How To Set Up Chrooted SSH

While this document was written for Debian GNU/Linux systems it will most likely as well work on other Linux systems (you just might have to adjust a couple of filenames)!

You will find the most up to date version of this document at http://www.tokkee.de/index.php?id=520.

This implementation uses PAM (Pluggable Authentication Modules). There is also a patch for OpenSSH that allows chrooting of users to their home directories. You can find more information about this patch at http://chrootssh.sourceforge.net/. Well, I think that the PAM implementation is slighly more flexible.

Required packages:

- `ssh ;)`
- `libpam-chroot`

Configuring PAM:

Add the following line to /etc/pam.d/ssh:

```
session required pam_chroot.so
```

Here's how this file looks like on my system:

```
# PAM configuration for the Secure Shell service

# Disallow non-root logins when /etc/nologin exists.
auth       required     pam_nologin.so

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
auth       required     pam_env.so # [1]

# Standard Un*x authentication.
@include common-auth

# Standard Un*x authorization.
@include common-account

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
session    optional     pam_motd.so # [1]
```

```
# Print the status of the user's mailbox upon successful login.
session    optional     pam_mail.so standard noenv # [1]

# Set up user limits from /etc/security/limits.conf.
session    required     pam_limits.so

# Set up user chroot from /etc/security/chroot.conf.
session    required     pam_chroot.so

# Standard Un*x password updating.
@include common-password
```

Now, change /etc/security/chroot.conf according to your needs. This file determines which users will be chrooted and which directories they are chrooted to. The format of this file is:

```
username chroot_dir
```

e.g.

```
foo /home/foo
```

You may also use regular expressions instead of the username, if you specified the use_regex option. To do so change the appropriate line in /etc/pam.d/ssh to look like this:

```
session required pam_chroot.so use_regex
```

Configuring SSH:

In /etc/ssh/sshd_config you have to enable PAM ;) The following line should be in the file:

```
UsePAM yes
```

In addition we need to turn privilege separation off in order to be able to chroot at all (chroot needs root privileges):

```
UsePrivilegeSeparation no
```

If you can grant the sshd user privileges to make the chroot(2) system call, you should leave privilege separation turned on due to security reasons.

My configuration file looks like this:

```
# Package generated configuration file
# See the sshd(8) manpage for defails

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
```

```
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
#However we need to turn it off for chroot to work
UsePrivilegeSeparation no

# ...but breaks Pam auth via kbdint, so we have to turn it off
# Use PAM authentication via keyboard-interactive so PAM modules can
# properly interface with the user (off due to PrivSep)
#PAMAuthenticationViaKbdInt no
# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 600
PermitRootLogin no
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys

# rhosts authentication should not be used
#RhostsAuthentication no
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Uncomment to disable s/key passwords
#ChallengeResponseAuthentication no

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes


# To change Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no

# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes
```

```
X11Forwarding no
X11DisplayOffset 10
PrintMotd no
#PrintLastLog no
KeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net
#ReverseMappingCheck yes

Subsystem sftp /usr/lib/sftp-server


UsePAM yes
```

Setting up a chroot environment:

Last but not least, we need to set up a chroot(8) environment. If you skip this step, a user will not have any tools to work with (including their shell) after logging in and will be logged off immediately.

```
$ ssh -l user host
Password:
[...]
/bin/bash: No such file or directory
Connection to host closed.
```

You will at the very least need the user's shell, e.g. bash, the most important libraries and a copy of the user's homedirectory with the shell and login configuration files. The following directory listing shows the absolut minimal requirements I needed on my Debian GNU/Linux Sid box:

```
/home/foo
|
+-- bin
|   +-- bash
|   `-- sh -> bash
|
+-- home
|   `--foo
|      `-- .bash_profile
|
`-- lib
    +-- ld-linux.so.2
    +-- libc.so.6
    +-- libdl.so.2
    `-- libncurses.so.5
```

However, this setup does not give much functionality at all. Pretty much all you can do is execute shell scripts. The following listing shows a much smarter setup:

```
/home/foo
|
+-- bin
|   +-- bash
```

```
|   +-- cat
|   +-- cp
|   +-- false
|   +-- grep
|   +-- gunzip
|   +-- gzip
|   +-- ls
|   +-- mkdir
|   +-- mv
|   +-- rm
|   +-- rmdir
|   +-- sed
|   +-- sh -> bash
|   +-- tar
|   +-- touch
|   `-- true
|
+-- dev
|   +-- null
|   `-- zero
|
+-- etc
|   +-- bash_completion
|   +-- group
|   +-- hosts
|   `-- passwd
|
+-- home
|   `-- foo
|       +-- .alias
|       +-- .bash_history
|       +-- .bash_profile
|       +-- .bashrc
|       `-- .profile
|
+-- lib
|   +-- ld-linux.so.2
|   +-- libacl.so.1
|   +-- libattr.so.1
|   +-- libc.so.6
|   +-- libdl.so.2
|   +-- libncurses.so.5
|   +-- libpthread.so.0
|   `-- librt.so.1
|
+-- usr
    +-- bin
    |   +-- less
    |   `-- vim
    |
    `-- lib
        `-- libgpm.so.1
```

You can create the null and zero devices using the following commands:

```
$ su
Password:
# cd /home/foo/dev
# mknod null c 1 3
```

```
# mknod zero c 1 5
# chmod 666 null zero
```

Most importantly, do not install any suid binaries to your chroot environment as this is a big security risk.

That's it - Enjoy!

Any comments are welcome.