CHROOTED SFTP - LOCKING DOWN

Openssh is a wonderful tool, providing encrypted links for a variety of fun connections. Once of the most useful drop-in replacements it can be used for is to replace *ftp*. Using the *sftp* subsystem of the *openssh* suite overcomes all the cleartext problems associated with normal ftp. But lets take it a little further, lets *"jail"* each user so that they can only see their own little part of the server - literally. This is called *chrooting*, or "*changing the root*". What happens is that the user, sees the start of their home folder (or as otherwise specified) as the root (/) folder of the server, making *sftp* just a bit more secure..

What is needed
Well you will need a linux box (Unix works similarily and BSD's have a different way of doing this) and thats about it. If you already have *Openssh* on the server you can over-install, although it can be easier to remove it and then install cleanly (either way make sure you keep a copy of the configuration files and host keys). You will then need to get the patched version of the *Openssh* package. You can either download the main source code package and then apply the patch, or download a fully patched version. We will be using the fully patched version in this example (we will also be using the latest *zlib* libraries). So get the patched *Openssh* from here, and the *zlib* libraries from here.

Installing
First we do the *zlib* stuff..

```
# gzip -d zlib-1.2.3.tar.gz
# tar -xf zlib-1.2.3.tar
#./configure && make && make install
```

Now we do the patched *openssh* (remember if you use any fancy options with your ssh then choose the appropriate options)..

```
# gzip -d openssh-4.2p1-chroot.tar.gz
# tar -xf openssh-4.2p1-chroot.tar
#./configure && make && make install
# ssh -V
OpenSSH_4.2-chrootsshp1, OpenSSL 0.9.7a Feb 19 2003
```

Now check to make sure that your *ssh* service can start up as a service (*/etc/init.d/sshd restart*), and make any changes you need to get it to startup. Once that is all working, then we can move on to creating the actual chroot jail.

Theory
Before we get the that though, lets just do a bit of theory. When you create a chroot jail, you are actually limiting the software to a specific set of folders. This has to do with the actual programming calls the software makes, but that is a bit too much detail for this setup. But once you have the software which uses the chroot system calls (in this case the patched openssh tarball), you need to make sure that the utilities and programs used by it have the libraries moved to the right place. You can see what libaries a certain program needs by using *ldd*..

```
# ldd /bin/bash
        linux-gate.so.1 =>  (0xffffe000)
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x005a4000)
        libdl.so.2 => /lib/libdl.so.2 (0x00468000)
        libc.so.6 => /lib/tls/libc.so.6 (0x0031a000)
        /lib/ld-linux.so.2 (0x00301000)
```

Here we can see what libraries the binary */bin/bash* needs to function, so if we are going to use that in our jail, we need to make sure that the libraries needed are also copied. This is why sometimes setting up the jail for the first time can be a bit a trial and error.

Making the jail
In my example I want each person who has *sftp* access must be in their own jail, and must only be able to use

sftp. First lets make sure that they can only use *sftp* when they login..

```
echo "/usr/local/libexec/sftp-server" >> /etc/shells
```

This allows your *sftp-server* (check where it is on yor system) binary to be specified as a shell in the */etc/passwd* file. This can be a bit ugly as if they try to login with anything else other then *sftp*, the session just hangs. But thats fine by me. Now we will move onto the full jail setup. I have a script I use for this, it creates the jail, creates the user and copies the binaries I want them to have access to into the jail. As always this script is not a work of art, but it works for me, please feel free to adapt, improve, simplify, etc as you see fit..

```
#WE NEED THE USERS NAME AS FIRST SETTING
#AND THE PASSWORD AS THE SECOND
#USAGE: setup-chroot bob bob
NAME=$1
PASS=$2

#CREATE INITIAL FOLDERS
mkdir -p /data/lockhome/$NAME/$NAME
cd /data/lockhome/$NAME
mkdir etc
mkdir bin
mkdir lib
mkdir usr
mkdir usr/bin
mkdir dev
mkdir -p ./usr/local/bin
mkdir -p ./usr/local/libexec/
mknod dev/null c 1 3
mknod dev/zero c 1 5

#COPY IN WANTED BINARIES AND RELATED LIBRARIES
cd /data/lockhome/$NAME
for bins in /bin/bash /usr/local/libexec/sftp-server /bin/ls /bin/mkdir /bin/mv /bin/rm /bin/rmdir
 do
  cp -v --reply=yes $bins ./$bins
  ldd $bins > /dev/null
  if [ "$?" = 0 ]
   then
    LIBS=`ldd $bins | awk '{ print $3 }'`
    for l in $LIBS
     do
      mkdir -p ./`dirname $l` > /dev/null 2>&1
      cp -v --reply=yes $l ./$l
     done
    LIBS2=`ldd $bins | grep -v -e "=>" | awk '{ print $1 }'`
    for u in $LIBS2
     do
      mkdir -p ./`dirname $u` > /dev/null 2>&1
      cp -v --reply=yes $u ./$u
     done
   fi
 done

#TIDY UP AND DO CREDENTIALS
cd bin
ln -s ./bash ./sh
cd ..
touch etc/passwd
grep /etc/passwd -e "^root" > etc/passwd

#CREATE USER
/usr/sbin/useradd -s /usr/local/libexec/sftp-server -m -d /data/lockhome/$NAME/./$NAME/$NAME $NAME
echo $PASS | passwd $NAME --stdin
grep /etc/passwd -e "^$NAME" >> /data/lockhome/$NAME/etc/passwd
```

Now when you login, you should only be able to use *sftp* and you should only be able to see what is in your jail.

Logging

You may want to do logging on your *sftp* usage as with a normal ftp server, the bad news this is not default behaviour, the good news is that you can do it. You will need to download a patch from here and follow a couple of simple steps. Lets first do the download and patch (I am assuming you already have *openssh* installed)..

```
cd openssh-x.x-chroot
wget http://sftplogging.sourceforge.net/download/v1.4/openssh-x.x.sftplogging-v1.4.patch
make clean
patch < openssh-x.x.sftplogging-v1.4.patch
./configure
make && make install
```

Now if you have previously created any *chroot* jails you will need to update the *sftp-server* binary which they use. Ok, no we need to setup the *ssh* service to use the sftp logging, so edit your *sshd_config* file (wherever you have it)..

```
 #vi /usr/local/etc/sshd_config

--now make sure you have the following section (these settings work for me, tweak them to fit your
needs)
LogSftp yes
#SftpLogfacility # Default is AUTH, see /etc/syslog.conf for more info
SftpLogLevel DEBUG3# Default is INFO
#SftpUmask # Set a global umask for sftp
SftpPermitChmod yes # Allow, or disallow chmod
SftpPermitChown yes # Allow, or disallow chown/chgrp

# /etc/init.d/sshd restart
```

The last step is you need to get the data out of the chroot jail into the normal syslog process. You first need to create a */dev/log* file in the */dev* folder of the chroot jail...

```
mksock /<chroot jail>/dev/log
```

Then update your *syslog* settings to use the new socket, this can be done by editing the *init* script or by editing a configuration file if your system uses one. Whichever way your *syslog* works you need to add *"-a /<chroot jail>/dev/log"* to your *syslog* daemon startup options. Once this is done, restart your *syslog* service. When you're finsihed, your new chroot'ed sftp account will have the sessions logged to your syslog log file.

Final Words

While this is a very useful setup, bear in mind that there are still ways to break out of a chroot jail, nothing is ever really 100%. But it does continue raising the bar on your systems a bit higher, making attackers work harder to get anywhere - thats always a good thing. This really is something you can play around with, and can use in many other instances, so as always, have fun and learn.