Lab 2 - Chroot Jail

Table of contents

1 Introduction

A chroot jail is a security mechanism that creates an isolated environment (a jail) for a running process and its children where the process cannot access files or commands outside of jail. Reasons for creating such an environment include providing limited resources for FTP and SSH users along with creating a clean environment for testing software.

2 A Simple Jail with Bash

First we setup a simple jail that allows the user to execute specific bash commands we allow.

1. Become the root user.
   ```
   sudo -i
   ```
2. Create the directory for the jail.
   ```
   mkdir /var/jail
   ```

3. Navigate to the jail.

```
cd /var/jail
```

4. Everything we want the user to access must be in the jail. We will want them to have access to the bash interpreter. Let's check the dependencies of the bash interpreter with the `ldd` command so that we can determine what to add to the jail.

```
ldd /bin/bash
```

   The output should be similar to the following depending on your setup:

```
linux-vdso.so.1
libtinfo.so.6 => /lib/aarch64-linux-gnu/libtinfo.so.6
libc.so.6 => /lib/aarch64-linux-gnu/libc.so.6
/lib/ld-linux-aarch64.so.1
```

   or

```
linux-vdso.so.1
libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
/lib64/ld-linux-x86-64.so.2
```

5. While in the /var/jail, create directories that mimic the outputs above. Only create the appropriate directories for your machine.

```
# ARM Machines:
mkdir bin lib
```

```
# x86-64 Machines:
mkdir bin lib lib64
```

6. Copy the bash dependencies and bash into the appropriate directories

```
# ARM Machines:
cp /lib/aarch64-linux-gnu/libtinfo.so.6 lib
cp /lib/aarch64-linux-gnu/libc.so.6 lib
cp /lib/ld-linux-aarch64.so.1 lib
cp /bin/bash bin
```

```
# x86-64 Machines:
cp /lib/x86_64-linux-gnu/libtinfo.so.6 lib
cp /lib/x86_64-linux-gnu/libc.so.6 lib
cp /lib64/ld-linux-x86-64.so.2 lib64
cp /bin/bash bin
```

7. Add the ls command to the jail.

i. Determine the dependencies

```
ldd /bin/ls
```

ii. Copy the items you haven't already copied over.

```
# ARM Machines:
cp /lib/aarch64-linux-gnu/libselinux.so.1 lib
cp /lib/aarch64-linux-gnu/libpcre2-8.so.0 lib
cp /bin/ls bin
```

```
# x86-64 Machines:
cp /lib/x86_64-linux-gnu/libselinux.so.1 lib
cp /lib/x86_64-linux-gnu/libpcre2-8.so.0 lib
cp /bin/ls bin
```

8. Now go to jail

```
chroot /var/jail
```

9. See if you can use `ls` to view the contents.
10. Bash built-ins such as `pwd` and `cd` should work. Test each out.
11. Try to run commands that aren't any of the built-in bash commands. They should not work.

```
touch some-file
mkdir some-dir
nano
vim
sudo apt list
```

You should get error messages on each of those.

12. Exit the jail with `ctrl + d` or `exit`.

3 A Robust Jail

Adding everything we want to the jail manually can be time consuming. In this section, we provide a script that you can use to add everything we will need to the jail.

1. Make sure you are the root user. Run the follow if you are not.

```
sudo -i
```

2. Navigate home.

```
cd ~
```

3. Remove the simple jail, since the script will recreate it.

```
rm -rf /var/jail
```

4. Create a file named **jailsetup.sh** and place the following bash script in it. Note that this script works regardless of your machine's architecture. The script can also be found on Canvas.

```
#!/bin/bash
CHROOT='/var/jail'
mkdir -p "$CHROOT"

# loop over the args provided on execution of this file
# these args are the files we want available in the jail
for file in "$@";
do
    if [ -x "$file" ] && file "$file" | grep -q 'ELF'
    then
    # For executable files:
        ldd "$file" |      # get the shared libraries for the file
        awk '{print $3}' | # extract 3rd column (the library filepath)
        grep -E '^/' |     # keep absolute paths (lines starting with /)
        sort -u |          # remove duplicates
        while read lib;
        do
        # copy the library and preserve its directory structure
        cp --parents "$lib" "$CHROOT"
        done
        # copy the file and preserve its directory structure
        cp --parents "$file" "$CHROOT"
    else
    # for non-executable files
    # just copy the file and preserve its directory structure
        cp --parents "$file" "$CHROOT"
    fi
done

# Copy the dynamic linker for the architecture of your machine.

# for aarch64 (arm)
if [ -f /lib/ld-linux-aarch64.so.1 ]; then
```

```
    cp --parents /lib/ld-linux-aarch64.so.1 "$CHROOT"
fi

# for amd64 (x86-64)
if [ -f /lib64/ld-linux-x86-64.so.2 ]; then
    cp --parents /lib64/ld-linux-x86-64.so.2 "$CHROOT"
fi

# for i386 (x86-32)
if [ -f /lib/ld-linux.so.2 ]; then
    cp --parents /lib/ld-linux.so.2
fi

echo "Ready. Enter the jail with: chroot $CHROOT"
```

5. Give jailsetup.sh executable permissions.

```
chmod +x jailsetup.sh
```

6. Run jailsetup.sh with the following provided input to setup the jail. Modifying the inputs will allow for more or less capability.

```
./jailsetup.sh  /bin/{ls,cat,rm,mkdir,bash} /usr/bin/{vim,nano,whoami,scp} /etc/hosts
```

7. Add some useful special files.

   i. Create the device folder for our special nodes.

```
mkdir /var/jail/dev
```

   ii. Create the /dev/null node for discarding output.

```
mknod -m 0666 /var/jail/dev/null c 1 3
```

   iii. Create the /dev/random node for random value generation.

```
mknod -m 0666 /var/jail/dev/random c 1 8
```

   iv. Create the /dev/urandom node for pseudo random value generation.

```
mknod -m 0444 /var/jail/dev/urandom c 1 9
```

8. Now we add a new user to the system to test everything out.

   i. Create a new user group called `chrootjail`

```
groupadd chrootjail
```

ii. Create a new user named **newuser**. Note that you will be prompted for a password and other information about the new user. You can leave the extra information blank.

```
adduser newuser
```

iii. Add **newuser** to the chrootjail group.

```
adduser newuser chrootjail
```

9. Confirm the user was added by listing out the files and folders at **/home**. You should see a folder for the newuser.

10. Add the new user's home directory to the jail.

```
mkdir -p /var/jail/home/newuser
```

11. Copy default shell configuration files to the user's home directory.

```
cp -r /etc/skel/  /var/jail/home/newuser
mv /var/jail/home/newuser/skel/.*  /var/jail/home/newuser
rm -rf /var/jail/home/newuser/skel/
```

12. Make the new user the owner of the new user's home directory.

```
chown -R newuser:newuser /var/jail/home/newuser
```

13. Copy the passwords and group files to the jail.

```
cp /etc/passwd /etc/group /var/jail/etc
```

14. Complete the following to remove users and groups that we don't need in the jail.

i. Open **/var/jail/etc/passwd** with vim or nano.

ii. Remove everything that does not contain references to **root**, **newuser**, and **chrootjail**.

iii. Save and close the file.

iv. Open **/var/jail/etc/group** with vim or nano.

v. Remove everything that does not contain references to **root**, **newuser**, and **chrootjail**.

vi. Save and close the file.

15. If it isn't already installed, install `openssh-server`.

```
sudo apt install openssh-server
```

16. Alter the ssh configuration to jail users in the chroot jail group.

    i. Open `/etc/ssh/sshd_config` with vim or nano.

    ii. Add the following to the configuration file.

```
Match group chrootjail
    ChrootDirectory /var/jail/
    X11Forwarding no
    AllowTcpForwarding no
```

    iii. Save and close the file.

    iv. Restart the ssh server.

```
service ssh restart
```

17. Login via ssh with `newuser`.

```
ssh newuser@localhost
```

4 Testing

Once logged in, try the following to test out the chroot jail.

1. Navigate to the root directory and view the contents.

```
cd /
ls
```

   You should only see the directories that were added to the chroot jail. Compare this with what is normally found at the root directory.

2. Try to navigate higher.

```
cd ..
```

   You should see that you cannot navigate higher.

3. Try printing the working directory to see if it shows any information about the system above the chroot jail's root directory.

```
pwd
```

   You should see that it doesn't.

4. Try switching to the root user.

```
sudo -i
```

   You should see that the sudo command doesn't exist in the chroot jail since we didn't add it.

5. Test anything else you would like to. If you would like to add any additional commands to the jail, simply run the `jailsetup.sh` script again with the appropriate arguments. For example, if you want to add `touch` you would run `./jailsetup.sh /bin/touch`.

6. Once done, disconnect from the ssh server and proceed to the cleanup section of this document.

5 Cleanup

To reset everything back to how it was prior, complete the following steps. Be sure you are not still connected through ssh with the newuser account.

1. Switch to the root user.

```
sudo -i
```

2. Remove the contents we added to the `sshd_config` file.

   i. Open `/etc/ssh/sshd_config` with vim or nano.

   ii. Find the lines that contain the following and remove them.

```
Match group chrootjail
    ChrootDirectory /var/jail/
    X11Forwarding no
    AllowTcpForwarding no
```

   iii. Save and close the file.

3. Remove the jail.

```
rm -rf /var/jail
```

4. Remove the new user.

```
deluser --remove-home newuser
```

5. Remove the chrootjail group.

```
delgroup --only-if-empty chrootjail
```