Remote Access

From Termux Wiki
Jump to: navigation, search

Termux is capable of accessing remote devices by using some common tools. It is also possible to turn a device running Termux into remote controlled server.

FTP

Warning: plain FTP is deprecated and insecure anyway. Termux FTP server supports only anonymous login, there no any authentication and everyone on your network can access files on your device. Use SFTP (OpenSSH) instead!

Termux FTP server is based on busybox and service is managed by [Termux-services]. If you decided to use FTP server, install these packages:

```
pkg install busybox termux-services
```

After installation you need to restart session or source this file:

```
source $PREFIX/etc/profile.d/start-services.sh
```

Now you ready to enable and start the FTP daemon service:

```
sv-enable ftpd
sv up ftpd
```

FTP server will run on port 8021 in read-only mode.

If you need to stop server, run `sv down ftpd` .

SSH

SSH provides a secure way for accessing remote hosts and replaces tools such as telnet, rlogin, rsh, ftp. Termux provides SSH via two packages: **dropbear** and **openssh**. If you never used these tools before, it is recommended to install 'openssh' as it is more common.

Using the SSH client

You can obtain an SSH client by installing either `openssh` or `dropbear`.

Usage example

To login to a remote machine where the ssh daemon is running at the standard port (22):

```
ssh user@hostname_or_ip
```

Same as above, but if the ssh daemon running on different port, e.g. 8022:

```
ssh -p 8022 user@hostname_or_ip
```

Using public key authentication with ssh running on the standard port and a private key stored in the file `id_rsa`:

```
ssh -i id_rsa user@hostname_or_ip
```

Note, that if `id_rsa` will be stored in `~/.ssh` directory, you can omit specifying it in the command. But if you have multiple keys, it is necessary to pick a specific key with `-i {path_to_privkey}`.

SSH Agent

Important note: this does not work for Dropbear.

If you wish to use an SSH agent to avoid entering passwords, the Termux openssh package provides a wrapper script named `ssha` (note the `a` at the end) for ssh, which:

- Starts the ssh agent if necessary (or connect to it if already running).
- Runs the `ssh-add` if necessary.
- Runs the `ssh` with the provided arguments.

This means that the agent will prompt for a key password at first run, but remember the authorization for subsequent runs.

Using the SSH server

OpenSSH

OpenSSH (also known as OpenBSD Secure Shell) is a suite of secure networking utilities based on the Secure Shell (SSH) protocol, which provides a secure channel over an unsecured network in a client–server architecture.

Default SSH port in Termux is 8022.

Starting and stopping OpenSSH server

Since Termux does not use initialization system, services are started manually from command line.

To start OpenSSH server, you need to execute this command:

```
sshd
```

If you need to stop `sshd`, just kill it's process:

```
pkill sshd
```

SSH daemon does logging to Android system log, you can view it by running `logcat -s 'sshd:*'`. You can do that either from Termux or ADB.

Setting up password authentication

Password authentication is enabled by default. This will allow you to get started with it much easier. Before proceeding, make sure that you understand that password authentication is less secure than a pubkey-based one.

1. Ensure that everything is up to date and package `openssh` is installed:

```
pkg upgrade
pkg install openssh
```

2. Password authentication is enabled by default in configuration file. But you can still review it ($PREFIX/etc/ssh/sshd_config), it should be like this:

```
PrintMotd yes
PasswordAuthentication yes
Subsystem sftp /data/data/com.termux/files/usr/libexec/sftp-server
```

3. Set new password. Execute command `passwd`. While program allows minimal password length is 1 character, the recommended password length is more than 8-10 characters. Passwords are not printed to console.

```
$ passwd
New password:
Retype new password:
New password was successfully set.
```

Setting up public key authentication

Public key authentication is the recommended way for logging in using SSH. To use this type of authentication, you need to have a public/private key pair. For successful login, the public key must exist in the authorized keys list on remote machine while private key should be kept safe on your local host.

In the following example it will be assumed that you want to establish public key authentication between your PC (host) and your Android device running Termux (remote). It also will be assumed that you running Linux distribution on your PC.

1. If you do not have keys, you can generate them. In this example we will generate RSA key. On PC, execute this command:

```
ssh-keygen -t rsa -b 2048 -f id_rsa
```

The command shown above generates private RSA key with 2048 bit key length and saves it to file `id_rsa`. In the same directory you can find a file `id_rsa.pub` – it is a public key.

Important note: 2048 bit is the minimal key length that is considered safe. You can use higher values, but do not use higher than 4096 as remote server may not support big keys.

2. Copy key to the remote machine (Termux). Password authentication has to be enabled in order to install pubkey on remote machine. Now do:

```
ssh-copy-id -p 8022 -i id_rsa IP_ADDRESS
```

Do not forget to replace `IP_ADDRESS` with the actual LAN IP address of your device. It can be determined by using command `ifconfig`.

- Alternative you can manually copy the content inside "id_rsa.pub"(public key) which is already on PC and looks like `ssh-rsa <A LOT OF RANDOM STRINGS> user@host` and paste to the Termux file `$HOME/.ssh/authorized_keys` (remote machine) Remember to connect through `ssh user@<Phone_IP> -p 8022` so you can copy the content of public key using any text editor available on PC and paste inside Termux

If everything was okay, you will see a message like this one:

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh -p '8022' '192.168.1.4'"
and check to make sure that only the key(s) you wanted were added.
```

3. From this point password authentication can be disabled. Edit file $PREFIX/etc/ssh/sshd_config and replace line beginning with "PasswordAuthentication" by

```
PasswordAuthentication no
```

Then execute command `pkill sshd; sshd` in order to restart server with updated configuration file.

Dropbear

Dropbear is a software package written by Matt Johnston that provides a Secure Shell-compatible server and client. It is designed as a replacement for standard OpenSSH for environments with low memory and processor resources, such as embedded systems.

Important note: Dropbear does not provide SFTP server.

Starting and stopping Dropbear server

Same as for OpenSSH, you will need to execute it's binary manually. Also, unlike OpenSSH, Dropbear does not use a configuration file but only command line arguments.

Server is running in background, both password and public key authentication available. To achieve this, just type in console:

```
dropbear
```

If you need only public key authentication, do this instead:

```
dropbear -s
```

Also, server can be started in foreground. For this purpose use a parameter `-F`:

```
dropbear -F
```

Server started in foreground can be stopped by just Ctrl-C key combination. If it is in the background, then you can use a `pkill`:

```
pkill dropbear
```

Setting up password authentication

Same as for OpenSSH, password authentication is enabled by default.

Everything you have to do, is:

1. Make sure that everything is up to date and dropbear is installed:

```
pkg upgrade
pkg install dropbear
```

2. Set password by executing command `passwd`.

3. Start dropbear server. You can execute either just `dropbear` to start it in background or `dropbear -F` to start it in the foreground.

Setting up public key authentication

Same as for OpenSSH, you can put your keys by using `ssh-copy-id`. But if you consider to setup a public key authentication from Termux to something else, it is worth to mention some important differences between OpenSSH and Dropbear.

1. Dropbear uses a different command for generating keys. Example of generating RSA key (2048 bit):

```
dropbearkey -t rsa -f id_rsa -s 2048
```

2. Public key should be obtained manually. To do this, you have to use 'dropbearkey' again, but in different way:

```
dropbearkey -f id_rsa -y
```

3. Dropbear and OpenSSH uses a different key formats. To use a Dropbear's key in OpenSSH, you will have to convert it:

```
dropbearconvert dropbear openssh ./id_rsa ./id_rsa_openssh
```

This procedure can be done vice versa to obtain a key in Dropbear's format:

```
dropbearconvert openssh dropbear ./id_rsa_openssh ./id_rsa_dropbear
```

Using the SFTP

Package OpenSSH provides a tool for accessing remote hosts over SFTP. This will allow you to work with files in same way as via FTP but with better security.

First install openssh-sftp-server

```
$ pkg install openssh-sftp-server
```

Connecting to Termux (sshd listening on port 8022):

```
$ sftp -P 8022 192.168.1.20
```

Connecting to somewhere else (sshd listening on standard port):

```
$ sftp sftp.example.com
```

However, to use command line SFTP client you should know some basic commands:

- **cd** PATH - change current directory to `PATH`.
- **get** REMOTE [LOCAL] - download file `REMOTE` and rename it as `LOCAL` (optional).
- **mkdir** PATH - create directory `PATH`.
- **ls** [PATH] - list files in directory `PATH`. If no argument, files in current directory will be listed.
- **put** LOCAL [REMOTE] - Upload file `LOCAL` and rename it as `REMOTE` (optional).
- **rm** FILE - Delete file `FILE`.

This is not a complete list of SFTP commands. To view all available commands, consider to view man page (`man sftp`) or view short help in interactive SFTP session by issuing command `help`.

MOSH

Mosh is a remote terminal application that allows roaming, supports intermittent connectivity, and provides intelligent local echo and line editing of user keystrokes.

Usage example

Important note: Mosh should be installed on both client and server side.

Connecting to remote host (sshd listening on standard port):

```
mosh user@ssh.example.com
```

Connecting to Termux (sshd listening on port 8022):

```
mosh --ssh="ssh -p 8022" 192.168.1.25
```

Rsync

Rsync is a tool for synchronizing files with remote hosts or local directories (or drives). For better experience of using rsync, make sure that package `openssh` (or `dropbear`) is installed.

Usage example

Sync your photos with PC:

```
$ rsync -av /sdcard/DCIM/ user@192.168.1.20:~/Pictures/Android/
```

Get photos from remote Android device:

```
$ rsync -av -e 'ssh -p 8022' 192.168.1.3:/sdcard/DCIM/ /sdcard/DCIM/
```

Sync local directories (e.g. from external sdcard to Termux home):

```
$ rsync -av /storage/0123-4567/myfiles ~/files
```

You may want to see man page (`man rsync`) to learn more about it's usage.

See Also

Accessing Termux from the Internet (/wiki/Bypassing_NAT)

Connecting to Termux with SSH over USB (https://glow.li/technology/2016/9/20/access-termux-via-usb/)