UNIX: CHANGE-ROOT ENVIRONMENTS FOR WEB APPLICATIONS
with examples for HP-UX

by Mark Meierjohann

for SANS GIAC LevelOne Security Essentials
March 30, 2001

---

1 Introduction

One of the most critical parts of a Unix-system on the internet is the security of the services it provides. It is essential not to run any services that are not required. Services that do not have to be publicly accessible should be reduced to a limited IP-address-range on the host (using tcp-wrappers/inetd.sec) as well as on perimeter defense (e.g. firewalls). Publicly accessible services as `httpd`, `ftp`, `dns` and `smtp` should be closely monitored and, if possible, run in their own environment.

Web-applications are often large programs. Because of their size they most likely have some errors that can lead to possible exploits. In order to reduce the risk from these applications it is advisable to give each application its own `chroot`-environment. If a vulnerability is found and the system gets compromised the hacker has only got a very limited access to the system, ideally none. Recent security holes in the DNS service `bind` are not as critical if the DNS nameserver is installed in a chroot-environment - if no shell is installed in the chroot-filesystem, no root-shell can be opened from it. It is still advisable to patch software with known security problems as soon as possible, but the chances of getting hacked and the resulting damage are a lot smaller if the software is installed in a chroot-environment

The only tools available in the `chroot`-environment are the tools you copied into it!

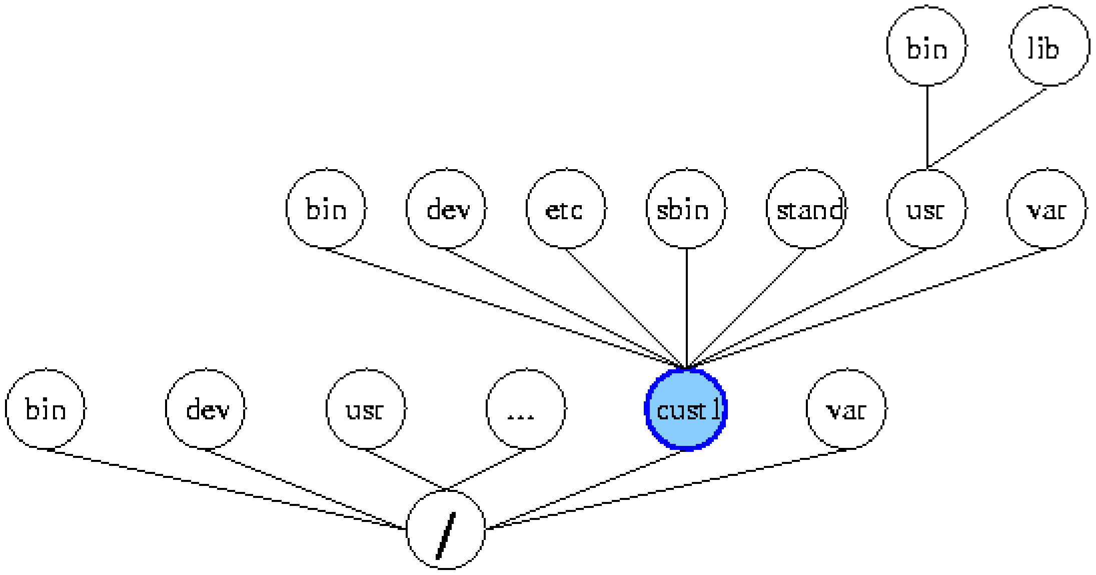
Often 3rd party applications are required by web-designers to provide them with their tools and interfaces to back-end databases. These application can create a vast amount of new security holes on the system and without a code review it is almost impossible to find all possible back-doors and other design errors in the application. Commercial software does not usually provide the source code for a code-review and the time constraints on developers do not usually allow the time for good testing in regards to security.

Bad CGI-bin/perl-scripts (see (5)) will not be able to disclose any of the real system information, if the web-application and perl as one of its tools are installed in a chroot-environment. Since communication with back-end processes usually use `ipc`(inter process communication) the application can still talk to its back-end (e.g. database clients, etc ...) even if the software runs in its own environment.

---

2 The chroot-Command

The `chroot` command allows a system to run services in a subdirectory of its filesystem-tree. After installing a *virtual* environment for the software to live in, it is possible to start specific services in their own *little world* with no possibility to see the underlying Unix system (see Figure 1). All tools and applications available in the chroot-environment are the tools you copied into the chroot-environment. Ideally this is a shell, maybe `ls` and `cd` for troubleshooting and possibly `cat, grep, ps & kill` for shutting down the application. Some more complex applications require a lot more tools to function (e.g. the postfix mta). Once the application is installed and running it is advisable to remove any tools, shells and libraries that are not required.



2.1 Files and Devices

The new environment needs a lot of the dynamically linked libraries (`/usr/lib/lib`) for the software to run in the new chroot-environment. On Linux and Solaris systems the `ldd`-command can be used to obtain a list of the required libraries an application needs to function. Other UNIX's (e.g. HP-UX) do not provide this possibility. Here a trial and error method should be used to create an environment with the least possible privileges.

In addition to the libraries all the programs needed by start-up and shutdown scripts have to be installed (e.g. cat, awk, sed,), if it is not possible to start the application from scripts in the real root filesystem.

There are a few character devices (`/dev/kmem, /dev/null, /dev/tcp`) that have to be created with `mknod`. Make sure to use the same *major* and *minor* numbers as the devices in the original filesystem have got.

2.2 syslog

For logging purposes a pipe called `/dev/log` has to be created in the chroot environment: `/sbin/mknod /$CHROOT/dev/log p`.

Most services use syslog for their default messages and errors. We do not want to loose these by running the software in a chroot-environment. `syslogd` only requires its configuration file `/etc/syslog.conf` and the pipe `/dev/log` for operation.

It is advisable to run one `syslogd`-process per chroot-environment. A modified start-up script `/sbin/init.d/syslogd` will start the required syslog-daemons. The config-file listing the chroot environments is `/etc/rc.config.d/syslog`.

Example syslog setup for HP-UX:

```
CHROOT=app-name
cp /usr/sbin/syslogd     /$CHROOT/usr/sbin/syslogd
cp /etc/syslog.conf      /$CHROOT/etc/syslog.conf
```

/etc/rc.config.d/syslog

```
# This files gives syslog a list of chroot-subsystems:
```

```
#
SYSLOG_CHROOT_SUBSYS="postfix bind"
```

/sbin/init.d/syslogd

```
[...]
'start')
  [..]
  if [ -f /etc/rc.config.d/syslog ]; then
    . /etc/rc.config.d/syslog
  fi
  for subsys in $SYSLOG_CHROOT_SUBSYS
  do
    echo "Starting syslogd in /$subsys..."
    # The -N option stops the syslogd from listening
    # to remote syslogds
    chroot /$subsys /$subsys/usr/sbin/syslogd -DN
    STATUS=$?
    if [ $STATUS -ne 0 ]

    then
      echo "syslogd in $subsys could not be started"
      echo "continuing "
      rval=2
    fi
  done
[...]
```

3 Applications

3.1 Apache - httpd

For apache a *web*-user is needed this is typically the user *nobody* since he is not allowed to do anything anyway (be aware that the user nobody does not work properly in HP-UX, here a user www:www seems to do the trick, but give the user a uid above 100). If a *real* user should be installed for ssh-login, just copy his line from the `/etc/passwd`. possibly a `/etc/group` file with the appropriate groups is needed, too. It is even safer, if the user has no password, but can log in using his *ssh-passphrase*.

```
   www:*:1030:500::/:
   custy:*:4326:501:Our happy customer:/home/custy:/bin/sh
```

The nsswitch.conf has to be created with at least the following entries:

```
   passwd: files
   shadow: files
   group:  files
   hosts:  files
```

Apache does not log to syslog - no syslogd is required for this application.

Now a modified start-up/shut-down script in the real root system that calls `chroot` and the application is all that is needed to start your web-server in the new chroot-environment.

3.2 DNS - bind

`bind` (Berkeley Internet Name Domain) the "dns server" is an other vulnerable service we do not want to provide in the *real* environment. `bind 8` does not require any extra files apart from runtime-libraries and the devices listed above. An swinstall (HP-UX) of the package into the chroot environment and moving the configuration files and directories to their new location in the chroot-environment is all that you have to do to install it.

A syslog-daemon should be installed into this chroot-environment for monitoring the application. `bind`'s logfiles to syslog can be of a lot of help when troubleshooting misconfigurations.

3.3 postfix - mail server/mail proxy

`postfix` (easy to configure and fast sendmail replacement) should be run in an change-root environment and has to be configured with care to prevent break-ins and being used as a spam mail relay.

For more information on postfix: www.postfix.org

Steps to install postfix:

1. shutdown sendmail

2. startup swagentd
   `/sbin/rc2.d/S100swagentd start`

3. create chroot environment

4. add the following required tools and configs:

   ```
   /usr/bin/id       /usr/bin/false
   /usr/bin/mailq    /usr/bin/newaliases
   /usr/bin/find     /usr/bin/grep
   /usr/bin/egrep    /usr/bin/sed

   /sbin/ls          /sbin/sh
   /sbin/cat         /sbin/false
   /sbin/mkdir       /sbin/chmod
   /sbin/chown

   /etc/services       /etc/passwd
   /etc/group          /etc/pam.conf
   /etc/pam_user.conf  /etc/protocols
   /etc/TIMEZONE       /etc/syslog.conf
   /etc/PATH           /etc/nsswitch.conf
   /etc/resolv.conf    /etc/hosts
   ```

   where `/etc/group` contains
   `mail::6:root`
   and `/etc/passwd` contains
   `root:*:0:3::/root:/sbin/sh`
   `nobody:*:123:20::/tmp:/bin/false`
   `postfix:*:124:6::/var/tmp:/bin/false`
   The other files are copies from the original system - maybe reduced to essential services.

5. Install postfix depot into chroot environment

6. configure the following lines in `/etc/postfix/main.cf` to match system:
   Example:

   ```
   myhostname = www.marki.de
   mydomain = marki.de
   mydestination = $myhostname, localhost.$mydomain, ¥
    $mydomain, mail.$mydomain, www.$mydomain, ¥
    ftp.$mydomain
   mynetworks = 192.xxx.yyy.0/24, 127.0.0.0/8
   ```

7. startup postfix with
   `chroot /usr/sbin/postfix start`

8. stop swagentd `/sbin/init.d/swagentdk start`

Now test your newly installed mail-server by monitoring the log file in `/postfix/var/adm/syslog/mail.log` and sending a few mails to the system:

```
telnet localhost 25
helo microsoft.com
```

```
mail from: bill@microsoft.com
rcpt to: me@spammail.de
data
This mail should work!!!
.
quit
```

To check if it relay's spam mails, just do the same thing from any system outside your network, it should fail:

```
telnet www.marki.de 25
helo microsoft.com
mail from: bill@microsoft.com
rcpt to: me@spammail.de
data
This mail should not be allowed!
.
quit
```

The log-file in your `chroot`-environment will give you details on what is happening.

To really verify if your system allows spam-mails testing most known spam methods just check `www.mail-abuse.org` or telnet to `mail-abuse.org` from the system you want to test.

3.4 Other Applications

Most other application can be installed in a similar way. Depending on how many dynamically linked libraries are required to run the application it can take longer and require more libraries to install it. If you are compiling an application yourself, think about linking the libraries statically, it makes the installation easier.

When patching your real system remember to copy updated files to your chroot environment, too. Tripwire is a great help to determine which files have been updated.

Be aware that any changes your application can do to your system in the change-root environment a hacker can do, too, after breaking into it.

4 Software Installation & Usage with HP-UX

To install software in the chroot-environment swinstall has to be called with the `@ /<chroot-directory>/` option. This specifies that the new environment is treated as the root for the package to be installed in.

The following script will making installations in chroot-environments easier:

```
#!/sbin/sh
SOURCE=$1
DESTINATION=$2
if [  $SOURCE ] && [ $DESTINATION ]
then
        swinstall -s $SOURCE "*" @ $DESTINATION
else
        echo "Usage : $0 <source depot> <destination dir>"
fi
```

To start a service in a chroot environment just run:

```
  # chroot /<chroot-directory>/ ¥
       /<path to program>/<program> <options>
```

The software will log to its default log-files, relative to the new root.

5 Advantages

Using a chroot-environment allows you to reduce access for services and users to a small environment i.e. only the chroot-filesystem. If a software bug allows to open a root shell (well known `ftp` & `sendmail` bugs) this shell will only have access to the chroot environment. Hackers will not ever see the *real* system. This should allow a system-administrator to sleep a little better at night.

---

6 Bibliography

(1)
Installing BIND v8 chroot'ed on Solaris 7 *by Seán Boran* 26 June, 2000
http://archives.neohapsis.com/archives/sf/sun/2000-q2/att-0272/01-chrooting_bind.html

(2)
Securing DNS in a chroot() environment *by Craig H. Rowland and Psionic Software Systems* 24 March 1999
http://www.guides.sk/psionic/dns/

(3)
Solaris2/FAQ Version: 1.72 *by Casper Dik* 06 February 2001
http://www.science.uva.nl/pub/solaris/solaris2.html

(4)
How to 'chroot' an Apache tree with Linux and Solaris *by Denice Deatrich* 26 Feb 2001
http://penguin.epfl.ch/chroot.html

(5)
Safer CGI Scripting *by Charles Walker and Larry Bennett* February 2001 `SysAdmin Magazine`
http://www.sysadminmag.com/articles/2001/0102/0102a/0102a.htm

---

Mark Meierjohann 2001-03-30