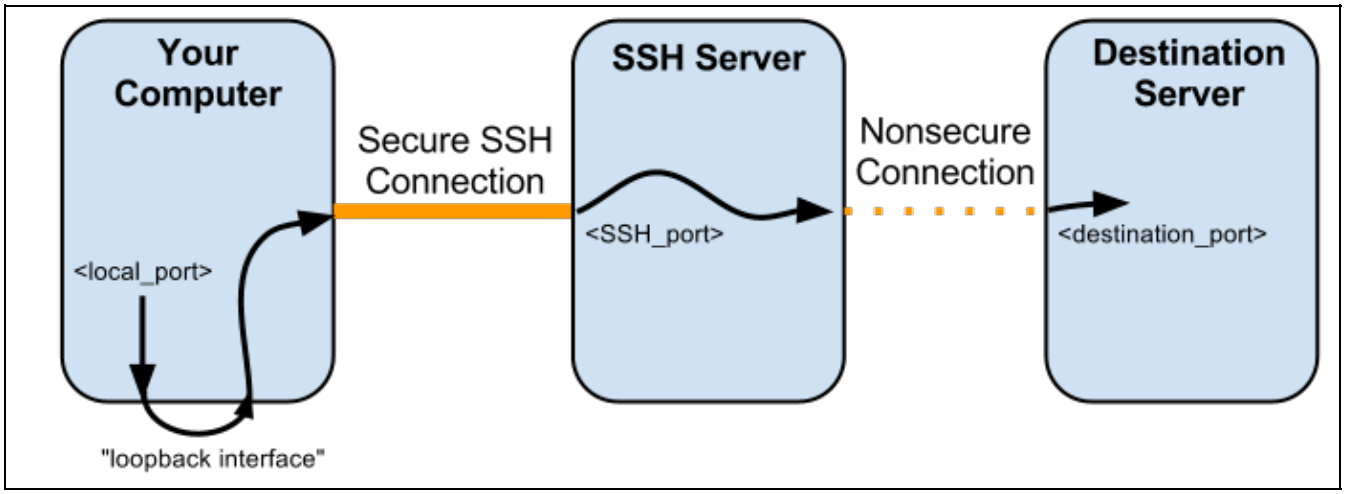
This is where SSH port forwarding comes in. This feature makes it possible to connect securely to the router's HTTP web interface, even when the interface has been configured to only be accessible by computers on the router's LAN.

First, 'Remote SSH Management' must be enabled under Administration -> Management if you wish to connect to your router from the WAN.

A local port forward can be established from the CLI with the following syntax:

```
ssh -L <local_port>:<destination_server>:<destination_port>
user@<ssh_server> -p <ssh_port>
```

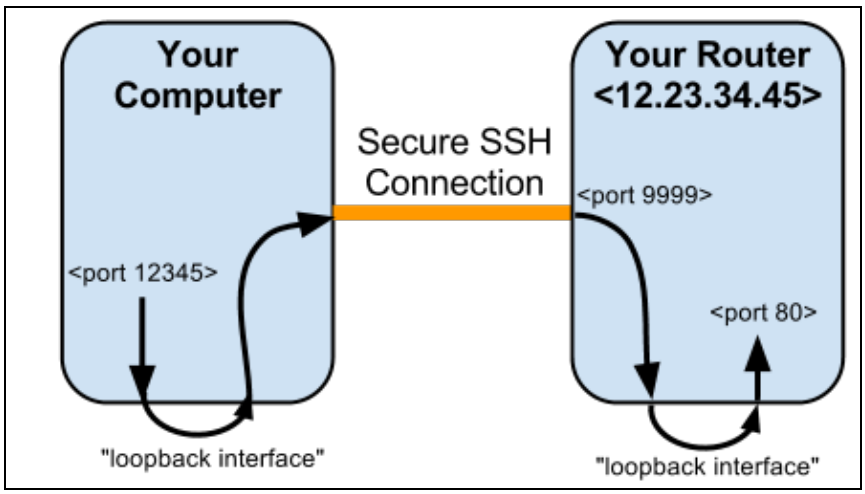
To explain more precisely what this command does: your computer establishes an SSH connection to <ssh_server>; a tunnel is created between your computer's <local_port>, the <ssh_server>, and <destination_port> on <destination_server>. Data sent to <local_port> is transferred over the secure SSH connection to the <ssh_server>, where it is then decrypted and forwarded to <destination_port> on <destination_server>.



For instance, if your router's WAN IP address is 12.23.34.35, its remote administration SSH port is 9999 and its LAN-accessible web interface is at port 80:

```
ssh -L 12345:localhost:80 root@12.23.34.45 -p 9999
```

The resulting connection:



Open your local browser window and point it to http://localhost:12345, and you should be able to log into the router's web interface as if you were on the router's local area network. This connection is secure!

If you're using PuTTY for SSH, the procedure is similar; SSH port forwarding is configured under SSH -> Tunnels

(NOTE: the PuTTY connection fails after web interface login when using PuTTY from the Ubuntu repositories, giving error: "SSH2_MSG_CHANNEL_FAILURE for nonexistent channel 257" --Brandonc 23:44, 2 August 2012 (CEST))

For more information related to the tunnel setup see here: Forum Discussion

Remote Port Forwarding

This is useful to tunnel things like RDP (Remote Desktop) through an encrypted SSH tunnel over the internet. For example, you want to be able to access your work computer from home.

If you had:

HomePC <-> Router <-> Internet <-> Firewall <-> WorkPC

WorkPC, which is running RDP on port 3389, issues *ssh -R 5555:localhost:3389 root@router.home*

HomePC can use his RDP client to connect to port 5555 on the router and this would create an SSH tunnel which will connect HomePC to port 3389 on the WorkPC.

Requirements

- DD-WRT v24 RC7+
- SSHd and SSH TCP Forwarding must be enabled under Services -> Secure Shell
- Remote SSH Management should be enabled as well, under Administration -> Management

Setup

Setting up a remote port forward is relatively straightforward when using the PuTTY utility under Windows. See Connections -> SSH -> Tunnels. Make sure your configuration includes parameters as illustrated above. Namely,

- Local and Remote ports should accept connections from other hosts
- Source port (port # on the router, should be > 1024)
- Destination IPAddress:Port
- Type: Remote

SCP

Secure Copy (SCP) allows one to copy files to and from the router and a remote host--usually a desktop machine.
Some good Windows clients to use are FileZilla and WinSCP.
Configure the client to use the Private Key you saved earlier, or use "root" and the webinterface password
Remember: only the /tmp and /jffs partitions are writable!

Drop Bear

DropBear is an SSH client/server installed by default on the WRT54G. DropBear allows one to connect from the WRT54G to a remote SSH server for scp, etc. I don't believe SSHD needs to be enabled through the Web Interface in order to use the client portion of DropBear.

If you have an SSH server on your desktop machine (such as OpenSSH) you pull files from your desktop machine using the scp command. This can be used to copy files from your desktop machine in a Startup Script

The DD-WRT Command Line

aka the DD-WRT Linux shell

This is an 'ash' shell. Ash is a version of sh, literally 'A SHell' (A command Interpreter)

Basic Syntax

The Linux Command Shell (Ash) is not the same as the Windows/DOS command prompt.

/ (and not \) is used to separate directories in a path, just like the interweb.

In order to execute a command, the path for that command must be provided. This may either be a full path or a relative path.

Relative Path Operators

There are two relative path operators.

```
.        The current path
..       One directory above the current path
```

Examples

1) If you are in the **/jffs/usr/bin** directory and wish to run the **/jffs/usr/bin/noip** command use:

```
/jffs/usr/bin # /jffs/usr/bin/noip
```

or

```
/jffs/usr/bin # ./noip
```

2) If you are in the **/jffs/usr/bin** directory and wish to run the **/jffs/usr/kismet** command use:

```
/jffs/usr/bin # /jffs/usr/kismet
```

or

```
/jffs/usr/bin # ../kismet
```

or

```
/jffs/usr/bin # cd ..
/jffs/usr # ./kismet
```

3) Relative paths can also be used as arguments. If you installed the noip package, you'd notice that the command is installed as **/jffs/usr/bin/noip** but its configuration file is installed as **/jffs/etc/no-ip.conf** When running noip, it is thus required to give it the path to its configuration file with the -c command. This can be done like:

```
/jffs/usr/bin # ./noip -c /jffs/etc/no-ip.conf
```

or

```
/jffs/usr/bin # ./noip -c ../../etc/noip.conf
```

notice that the first *../* brings us to /jffs/usr/. The second *../* brings us to /jffs/, and then the rest of the path can be appended.

4) While the other examples all showed how to save typing, you can also really screw around with relative paths. To launch the noip command in example 1, you could also use

```
/jffs/usr/bin # ../../../jffs/./usr/./bin/./././../bin/./././noip
```

Here we browse all the way back to the root **/** directory, then climb back up to **/jffs/usr/bin**, drop back down to **/jffs/usr** and then climb back up to **/jffs/usr/bin**.
Current path references of **/./** are thrown in sporadically just to mix things up. Notice how **/./** always references the then current path, not the original path of the shell when the command was entered.

Pipes and Redirects

The output of commands can be *piped* through other commands or redirected to devices and files.

< and > are the redirect operators. < Takes input from a device or file and routes it as input to the command given. > Takes output from a command and redirects it as input for a device or file. Ex: If you don't want to see the output of a command, redirect it to the null device:

```
command > /dev/null
```

| is the pipe character, and pipes the output through another command (for formatting, etc) Ex: the most common use of the pipe is to limit the output of a command:

```
command | more
```

This is extremely useful for commands like *nvram show* which list some 800-1200 lines. *nvram show | more* will list the results 1 page at a time.

Background processes

It is possible to run programs in the background (returning you to the command prompt immediately) by terminating your command with the & character. ex:

```
command &
```

Make sure you add a space between your command and the ampersand or you will result with a *File not found* error.

WEB-GUI (http[s]) Special note

The built-in WEB-GUI command line interface (Diagnostics.asp page) allows only about 200 characters max per line.
Special characters such as " or | must be entered after a \
Example, if you want to set a text nvram value:
Instead of

```
nvram set svqos_svcs="edonkey p2p 0:0 40 | bittorrent p2p 0:0 40 |"
```

Enter

```
nvram set svqos_svcs=\"edonkey p2p 0:0 40 \| bittorrent p2p 0:0 40 \|\"
```

Basic Commands

```
<command> -h                 The -h flag almost always provides help on a command. Use it!
ls                           List the contents of the current directory
cd <directory or full path>  Change to that directory or path
cp <source> <destination>    Copy the source file to the destination
cp -r <source> <destination>    Copy the source directory to the destination directory
mv <source> <destination>    Move the source file to the destination
mkdir <directory name>       Create a new directory
wget <URI>                   Download the file at the given URI to the current path
tar -xz -f <file>            un-gzip and un-tar the given *.tgz or *.tar.gz file
rm <file>                    Delete the file
rm -r <directory>            Delete the directory and all contents
killall <program name>       Kill all running processes of the program
ps                           Show running processes
top                          Show running processes in a graphical frontend
```

More Advanced Commands

These commands warrant their own wikis:

- brctl
- cut
- expr
- dnsmasq
- ip
- ifup
- ifconfig
- iptables
- scp
- tc
- udhcpd
- wl

See also

Script Examples
Sshfs
Startup Scripts
SSH access from internet
Tunnel all traffic over ssh using remote windows machine and Putty

External Links

Wikipedia's SSH article
Linux Shell Scripting Tutorial
Telnet/SSH BusyBox Commands
Configuring FTP/Telnet
Increasing number of Telnet sessions allowed