



Chroot Unix Services

Compass Security

April 5, 2004

Document Name:	Chrooting_Applications_CSNC_V2.0.doc
Version:	V 2.0
Authors:	Ivan Buetler, Compass Security AG Cyrill Brunschwiler, Compass Security AG Mathias Kengelbacher, Compass Security AG
Date of Delivery:	April 5, 2004
Document Status:	PUBLIC

GLÄRNISCHSTR. 7
POSTFACH 1671
CH-8640 RAPPERSWIL

Tel. +41 55-214 41 60
Fax +41 55-214 41 61
info@csnc.ch www.csnc.ch

Table of Contents

1 TECHNICAL SUMMARY	1
1.1 Introduction	1
1.2 To the Reader	1
1.3 Hardening	2
1.4 Definition	3
1.5 General Procedure	3
1.6 Chroot Templates	4
1.7 Privileges (uid=0)	4
1.8 Known Problems	5
1.9 Solaris	5
1.10 Linux	5
1.11 Debugging	6
1.12 Ldd	8
2 STEP BY STEP CHROOTING.....	9
2.1 Chrooting Apache Webserver (Solaris)	9
2.2 Chrooting Apache Webserver (Linux)	10
2.3 Chrooting MySQL (Solaris)	11
2.4 Chrooting MySQL (Linux)	12
2.5 Chrooting Tomcat (Solaris)	13
2.6 Chrooting Tomcat (Linux)	17
3 EXAMPLE SCRIPTS	19
3.1 Apache Webserver Chrooting Script (Solaris)	19
3.2 Apache Webserver Chrooting Script (Linux)	21
3.3 MySQL Chrooting Script (Solaris)	22
3.4 MySQL Chrooting Script (Linux)	24
3.5 Tomcat Chrooting Script (Solaris)	26
3.6 Tomcat Chrooting Script (Linux)	29
3.7 Example Script Solving Shared Libraries Problem	32
4 APPENDIX.....	34
4.1 Solaris Apache HTTPD Debugging Example	34
4.2 Solaris MySQL Debugging Example	36

1 Technical Summary

1.1 Introduction

This article introduces appropriate steps when chrooting a Unix service in its jail. Chrooting is one of the armoring tasks of the Unix hardening procedure. It creates additional borders in case of zero day threats, where working hacking exploits could compromise the vulnerable system. Unfortunately, the initial steps of chrooting a specific service could lead into a time-consuming procedure. This guide shall help understanding the chroot concept and giving motivation in really applying the chroot tasks for your public available services, such as Apache Web server.

This guide assumes the reader is using self-compiled versions of their running Internet services, instead of using prepared rpm's or Unix packages.

It is advised creating an adequate chroot'ing script during the initial setup and engineering phase. This will speed up the time when Unix administrators are patching and jailing new versions of their services. We expect Unix administrators not chrooting services after a patching night, if this is not easy applicable. This paper offers template chrooting scripts for Linux and Solaris operating systems. Please use the provided template scripts with care and really read through the different sections. Feedback is welcomed and we are happy serving updated versions in the near future.

This document contains chrooting guides and templates for the following applications and operating systems:

Application	Solaris	Linux
Apache 2.X	X	X
MySQL	X	X
Tomcat	X	X

1.2 To the Reader

This document is geared towards IT staff, Unix personnel, and other individuals concerned about the security issues of Unix security and Unix hardening.

1.3 Hardening

The following section briefly introduces common hardening tasks. The list is not very detailed, but shall give the context for the chrooting task. If you are really interested in comprehensive hardening guides, please google the web.

Against network attacks

- Disable all unused services.
- Restrict administrative services to trusted addresses, and use encrypted protocols like ssh.
- Turn off ICMP features
- Configure services not to advertise their type and version in their banner, headers or in response to queries like help, syst or version.bind.
- Install a firewall
- **Chrooting of the offered services**

Against Denial-of-Service attacks

- Limit number of connections per IP address per second, also ICMP replies, etc.
- Limit number of processes per user, memory and CPU time per process, etc.

Against malicious local users

- This may be an attacker who has already hacked a non-root service.
- Remove/change setuid/gid bit off binaries e.g. ping, making them usable only by root.
- Restrict access to /proc and other sensitive locations to root.
- Turn off users' access to cron, sendmail, etc.
- Disable ptrace or other powerful debug functions

Against legitimate users' clumsiness

- Use a password checker to stop people using weak passwords.
- Don't let administrators use the same password for admin services as they do for unencrypted things like POP3.
- Make users home directories root owned, with permissions 770, so the user can still write to their home directory, but they cannot accidentally change the permissions to let others in.

Against software bugs

- Use library functions to escape HTML, SQL, etc.
- One tmp directory per user, instead of a system-wide one.
- Input validation
- Proper authentication and authorization

Against a hacker who has broken one service

- Don't run services as root.
- **Consider running services inside a chroot.**

Against an attacker who has gained root

- secure level / capabilities
- Use chflags to make logs append-only, or use a central log server.
- mount file systems read-only where possible

1.4 Definition

DESCRIPTION (from manual page)

The `chroot()` and `fcchroot()` functions cause a directory to become the `rootdirectory`, the starting point for path searches for path names beginning with `/` (slash). The user's working directory is unaffected by the `chroot()` and `fcchroot()` functions.

The effective user ID of the process must be `super-user` to change the `rootdirectory`. While it is always possible to change to the system root using the `fcchroot()` function, it is not guaranteed to succeed in any other case, even should `fildev` be valid in all respects.

The `".."` entry in the root directory is interpreted to mean the root directory itself. Therefore, `".."` cannot be used to access files outside the subtree rooted at the root directory. Instead, `fcchroot()` can be used to reset the root to a directory that was opened before the root directory was changed.

1.5 General Procedure

The proposed procedure introduces the chrooting task and reflects the template chrooting script structure.

1. Compile the source. Using the prefix switch is mandatory. (`--prefix=/opt/applic/`)
2. Install the compiled version by "make install". This will install the compiled version to the prefix location
3. Start the service and analyze its functionality (non-chroot yet). Configure the service like you want the software to behave in production. This step is mandatory and very important!
4. Test the service and it's functionality (still not in the chroot)
5. Configure the chroot template script for your needs (Section 1 = Configuration). Especially configure the prefix (`prefix=/opt/applic/`) and jail location.
6. Execute the chrooting script. It will copy the service from the prefix location to the jail destination. The script enumerates binary dependencies and creates some of the chroot-required files to the jail directory.
7. Start the chroot'ed service in its jail. Analyze upcoming problems and improve the chrooting script by applying remediation to it. This will make your chroot script very powerful and helps speeding up chrooting the service three months later.
8. Loop at step 7 as long as you have a working chroot'ed service and a perfect chrooting script.
9. Configure the boot startup scripts of the chroot'ed service

The day vendor advisories appear giving information about high risks – remote exploitable vulnerabilities - follow the patching or update procedure at the non-chroot'ed prefix location. After proper testing take advantage of the prepared chroot script and jail the newest version it its final destination.

1.6 Chroot Templates

This section briefly introduces the chrooting template scripts. The scripts are all following the structure below

1. Configuration section
2. Remove “old” chroot files (not backup – these will be removed!)
3. Copy compiled binaries from prefix into jail location by keeping permissions, links special files etc.
4. Identification of static dependencies by using the “ldd” command. Copying the required shared objects to the chroot destination
5. Copying standard Unix files to the jail destination (/etc/passwd)
6. Creation of special files within the jail destination (/dev/null)
7. Manual section (copying files not already identified by step 4, 5 or 6). The step 7 of the above general procedure description (chapter 1.5) will disclose missing libraries, links or files.

KEEP IN MIND; THAT THE PROVIDED CHROOT-SCRIPT WILL DELETE THE OLD CHROOT-DIRECTORY CONTAINING ALL CONFIGURATION-, LOG- AND CONTENT-FILES! PLEASE CREATE A BACKUP OF YOUR CONFIGRATION FILES BEFORE APPLYING THE SCRIPT.

We use the *tar* command for copying the files from the prefix to the jail destination. In the way the tar command is used, it works for gnu and standard SUN tar.

1.7 Privileges (uid=0)

A chroot'ed service will not prevent the intruder from placing malicious code into the memory segment of the vulnerable service. A chroot jail is somehow a hacker-unfriendly place to be. Keep in mind even chroot'ed services are breakable by hackers. Unfortunately standard chroot does not offer non-root execution. Known attacks where escaping the jail are based on the fact, the chroot was initially started as root user.

Wietse Venema offers the open source tool chrootuid, which will give opportunity of chrooting without being root. The arrangement greatly reduces the impact of possible loopholes in network software. Wietse's tool “chrootuid” uses the passwd and group files from *outside* the chroot area.

Unfortunately, this guide describes usage of chrootuid tool for the template “Tomcat Solaris” only. All other descriptions are still not chrootuid aware.

1.8 Known Problems

The template chrooting script does not recursively resolve dynamic shared libraries of the target binary file. Especially step 7 of chapter 1.5 is time-consuming. Even your service starts after resolving all dependencies, later problems could occur. Compass learned from chrooting MySQL the nsswitch.conf file is not required for successful startup but once the user is hostbased authenticated, it becomes a required functionality.

However – the following scripts is a first draft in recursively identify object dependencies. Please use it with care. It is described in chapter 3.7.

1.9 Solaris

Earlier versions of SUN Solaris don't have a /dev/random and /dev/urandom device. Compass used the following information while solving this problem within the chroot.

<http://www.cosy.sbg.ac.at/~andi/SUNrand/>

```
Solaris /dev/random
A Solaris kernel module to emulate /dev/random and /dev/urandom known from Linux in Solaris.
Use at your own risk.
FYI: SUN now provides /dev/random too:

Solaris 9 (SunOS 5.9):
Included in the OS distribution (32Bit / 64Bit).

Solaris 8 (SunOS 5.8):
Patch 112438 (32Bit / 64Bit).

Solaris 8_x86 (SunOS 5.8_x86):
Patch 112439 (32Bit / 64Bit).

Solaris 7 / 7_x86 (SunOS 5.7 / 5.7_x86):
Install "SUNWski" package, available at the "Solaris Easy Access CDs" in the "Sun Webserver"
product.

Solaris 2.6 (SunOS 5.6):
Install "SUNWski" package, available at the "Solaris Easy Access CDs" in the "Sun Webserver"
product.

More information is available at http://sunsolve.Sun.COM/. Please search for document
"27606": "Differing /dev/random support requirements within Solaris [TM] Operating
Environments".
```

1.10 Linux

A Bug in the J2SDK 1.4.1_03 from Sun Microsystems for Linux requires to mount the proc file system in the chroot environment. This is normally only necessary, if the J2SDK wants to use the multi processor functionality of Linux. This should be fixed in further releases.

1.11 Debugging

STRACE (Linux only)

```

NAME
    strace - trace system calls and signals

SYNOPSIS
    strace [ -dffhiqrsttvVxx ] [ -acolumn ] [ -eexpr ] ... [
    -ofile ] [ -ppid ] ... [ -sstrsize ] [ -username ] [
    command [ arg ... ] ]

    strace -c [ -eexpr ] ... [ -Ooverhead ] [ -Ssortby ] [
    command [ arg ... ] ]

DESCRIPTION
    In the simplest case strace runs the specified command
    until it exits. It intercepts and records the system
    calls which are called by a process and the signals which
    are received by a process. The name of each system call,
    its arguments and its return value are printed on standard
    error or to the file specified with the -o option.

    strace is a useful diagnostic, instructional, and debug-
    ging tool. System administrators, diagnosticians and trou-
    ble-shooters will find it invaluable for solving problems
    with programs for which the source is not readily avail-
    able since they do not need to be recompiled in order to
    trace them.

usage: strace [-dffhiqrsttvVxx] [-a column] [-e expr] ... [-o file]
        [-p pid] ... [-s strsize] [-u username] [command [arg ...]]
    or: strace -c [-e expr] ... [-O overhead] [-S sortby] [command [arg ...]]
-c -- count time, calls, and errors for each syscall and report summary
-f -- follow forks, -ff -- with output into separate files
-F -- attempt to follow vforks, -h -- print help message
-i -- print instruction pointer at time of syscall
-q -- suppress messages about attaching, detaching, etc.
-r -- print relative timestamp, -t -- absolute timestamp, -tt -- with usecs
-T -- print time spent in each syscall, -V -- print version
-v -- verbose mode: print unabbreviated argv, stat, termio[s], etc. args
-x -- print non-ascii strings in hex, -xx -- print all strings in hex
-a column -- alignment COLUMN for printing syscall results (default 40)
-e expr -- a qualifying expression: option=[!]all or option=[!]vall[,val2]...
    options: trace, abbrev, verbose, raw, signal, read, or write
-o file -- send trace output to FILE instead of stderr
-O overhead -- set overhead for tracing syscalls to OVERHEAD usecs
-p pid -- trace process with process id PID, may be repeated
-s strsize -- limit length of print strings to STRSIZE chars (default 32)
-S sortby -- sort syscall counts by: time, calls, name, nothing (default time)
-u username -- run command as username handling setuid and/or setgid

```

```

Example:
amibi:~/ivan # strace -e open,close ls
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
close(3) = 0
open("/lib/libc.so.6", O_RDONLY) = 3
close(3) = 0
open("/dev/null", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = -1 ENOTDIR (Not a directory)
open(".", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = 3

```

```
close(3) = 0
superfile
close(1) = 0
```

TRUSS (Solaris)

NAME

truss - trace system calls and signals

SYNOPSIS

```
truss [ -fcaieldD ] [ - [ tTvx ] [ ! ] syscall , ... ]
[ - [ sS ] [ ! ] signal , ... ] [ - [ mM ] [ ! ]
fault , ... ] [ - [ rw ] [ ! ] fd , ... ] [ -
[ uU ] [ ! ] lib , ... : [ : ] [ ! ] func , ... ]
[ -o outfile ] command | -p pid ...
```

DESCRIPTION

The truss utility executes the specified command and produces a trace of the system calls it performs, the signals it receives, and the machine faults it incurs. Each line of the trace output reports either the fault or signal name or the system call name with its arguments and return value(s). System call arguments are displayed symbolically when possible using defines from relevant system headers; for any path name pointer argument, the pointed-to string is displayed. Error returns are reported using the error code names described in intro(3).

Optionally (see the -u option), truss will also produce an entry/exit trace of user-level function calls executed by the traced process, indented to indicate nesting.

tutorialba:/ # truss

```
usage: truss [-fcaieldD] [-[tTvx] [!]syscalls] [-[sS] [!]signals] \
[-[mM] [!]faults] [-[rw] [!]fds] [-[uU] [!]libs:[:][!]functs] \
[-o outfile] command | -p pid ...
```

Example:

```
tutorialba:/ # truss -topen,close ls
open("/var/ld/ld.config", O_RDONLY) = 3
close(3) = 0
open("/usr/lib/libc.so.1", O_RDONLY) = 3
close(3) = 0
open("/usr/lib/libdl.so.1", O_RDONLY) = 3
close(3) = 0
open("/usr/platform/SUNW,UltraAX-i2/lib/libc_psr.so.1", O_RDONLY) = 3
close(3) = 0
open64(".", O_RDONLY|O_NDELAY) = 3
close(3) = 0
backup      dev      kernel      merlin2     platform    tmp
bin         devices  lib         mnt         proc        usr
cdrom       etc      lost+found  net         root        var
core        export   mail        opt         sbin        vol
cornelia    home     mbox        pen,close   scripts     xfn
```

1.12 Ldd

Linux

<p>NAME</p> <p>ldd - print shared library dependencies</p> <p>SYNOPSIS</p> <p>ldd [-vVdr] program library ...</p> <p>DESCRIPTION</p> <p>ldd prints the shared libraries required by each program or shared library specified on the command line. If a shared library name does not contain a '/', ldd attempts to locate the library in the standard locations. To run ldd on a shared library in the current directory, a "." must be prepended to it's name.</p>
--

Solaris

<p>NAME</p> <p>ldd - list dynamic dependencies of executable files or shared objects</p> <p>SYNOPSIS</p> <p>ldd [-d -r] [-c] [-f] [-i] [-L] [-l] [-s] [-v] filename ...</p> <p>DESCRIPTION</p> <p>The ldd utility lists the dynamic dependencies of executable files or shared objects. ldd uses the runtime linker, ld.so.1, to generate the diagnostics, since it takes the object being inspected and prepares it as it would in a running process. By default, ldd triggers the loading of any lazy dependencies.</p>
--

2 Step by Step Chrooting

2.1 Chrooting Apache Webserver (Solaris)

Compile Apache by using the prefix switch

```
$# ./configure --prefix=/httpd --exec-prefix=/httpd --enable-modules="core rewrite ssl auth mime mime_magic speling headers file_cache deflate cgi auth access" --disable-beos --disable-os2 --disable-netware --disable-winnnt --disable-userdir --enable-so --enable-autoindex --with-ssl=/usr/local/ssl --enable-ssl --enable-proxy
```

```
$# make
$# make install
```

Apache is now installed to the prefix location

Go and copy the appropriate chroot template script. Edit the first "config" section

```
#####
set chroot=/chroot/httpd
set working_apache_dir=/httpd
set apache_dirname=httpd
#####
```

Execution of the chroot script. This will chroot /httpd into /chroot/httpd

```
$# ./make_chroot_apache_solaris.sh
```

Try starting apache within its jail (test config). Use the httpd binary directly and not the provided apachectl script, because the shell is not available within the jail.

```
$# chroot /chroot/httpd /httpd/bin/httpd -t
```

In case of no other problems, start the service in it's jail. Otherwise analyze the errors and improve your chrooting script. Chapter 4 introduces some scenarios, where Compass debugged the apache process. After the above command succeeds, try really starting the apache daemon.

```
$# chroot /chroot/httpd /httpd/bin/httpd -k start
```

Setup the boot startup script. In our case, we have modified the existing apachectl script and reconfigured the HTTPD variable.

```
HTTPD='chroot /chroot/httpd /httpd/bin/httpd'
```

What does `ps -ef` look like, once the process runs in its jail?

```
$# ps -ef | grep httpd
wwwrun 23034 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23031 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23035 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23036 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23032 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
root    21005      1 0   Mar 13 ?        0:18 /httpd/bin/httpd -k start
wwwrun 23033 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23027 21005 0 12:08:27 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23030 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
```

2.2 Chrooting Apache Webserver (Linux)

Compile Apache by using the prefix switch

```
$# ./configure --prefix=/httpd --exec-prefix=/httpd --enable-modules="core rewrite ssl auth
mime mime_magic speling headers file_cache deflate cgi auth access" --disable-beos --
disable-os2 --disable-netware --disable-winnt --disable-userdir --enable-so --enable-
autoindex --with-ssl=/usr/local/ssl --enable-ssl --enable-proxy
```

```
$# make
$# make install
```

Apache is now installed to the prefix location

Go and copy the appropriate chroot template script. Edit the first "config" section

```
#####
set chroot=/chroot/httpd
set working_apache_dir=/httpd
set apache_dirname=httpd
#####
```

Execution of the chroot script. This will chroot /httpd into /chroot/httpd

```
$# ./make_chroot_apache_linux.sh
```

Try starting apache within its jail (test config). Use the httpd binary directly and not the provided apachectl script, because the shell is not available within the jail.

```
$# chroot /chroot/httpd /httpd/bin/httpd -t
```

In case of no other problems, start the service in it's jail. Otherwise analyze the errors and improve your chrooting script. Chapter 4 introduces some scenarios, where Compass debugged the apache process. After the above command succeeds, try really starting the apache daemon.

```
$# chroot /chroot/httpd /httpd/bin/httpd -k start
```

Setup the boot startup script. In our case, we have modified the existing apachectl script and reconfigured the HTTPD variable.

```
HTTPD='chroot /chroot/httpd /httpd/bin/httpd'
```

What does ps -ef look like, once the process runs in its jail?

```
$# ps -ef | grep httpd
wwwrun 23034 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23031 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23035 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23036 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23032 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
  root  21005      1 0   Mar 13 ?        0:18 /httpd/bin/httpd -k start
wwwrun 23033 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23027 21005 0 12:08:27 ?        0:00 /httpd/bin/httpd -k start
wwwrun 23030 21005 0 12:08:28 ?        0:00 /httpd/bin/httpd -k start
```

2.3 Chrooting MySQL (Solaris)

Compile MySQL by using the prefix switch

```
$# ./configure --prefix=/mysql --exec-prefix=/mysql --with-mysql-user=mysqlrun --without-bench
```

```
$# make
$# make install
```

MySQL is now installed to the prefix location

Go and copy the appropriate chroot template script. Edit the first "config" section

```
#####
set chroot=/opt2/applic/chroot/mysql
set working_mysql_dir=/mysql
#####
```

Execution of the chroot script. This will chroot /opt2/applic/chroot/mysql into /mysql

```
$# ./make_chroot_mysql_solaris.sh
```

In case of no other problems, start the service in it's jail. Otherwise analyze the errors and improve your chrooting script. Chapter 4 introduces debugging examples. After the above make_chroot command succeeds, try really starting the MySQL daemon.

```
$# chroot /opt2/applic/chroot/mysql /mysql/libexec/mysqld &
```

Setup the boot startup script. We use the following, simple startup script.

```
#!/bin/sh

case "$1" in
  start)
    echo "starting mysql server in chroot";
    chroot /opt2/applic/chroot/mysql /mysql/libexec/mysqld &
    ;;
  stop)
    echo "stopping mysql server";
    MYSQLHOSTNAME=`/bin/hostname`
    PID_FILE/opt2/applic/chroot/mysql/mysql/var/$MYSQLHOSTNAME.pid
    if test -f $PID_FILE
    then
      PID=`/bin/cat $PID_FILE`
      kill $PID > /dev/null 2> /dev/null
    fi
    ;;
  restart|reload)
    echo "restart mysql server";
    $0 stop
    $0 start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac
```

2.4 Chrooting MySQL (Linux)

Compile MySQL by using the prefix switch

```
$# ./configure --prefix=/mysql --exec-prefix=/mysql --with-mysql-user=mysqlrun --without-bench
```

```
$# make
$# make install
```

MySQL is now installed to the prefix location

Go and copy the appropriate chroot template script. Edit the first "config" section

```
#####
set chroot=/opt2/applic/chroot/mysql
set working_mysql_dir=/mysql
#####
```

Execution of the chroot script. This will chroot /opt2/applic/chroot/mysql into /mysql

```
$# ./make_chroot_mysql_linux.sh
```

In case of no other problems, start the service in it's jail. Otherwise analyze the errors and improve your chrooting script. Chapter 4 introduces debugging examples. After the above make_chroot command succeeds, try really starting the MySQL daemon.

```
$# chroot /opt2/applic/chroot/mysql /mysql/libexec/mysqld &
```

Setup the boot startup script. We use the following, simple startup script.

```
#!/bin/sh

case "$1" in
  start)
    echo "starting mysql server in chroot";
    chroot /opt2/applic/chroot/mysql /mysql/libexec/mysqld &
    ;;
  stop)
    echo "stopping mysql server";
    MYSQLHOSTNAME=`/bin/hostname`
    PID_FILE/opt2/applic/chroot/mysql/mysql/var/$MYSQLHOSTNAME.pid
    if test -f $PID_FILE
    then
      PID=`/bin/cat $PID_FILE`
      kill $PID > /dev/null 2> /dev/null
    fi
    ;;
  restart|reload)
    echo "restart mysql server";
    $0 stop
    $0 start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac
```

2.5 Chrooting Tomcat (Solaris)

The following description would not final include all steps for hardening a tomcat installation but chrooting a service such as tomcat is part of a complete hardening. Further would the description point out a way to improve a chroot jails security by running it as a low privileged user.

Prerequisites:

- Building the chroot requires a running java installation.
- You should also have created two users which belong to the same group (eg. www). One user will run the chroot (eg. wwwrun) and the other one will be set as fileowner (eg. wwwadm)
- Make sure tomcat starts without errors outside the chroot

The chroot script is able either to copy an already running tomcat installation or a recently untared tomcat. Make sure both scripts (the chroot script and the shared library copy script) reside in the same folder.

```
razzia:/jail/scripts # ls -la
total 22
drwxr-xr-x  2 root    other      512 Mar 14 18:06 .
drwxr-xr-x  5 root    other      512 Mar 14 18:06 ..
-rwxr-x---  1 root    other     2637 Mar 14 18:06 copy_shared_libs.sh
-rwxr-x---  1 root    other     6070 Mar 14 18:06 make_chroot_tomcat_solaris.sh
razzia:/jail/scripts #
```

Configure the config section of the script `make_chroot_tomcat_solaris.sh`

```
# PATH AND FILE SETTINGS
#-----
JAIL=/jail/tomcat
JAVA_HOME=/opt/applic/j2sdk1.4.2_03
CATALINA_HOME=/opt/applic/jakarta-tomcat-5.0.19
CATALINA_PID=$JAIL$CATALINA_HOME/logs/catalina.pid

# USER CONFIGURATION
#-----
CATALINA_RUNNING_USER=wwwrun

# DEBUG OPTIONS
#-----
JPDA_TRANSPORT=dt_socket
JPDA_PORT=55081

#####
# END OF CONFIGURATION SECTION
#####
```

Run the script from its residing folder as follows:

```
razzia:/ # cd /jail/scripts
razzia:/jail/scripts # ./make_chroot_tomcat_solaris.sh
```

Check the output of the script for any unusual errors. If the script was run successfully the output should be something like:

```
CHROOTING TOMCAT
=====
Shared lib copy script exists.
=====
Chroot dir /jail/tomcat created
=====
Copying Tomcat from /opt/applic/jakarta-tomcat-5.0.19 to /jail/tomcat/opt/applic/jakarta-
tomcat-5.0.19
```

```

Copying Java from /opt/applic/j2sdk1.4.2_03 to /jail/tomcat/opt/applic/j2sdk1.4.2_03
Cleaning up tomcat installation
Setting fileowner for tomcat installation
Setting permission for tomcat installation
=====
Detecting shared libraries
/jail/tomcat/opt/applic/j2sdk1.4.2_03/bin/java : resolving...
...

=====
Copying additional libraries
=====
Copying file for java dns support
Creating tmp folder and devices
=====
CHROOTING TOMCAT: done.

```

Lets have a look at the new chroot folder.

```

razzia:/jail/tomcat # ls -la
total 16
drwxr-xr-x  7 root   other    512 Mar 14 18:26 .
drwxr-xr-x  6 root   other    512 Mar 14 18:08 ..
drwxr-xr-x  2 root   other    512 Mar 14 18:09 dev
drwxr-xr-x  2 root   other    512 Mar 14 18:23 etc
drwxr-xr-x  3 root   other    512 Mar 14 18:08 opt
drwxrwxrwt  2 root   other    512 Mar 14 18:09 tmp
drwxr-xr-x  3 root   other    512 Mar 14 18:09 usr

```

Special considerations

- /dev: The only device required by java is the zero device.
- /etc: In fact we will run the chroot jail as low privileged user with the command chrootuid. We do not need having the files passwd and group installed in the folder because the tool will always use the files from the origin file system for security reasons. The only files you'll come across will be used from java to resolve host and domain names.
- /opt: This folder includes the java and tomcat installation. Having the parts installed in opt depends on your policy and could be different (such as /usr/local/) in other environments.
- /tmp: Empty temporary folder
- /usr: Holds a single folder called lib where the shared libraries reside.
- /lib: Does not exist because solaris will lookup shared libraries in /usr/lib.
- /bin: Does not exist because we will run tomcat from our own startup script other than the occasionally startup.sh resp. shutdown.sh. Having a custom start script improves security because we do not need a shell script interpreter inside the chroot jail.

To run a chroot jail as different user than root, we need the tool chrootuid (written from Wietse Venema, also known for its effort on the postfix mailserver project). The software is available from <ftp://ftp.porcupine.org/pub/security/> and is distributed with a BSD-style license. The syntax is slightly different to the one of the chroot command.

```

chroot /path/to/chroot /path/to/command
chrootuid /path/to/chroot username /path/to/command

```

```

eg.
chrootuid /jail/tomcat wwwrun /opt/applic/j2sdk1.4.2_03/bin/java -version

```

At this point most of the chrooting work is done. Further steps will describe how to check and debug the java installation and how to start tomcat without using the default tomcat start and stop scripts. Then let us make sure java runs without any problems in the chroot jail.

```

razzia:/ # chrootuid /jail/tomcat wwwrun /opt/applic/j2sdk1.4.2_03/bin/java -version
java version "1.4.2_03"

```

```
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_03-b02)
Java HotSpot(TM) Client VM (build 1.4.2_03-b02, mixed mode)
```

We further check whether domain name lookups work properly. Use the following piece of code to check for proper name resolution:

```
import java.net.*;

public class NameLookupTest {
    public static void main (String[] args) throws Exception {
        InetAddress addr = InetAddress.getByName(args[0]);
        System.out.println(args[0] + "/" + addr.getCanonicalHostName());
    }
}
```

Create a file called NameLookupTest.java in the chroot folder and paste the above snippet into. Use the java compiler into the chroot to compile the the source

```
chroot /jail/tomcat /opt/applic/j2sdk1.4.2_03/bin/javac NameLookupTest.java
```

If DNS works properly the tool will be able to resolve host and domain names. Use it as follows:

```
chroot /jail/tomcat /opt/applic/j2sdk1.4.2_03/bin/java NameLookupTest localhost
localhost/localhost
chroot /jail/tomcat /opt/applic/j2sdk1.4.2_03/bin/java NameLookupTest www.csnc.ch
www.csnc.ch/212.243.104.215
```

If the chroot environment passed all the tests, tomcat should run properly. Use the following script to start the service. The default start and stop scripts which are shipped with tomcat will not work because the shell script interpreter (sh) isn't present in the chroot jail.

```
#!/sbin/sh
#
# This script supports starting tomcat without having the startup scripts
# installed in $CATALINA_HOME/bin. It provides starting tomcat in a chroot
# environment where the shell should not be installed in
#
# date:    Mon 03/15/2004
# author:  Cyrill Brunschwiler - BRU - Compass Security AG
#####

# PATH AND FILE SETTINGS
#-----
JAIL=/jail/tomcat
JAVA_HOME=/opt/applic/j2sdk1.4.2_03
CATALINA_HOME=/opt/applic/jakarta-tomcat-5.0.19
CATALINA_PID=$JAIL$CATALINA_HOME/logs/catalina.pid

# USER CONFIGURATION
#-----
CATALINA_RUNNING_USER=wwwrun

# DEBUG OPTIONS
#-----
JPDA_TRANSPORT=dt_socket
JPDA_PORT=55081

#####
# END OF CONFIGURATION SECTION
#####

case "$1" in
'start')
    /usr/local/bin/chrootuid $JAIL $CATALINA_RUNNING_USER $JAVA_HOME/bin/java \
        -Djava.endorsed.dirs=$CATALINA_HOME/common/endorsed \
```

```

        -classpath
$JAVA_HOME/lib/tools.jar:$CATALINA_HOME/bin/bootstrap.jar:$CATALINA_HOME/bin/commons-
logging-api.jar \
        -Dcatalina.base=$CATALINA_HOME \
        -Dcatalina.home=$CATALINA_HOME \
        -Djava.io.tmpdir=/tmp \
        org.apache.catalina.startup.Bootstrap start \
        >> $JAIL$CATALINA_HOME/logs/catalina.out 2>&1 &
    echo $! >> $CATALINA_PID
    ;;

'debug')
    /usr/local/bin/chrootuid $JAIL $CATALINA_RUNNING_USER $JAVA_HOME/bin/java -server -
Xdebug -Xnoagent -Djava.compiler=NONE \
        -Xrunjdpw:transport=$JPDA_TRANSPORT,server=y,suspend=n,address=$JPDA_PORT \
        -Djava.endorsed.dirs=$CATALINA_HOME/common/endorsed \
        -classpath
$JAVA_HOME/lib/tools.jar:$CATALINA_HOME/bin/bootstrap.jar:$CATALINA_HOME/bin/commons-
logging-api.jar \
        -Dcatalina.base=$CATALINA_HOME \
        -Dcatalina.home=$CATALINA_HOME \
        -Djava.io.tmpdir=/tmp \
        org.apache.catalina.startup.Bootstrap start \
        >> $JAIL$CATALINA_HOME/logs/catalina.out 2>&1 &
    echo $! >> $CATALINA_PID
    ;;

'stop')
    cat $CATALINA_PID | xargs kill -TERM
    rm $CATALINA_PID
    ;;

'cmd')
    echo /usr/local/bin/chrootuid $JAIL $CATALINA_RUNNING_USER $JAVA_HOME/bin/java \
        -classpath
$JAVA_HOME/lib/tools.jar:$CATALINA_HOME/bin/bootstrap.jar:$CATALINA_HOME/bin/commons-
logging-api.jar \
        -Dcatalina.base=$CATALINA_HOME \
        -Dcatalina.home=$CATALINA_HOME \
        -Djava.io.tmpdir=/tmp \
        org.apache.catalina.startup.Bootstrap start
    ;;

*)
    echo
    echo "Usage: $0 (start|debug|stop|cmd)"
    echo "    start: start tomcat"
    echo "    debug: start tomcat with jpda support"
    echo "    stop: stop tomcat"
    echo "    cmd: get command line to start tomcat (eg. for truss)"
    echo
exit 1
    ;;
esac
exit 0

```

2.6 Chrooting Tomcat (Linux)

To chroot the Jakarta Tomcat you will have to install the J2SDK in the chroot directory.

Using the Linux JDK 1.4.1_03 you will also need to mount the proc-filesystem in the chroot due a bug in the JDK (should be fixed in the next release):

```
mount -tproc proc /opt/applic/chroot/tomcat/proc/
```

Please keep in mind, that the proc system has to be mounted before you start the tomcat server. If you want to startup tomcat at boot time, write a script, which mounts the proc system in the chroot and is executed before the Tomcat server is started or add a line in your */etc/fstab*.

Moreover I included the *bash shell*, *su* and *uname* in my chroot jail to simplify the startup of the server and to use the standard startup script with some slight modifications. The *su* tool requires the directory */etc/pam.d-*, */etc/passwd-* and */etc/group-file* in the chroot. The file */etc/services* is also needed.

The first step is to configure and run the server in the standard environment for testing purpose¹.

In the second step configure and run the chroot-script (see appendix). The script mounts automatically the proc system in the chroot directory.

```
$# ./make_chroot_tomcat_linux.sh
```

To startup the tomcat server, I added the following script in the chroot tomcat bin (*/opt/applic/chroot/tomcat/opt/tomcat/bin*) directory to change the user before the startup of the daemon:

```
#!/bin/bash
case "$1" in
  start)
    echo "starting upload tomcat in chroot"
    su tomcatuploadrun -s /bin/bash -c "/opt/tomcat/bin/catalina.sh"
  start"
    ;;
  stop)
    echo "stopping upload tomcat in chroot"
    su tomcatuploadrun -s /bin/bash -c "/opt/tomcat/bin/catalina.sh stop"
    ;;
  restart|reload)
    $0 stop
    $0 start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac
```

This script can be executed within an init script using the following command:

```
chroot /opt/applic/chroot/tomcat /opt/tomcat/bin/catalina_su.sh $1;
```

In the *catalina.sh* script, all operation-system specific checks can be disabled to simplify the startup and to use as less command line tools as possible.

¹ The configuration of MySQL is not described in this document

Execute the following command to start the daemon:

```
$# chroot /opt/applic/chroot/tomcat /opt/tomcat/bin/catalina_su.sh start
```

In case of an error, the daemon returns "file not found" or "library not found" errors. This indicates, that the SDK is still missing some libraries or configuration files in its chroot jail. If this occurs, check first, if there are any unresolved symbolic links in the chroot- or its subdirectories. If you find some, add a line in your chroot-script, which copies the missing files into the jail and rerun the script.

It is also possible; that there are some not directly related libraries or configuration-files missing. To locate these libraries you can use the truss tool.

```
$# truss -topen,close chroot /opt/applic/chroot/tomcat /opt/tomcat/bin/catalina_su.sh start
```

Check if the process is started properly and is running under the specified user:

```
$# ps axgf | grep java
tomcata 11333 0.4 23.3 229192 29732 ?        S    08:07   0:21 /opt/jdk/bin/java -
Djava.endorsed.dirs=/opt/tomcat/common/endorsed -classpath
/opt/jdk/lib/tools.jar:/opt/tomcat/bin/bootstrap.jar -Dcatalina.base=/opt/tomcat -
Dcatalina.home=/opt/tomcat -Djava.io.tmpdir=/opt/tomcat/temp
org.apache.catalina.startup.Bootstrap start
tomcata 11334 0.0 23.3 229192 29732 ?        S    08:07   0:00 /opt/jdk/bin/java -
Djava.endorsed.dirs=/opt/tomcat/common/endorsed -classpath
/opt/jdk/lib/tools.jar:/opt/tomcat/bin/bootstrap.jar -Dcatalina.base=/opt/tomcat -
Dcatalina.home=/opt/tomcat -Djava.io.tmpdir=/opt/tomcat/temp
org.apache.catalina.startup.Bootstrap start [...]
```

To stop the process execute the command:

```
$# chroot /opt/applic/chroot/tomcat /opt/tomcat/bin/catalina_su.sh stop
```

3 Example Scripts

3.1 Apache Webserver Chrooting Script (Solaris)

```

#!/bin/csh -f
#####
# Start config section
# Please edit only these configuration parameters
#####
set chroot=/opt/reverseproxy/chroot/httpd
set working_apache_dir=/httpd
set apache_dirname=httpd
#####
# End config section
#####
#####
# Create CHROOT DIR
#####
if ( -d $chroot ) then
    echo "====="
    echo "chroot dir $chroot already there"
    rm -r $chroot
    echo "APACHE chroot dir $chroot deleted"
    echo "====="
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "====="
else
    echo "====="
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "====="
endif
#####
# Copy Apache working directory into CHROOT space
#
#####
#cp -Rip $prefix/$working_apache_dir $chroot/$prefix
echo "====="
echo "Copying $working_apache_dir into $chroot"
(cd / ; tar -cf - $apache_dirname) | (cd $chroot; tar -xpf -)
echo "COPY FINISHED"
#####
# Identification of Shared Libraries and
# installation into chroot env
#####
set shared_objects=`ldd $chroot/$working_apache_dir/bin/httpd |awk '{print $3}' | grep -v
$working_apache_dir | sort|uniq | tr "\t" " " | sed -e 's#^/##'`
set i=1
while ( $i <= $#shared_objects )
    (cd /; tar -cf - $shared_objects[$i]) |
    (cd $chroot; tar -xpf -)@ i = $i + 1
end
set shared_objects1=`ldd $chroot/$working_apache_dir/bin/httpd |grep SUN | tr "\t" " " |
sed -e 's#^.*[ ]/##'`
set y=1
while ( $y <= $#shared_objects1 )
    (cd /; tar -cf - $shared_objects1[$y]) | (cd $chroot; tar -xpf -)
    @ y = $y + 1
end
#####
# Installation of standard chroot files in Solaris

```

```
#####
(cd /; tar -cf - etc/passwd) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/resolv.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/nsswitch.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/log) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/share/lib/zoneinfo/MET) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libsendfile.so.1) | (cd $chroot; tar -xpf -)
#####
# Creation of standard chroot files in Solaris
#####
mkdir $chroot/tmp
chmod 777 $chroot/tmp
chmod +t $chroot/tmp
mkdir $chroot/dev
mknod $chroot/dev/null c 13 2
mknod $chroot/dev/zero c 13 12
mknod $chroot/dev/tcp c 11 42
chmod 666 $chroot/dev/*
#####
# Creation of special files not seen using ldd, but using
# truss or strace while startup
#####
(cd /; tar -cf - usr/lib/ld.so.1) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libc.so.1) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/netconfig) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/nss_files.so.1) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - mysql-4.0.17/lib/mysql/*so*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libcrypt_i.so*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libgen.so*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/librt.so.1) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libm.so.1) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - mysql/lib/mysql/libmysqlclient.so*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libcrypt_i.so*) | (cd $chroot; tar -xpf -)
#####
# Copy of /dev/random and /dev/urandom into chroot
# Pls. remember Solaris 8.0 requires patch 112438-01
# to get a random device. The simple copy command below
# fullfills my needs to have a random and urandom device.
#####
if ( -f /kernel/drv/random ) then
    echo "======"
    echo "PATCH 112438-01 installed"
    cp /kernel/drv/random $chroot/dev/random
    cp /kernel/drv/random $chroot/dev/urandom
    echo "======"
else
    echo "======"
    echo "PATCH 112438-01 not installed"
    echo "Unable to create random & urandom device within chroot"
    echo "======"
endif
endif
```

3.2 Apache Webserver Chrooting Script (Linux)

```

#!/bin/csh -f
#####
# Start config section
# Please edit only these configuration parameters
#####
set chroot=/opt/applic/chroot/httpd
set working_apache_dir=/httpd
set apache_dirname=httpd
#####
# End config section
#####
# Create CHROOT DIR
#####
if ( -d $chroot ) then
    echo "=====
    echo "chroot dir $chroot already there"
    rm -r $chroot
    echo "Apache chroot dir $chroot deleted"
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
else
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
endif
#####
# Copy Apache working directory into CHROOT space
#####
echo "=====
echo "Copying $apache_dirname into $chroot"
(cd / ; tar -cf - $apache_dirname) | (cd $chroot; tar -xpf -)
#####
# Identification of Shared Libraries and
# installation into chroot env
#####
set shared_objects=`ldd $chroot/$working_apache_dir/bin/httpd |awk '{print $3}' | grep -v
$working_apache_dir | sort|uniq | tr "\t" " " | sed -e 's#^/##'`
set i=1
while ( $i <= $#shared_objects )
    (cd /; tar -cf - $shared_objects[$i]) | (cd $chroot; tar -xpf -)
    @ i = $i + 1
end
#####
# Installation of standard chroot files in Linux
#####
(cd /; tar -cf - etc/passwd) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/HOSTNAME) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/nsswitch.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/share/zoneinfo/MET) | (cd $chroot; tar -xpf -)
cp $chroot/usr/share/zoneinfo/MET $chroot/etc/localtime
#####
# Creation of standard chroot files in Linux
#####
mkdir $chroot/tmp
chmod 777 $chroot/tmp
chmod +t $chroot/tmp
mkdir $chroot/dev

```

```

mknod -m 666 $chroot/dev/null c 1 3
chmod 666 $chroot/dev/*
#####
# Creation of special files not seen using ldd, but using
# truss or strace while startup
#####
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/host.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/services) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/HOSTNAME) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libnss_compat.so.2) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/ld-2.3.2.so) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libstdc++.so.5.0.5) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libz.so.1*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libnss_files.so.2) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/local/ssl/lib/libssl.so.0.9.7) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/ld.so.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/ld.so.cache) | (cd $chroot; tar -xpf -)

```

3.3 MySQL Chrooting Script (Solaris)

```

#!/bin/csh -f
#####
# Start config section
# Please edit only these configuration parameters
#####
set chroot=/opt2/applic/chroot/mysql
set working_mysql_dir=/mysql
#####
# End config section
#####
#####
# Create CHROOT DIR
#####
if ( -d $chroot ) then
    echo "=====
    echo "chroot dir $chroot already there"
    rm -r $chroot
    echo "MySQL chroot dir $chroot deleted"
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
else
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
endif
#####
# Copy MySQL working directory into CHROOT space
#####
echo "=====
echo "Copying $working_mysql_dir into $chroot"
(cd $working_mysql_dir; /usr/local/bin/tar -cf - $working_mysql_dir) | (cd $chroot;
/usr/local/bin/tar -xpf -)
echo "COPY FINISHED"
#####
# Identification of Shared Libraries and
# installation into chroot env
#####
set shared_objects=`ldd $chroot/$working_mysql_dir/libexec/mysqld |awk '{print $3}' | grep -
v $working_mysql_dir | sort|uniq | sed -e '/^[ ]*$/'`

```

```

set i=1
while ( $i <= $#shared_objects )
    /usr/local/bin/tar -cf - $shared_objects[$i] | (cd $chroot; /usr/local/bin/tar -xpf
-)
    @ i = $i + 1
end
set shared_objects1=`ldd $chroot/$working_mysql_dir/libexec/mysqld |grep SUN |sed -e '/^[
]*$/d'`
set y=1
while ( $y <= $#shared_objects1 )
    /usr/local/bin/tar -cf - $shared_objects1[$y] | (cd $chroot; /usr/local/bin/tar -xpf
-)
    @ y = $y + 1
end
#####
# Installation of standard chroot files in Solaris
#####
/usr/local/bin/tar -cf - /etc/passwd | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /etc/group | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /etc/resolv.conf | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /etc/hosts | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /etc/nsswitch.conf | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /etc/log | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /usr/share/lib/zoneinfo/MET | (cd $chroot; /usr/local/bin/tar -xpf
-)
#####
# Creation of standard chroot files in Solaris
#####
mkdir $chroot/tmp
chmod 777 $chroot/tmp
chmod +t $chroot/tmp
mkdir $chroot/dev
mknod $chroot/dev/null c 13 2
mknod $chroot/dev/zero c 13 12
mknod $chroot/dev/tcp c 11 42
chmod 666 $chroot/dev/*
#####
# Creation of special files not seen using ldd, but using
# truss or strace while s/usr/local/bin/tartup
#####
/usr/local/bin/tar -cf - /usr/lib/ld.so.1 | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /usr/local/lib/libstdc++* | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /usr/lib/libm.so.1 | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /usr/lib/libc.so.1 | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /usr/lib/libdl.so.1 | (cd $chroot; /usr/local/bin/tar -xpf -)
/usr/local/bin/tar -cf - /usr/lib/nss_files.so.1 | (cd $chroot; /usr/local/bin/tar -xpf -)
#####
# Copy of /dev/random and /dev/urandom into chroot
# Pls. remember Solaris 8.0 requires patch 112438-01
# to get a random device. The simple copy command below
# fullfills my needs to have a random and urandom device.
#####
if ( -f /kernel/drv/random ) then
    echo "=====
    echo "PATCH 112438-01 installed"
    cp /kernel/drv/random $chroot/dev/random
    cp /kernel/drv/random $chroot/dev/urandom
    echo "=====
else
    echo "=====
    echo "PATCH 112438-01 not installed"
    echo "Unable to create random & urandom device within chroot"
    echo "=====
endif
endif

```

3.4 MySQL Chrooting Script (Linux)

```
#!/bin/csh -f
#####
# Start config section
# Please edit only these configuration parameters
#####
set chroot=/opt/applic/chroot/mysql
set working_mysql_dir=/mysql
set mysql_dirname=mysql
#####
# End config section
#####
# Create CHROOT DIR
#####
if ( -d $chroot ) then
    echo "=====
    echo "chroot dir $chroot already there"
    rm -r $chroot
    echo "MySQL chroot dir $chroot deleted"
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
else
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
endif
#####
# Copy mysql working directory into CHROOT space
#####
echo "=====
echo "Copying $mysql_dirname into $chroot"
(cd / ; tar -cf - $mysql_dirname) | (cd $chroot; tar -xpf -)
#####
# Identification of Shared Libraries and
# installation into chroot env
#####
set shared_objects=`ldd $chroot/$working_mysql_dir/libexec/mysqld |awk '{print $3}' | grep -
v $working_mysql_dir | sort|uniq | tr "\t" " " | sed -e 's#^/##'`
set i=1
while ( $i <= $#shared_objects )
    (cd /; tar -cf - $shared_objects[$i]) | (cd $chroot; tar -xpf -)
    @ i = $i + 1
end
#####
# Installation of standard chroot files in Linux
#####
(cd /; tar -cf - etc/passwd) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/HOSTNAME) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/nsswitch.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/share/zoneinfo/MET) | (cd $chroot; tar -xpf -)
cp $chroot/usr/share/zoneinfo/MET $chroot/etc/localtime
(cd /; tar -cf - bin/bash) | (cd $chroot; tar -xpf -)
cp $chroot/bin/bash $chroot/bin/sh
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/host.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
```

```
(cd /; tar -cf - etc/services) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/HOSTNAME) | (cd $chroot; tar -xpf -)
#####
# Creation of standard chroot files in Linux
#####
mkdir $chroot/tmp
chmod 777 $chroot/tmp
chmod +t $chroot/tmp
mkdir $chroot/dev
mknod -m 666 $chroot/dev/null c 1 3
chmod 666 $chroot/dev/*
#####
# Creation of special files not seen using ldd, but using
# truss or strace while startup
#####
(cd /; tar -cf - lib/libnss_compat.so.2) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/ld-2.3.2.so) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libstdc++.so.5.0.5) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libz.so.1*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libnss_files.so.2) | (cd $chroot; tar -xpf -)
```

3.5 Tomcat Chrooting Script (Solaris)

This script sets also the file- and user-permission on the files in the chroot. Find the script which deals with resolving shared libraries in a recursively manner below.

```

#!/bin/csh -f
#####
# TOMCAT CHROOT SCRIPT
#-----
# date:    Mon 03/15/2004
# author:  Cyrill Brunschwiler - BRU - Compass Security AG
#-----
# This script assumes a working java installation. Its further
# required having a running tomcat installation. Configure the
# paths to the running installations below in in the config
# section.
#
# Before chrooting the above tomcat directory, pls. make sure
# tomcat will start properly.
#
# If this basic test succeeds, you can use this script and
# change the current config settings to apply your needs.
#
# After you have applied the commands within the script you
# should adjust the params in the start/stop script for your
# needs.
#
# In any case the tomcat daemon will not start, please debug
# tomcat using truss
#####
#####
# CONFIGURATION SECTION (pls. apply your needs here)
#
# $chroot          (CHROOT DIR)
# $working_tomcat_dir (WORKING TOMCAT DIR itself)
#####

set chroot=/jail/tomcat

# set source of installations W/O LEADING SLASH
#-----
set working_tomcat_dir=opt/applic/jakarta-tomcat-5.0.19
set working_jdk_dir=opt/applic/j2sdk1.4.2_03

# set the tomcats user and group
#-----
set user=wwwadm
set group=www

# set only if you changed the name of the script manually
#-----
set copy_script=./copy_shared_libs.sh

#####
# END OF CONFIGURATION SECTION: no more changes need
#####

#####
# Create CHROOT DIR
#####
echo
echo "CHROOTING TOMCAT"
echo "====="

```

```

if (-e $copy_script) then
    echo "Shared lib copy script exists."
else
    echo "Shared lib copy script $copy_script does not exist. aborted"
    exit 1
endif
echo "====="

if ( -d $chroot ) then
    echo "Chroot dir $chroot already exists"
    rm -r $chroot
    echo "Chroot dir $chroot removed"
    mkdir $chroot
    echo "Chroot dir $chroot created"
    echo "====="
else
    echo "Chroot dir $chroot created"
    mkdir $chroot
    echo "====="
endif

#####
# Copy Tomcat & Java working directory into CHROOT space
#####
echo "Copying Tomcat from /$working_tomcat_dir to $chroot/$working_tomcat_dir"
(cd /; tar -cf - $working_tomcat_dir) | (cd $chroot; tar -xpf -)
echo "Copying Java from /$working_jdk_dir to $chroot/$working_jdk_dir"
(cd /; tar -cf - $working_jdk_dir) | (cd $chroot; tar -xpf -)

#####
# Set user, group for tomcat
#####
echo "Cleaning up tomcat installation"

# cleanup old config
#-----
if (-d $chroot/$working_tomcat_dir/conf/Catalina) then
    rm -r $chroot/$working_tomcat_dir/conf/Catalina
endif

# cleanup old work directory entries
#-----
if (-d $chroot/$working_tomcat_dir/work/Catalina) then
    rm -r $chroot/$working_tomcat_dir/work/Catalina
endif

# clear old logs
#-----
rm -r $chroot/$working_tomcat_dir/logs
mkdir $chroot/$working_tomcat_dir/logs

echo "Setting fileowner for tomcat installation"
chgrp -R $group $chroot/$working_tomcat_dir
chown -R $user $chroot/$working_tomcat_dir

echo "Setting permission for tomcat installation"
chmod 775 $chroot/$working_tomcat_dir/conf
chmod 775 $chroot/$working_tomcat_dir/work
chmod 775 $chroot/$working_tomcat_dir/temp
chmod 775 $chroot/$working_tomcat_dir/logs
chmod 640 $chroot/$working_tomcat_dir/conf/*
echo "====="

#####
# Identification of Shared Libraries and installation into the
# chroot environment
#####

```

```
echo "Detecting shared libraries"
$copy_script $chroot/$working_jdk_dir/bin/java $chroot
$copy_script $chroot/$working_jdk_dir/jre/lib/sparc/libjava.so $chroot
$copy_script $chroot/$working_jdk_dir/jre/lib/sparc/client/libjvm.so $chroot
echo "====="

#####
# Copying additional libraries
#####
echo "Copying additional libraries"
(cd /; tar -cf - usr/lib/ld.so.1)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/libresolv.so.2)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/nss_compat.so.1)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/nss_dns.so.1)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/lib/nss_files.so.1)|(cd $chroot; tar -xpf -)
echo "====="

#####
# Copying special files
#####
echo "Copying file for dns support"
(cd /; tar -cf - etc/netconfig)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/resolv.conf)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/nsswitch.conf)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts)|(cd $chroot; tar -xpf -)

#####
# Creation of temp folder and devices
#####
echo "Creating tmp folder and devices"
mkdir $chroot/tmp
chmod 777 $chroot/tmp
chmod +t $chroot/tmp

mkdir $chroot/dev
mknod $chroot/dev/zero c 13 12
chmod 666 $chroot/dev/*
echo "====="

#####
# Finished
#####
echo "CHROOTING TOMCAT: done."
echo
```

3.6 Tomcat Chrooting Script (Linux)

This script sets also the file- and user-permission on the files in the chroot.

```

#!/bin/csh -f
#####
# Start config section
# Please edit only these configuration parameters
#####
set chroot=/opt/applic/chroot/tomcat-admin
set working_tomcat_dir=/opt/tomcat
set working_jdk_dir=/opt/j2sdk1.4.2_03
set tomcat_dirname=opt/tomcat
set jdk_dirname=opt/j2sdk1.4.2_03
set webappconfigdir=/opt/applic/webappconfig
set configfile=opt/tomcat/conf/server.xml
set startupfile=opt/tomcat/bin/catalina_su.sh
set webappconfigname=server_admin.xml
set webappstartupname=catalina_su_admin.sh
set webappdir=/opt/applic/webapps
set webappname=uploadadmin.war
set tomcatrunuser=tomcatadminrun
set tomcatadminuser=tomcatadminadm
set tomcatgroup=tomcatadmin
#####
# End config section
#####
# Create CHROOT DIR
#####
if ( -d $chroot ) then
    echo "=====
    echo "chroot dir $chroot already there"
    echo "unmount /proc"
    umount $chroot/proc
    rm -r $chroot
    echo "Tomcat chroot dir $chroot deleted"
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
else
    echo "=====
    echo "make empty CHROOT directory $chroot"
    mkdir $chroot
    echo "=====
endif
#####
# Copy Apache working directory into CHROOT space
#####
echo "=====
echo "Copying $jdk_dirname into $chroot"
(cd / ; tar -cf - $jdk_dirname) | (cd $chroot; tar -xpf -)
echo "COPY FINISHED -- set up link"
(cd $chroot; ln -s $working_jdk_dir opt/jdk)
#####
# Copy tomcat working directory into CHROOT space
#####
echo "=====
echo "Copying $tomcat_dirname into $chroot"
(cd / ; tar -cf - $tomcat_dirname) | (cd $chroot; tar -xpf -)
#####
# Identification of Shared Libraries and
# installation into chroot env
#####

```

```

set shared_objects=`ldd $chroot/$working_jdk_dir/bin/java |awk '{print $3}' | grep -v
$working_jdk_dir | sort|uniq | tr "\t" " " | sed -e 's#^/##'`
set i=1
while ( $i <= $#shared_objects )
    (cd /; tar -cf - $shared_objects[$i]) | (cd $chroot; tar -xpf -)
    @ i = $i + 1
end
#####
# Installation of standard chroot files and tools in Linux
#####
(cd /; tar -cf - etc/passwd) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/nsswitch.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - usr/share/zoneinfo/MET) | (cd $chroot; tar -xpf -)
cp $chroot/usr/share/zoneinfo/MET $chroot/etc/localtime
(cd /; tar -cf - bin/bash) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - bin/uname) | (cd $chroot; tar -xpf -)
cp $chroot/bin/bash $chroot/bin/sh
(cd /; tar -cf - bin/su) | (cd $chroot; tar -xpf -)

set shared_objects=`ldd /bin/su |awk '{print $3}' | grep -v $working_jdk_dir | sort|uniq |
tr "\t" " " | sed -e 's#^/##'`
set i=1
while ( $i <= $#shared_objects )
    (cd /; tar -cf - $shared_objects[$i]) | (cd $chroot; tar -xpf -)
    @ i = $i + 1
end
(cd /; tar -cf - lib/libpam.so.0.77)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libpam_misc.so.0)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libpam_misc.so.0.77)|(cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libnss_compat.so.2) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/group) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/host.conf) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/hosts) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/services) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/HOSTNAME) | (cd $chroot; tar -xpf -)
mkdir $chroot/tmp
chmod 777 $chroot/tmp
chmod +t $chroot/tmp
mkdir $chroot/dev
mknod -m 666 $chroot/dev/null c 1 3
chmod 666 $chroot/dev/*
#####
# Creation of special files not seen using ldd, but using
# truss or strace while startup
#####
(cd /; tar -cf - lib/ld-linux.so.2) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/ld-2.3.2.so) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libnsl.so.1) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libm.so.6) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libreadline.so.4*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libhistory.so*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libncurses.so*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - etc/pam.*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/security/*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libxcrypt.so.1*) | (cd $chroot; tar -xpf -)
(cd /; tar -cf - lib/libnss_files.so.2) | (cd $chroot; tar -xpf -)
(cd $webappdir; tar -cf - $webappname) | (cd $chroot; tar -xpf -)
chmod 640 $chroot/$webappname
#####
# Copy of /dev/random and /dev/urandom into chroot
# Pls. remember Solaris 8.0 requires patch 112438-01
# to get a random device. The simple copy command below
# fullfills my needs to have a random and urandom device.
#####

```

```
mkdir $chroot/proc
mount -tproc proc $chroot/proc/
echo "== copy config file ($webappconfigdir/$webappconfigname to $chroot/$configfile)"
cp $webappconfigdir/$webappconfigname $chroot/$configfile
echo "== copy startup file ($webappconfigdir/$webappstartupname to $chroot/$startupfile)"
cp $webappconfigdir/$webappstartupname $chroot/$startupfile
echo "== setting userprivs"
echo "== setting group to $tomcatgroup"
(cd $chroot; chgrp -R $tomcatgroup $tomcat_dirname)
(cd $chroot; chgrp -R $tomcatgroup $webappname)
echo "== setting admin user to $tomcatadminuser"
(cd $chroot; chown -R $tomcatadminuser $tomcat_dirname)
(cd $chroot; chown -R $tomcatadminuser $webappname)
```

3.7 Example Script Solving Shared Libraries Problem

The above scripts are not able to recursively resolving shared libraries. This could lead into problems, while chrooting a new service. The following script tries giving a help recursively resolving the required shared libraries.

The following script has beta-status and is not widely used by Compass.

```
#!/bin/csh -f
#
# This script was written to improve chroot processes. It supports
# copying shared libraries in a recursively manner by detecting
# each libraries dependencies again.
# To avoid neverending recursion, the script would check whether the
# library was already copied to the destination chroot.
#
# date: Mon 03/15/2004
# author: Cyrill Brunschwiler - BRU - Compass Security AG
#####

# did we receive all params?
#####
if ($#argv != 2) then
    echo "Usage: $0 (path_to_binary|path_to_library) chroot_path"
    exit 1
endif

# get params from command line
#####
set library=$1
set chroot=$2

# check if chroot already exists otherwise leave script
#####
if (-d $chroot) then

    # check if file exists otherwise leave script
    #####
    if (-e $library) then

        # we're sure to have valid paths and files yet. so let us
        # check for the files dependencies
        #####
        echo "$library : resolving..."
        set subs=`ldd $library | awk '{print $3}' | grep -v $chroot | sort | uniq | tr "\t" "
" | sed -e 's#^/##'`

        # we hold all libraries in subs now. first check if already
        # a copy of the library exists. if not copy it and check it
        # for further dependencies by calling this script again.
        #####
        foreach subelement ($subs)
            echo "/$subelement : found"

            # check if already copied to chroot otherwise copy the
            # library and recurse
            #####
            if (-e $chroot/$subelement) then
                echo "/$subelement : already exists. skipped."
            else
                (cd /; tar -cf - $subelement) | (cd $chroot; tar -xpf -)
                echo "/$subelement : copied."
            endif
        endforeach
    endif
endif
```

```
        # now check if the copied library has any other
        # dependencies -> call this script again
        #####
        $0 "$subelement" $chroot
    endif
end
echo "$library : resolving... done."
exit 0
else
    echo "Error: library $library does not exist. aborted."
    exit 1
endif
else
    echo "Error: folder $chroot does not exist. aborted."
    exit 1
endif
```

4 Appendix

4.1 Solaris Apache HTTPD Debugging Example

This example assumes the default provided default chroot script for Solaris and apache does not work.

In case of an error, the daemon returns “file not found” or “library not found” errors:

```
# chroot /opt/applic/chroot/httpd/ /httpd/bin/httpd -k start
ld.so.1: /httpd/bin/httpd: fatal: libsocket.so.1: open failed: No such file or directory
Killed
```

This indicates, that Apache is still missing some libraries or configuration files in its chroot jail. If this occurs, check first, if there are any unresolved symbolic links in the chroot- or its subdirectories. If you find some or if the library is completely missing add a line in your chroot-script, which copies these files into the jail and rerun the script.

In our example we have to copy the library *libsocket.so.1* into the chroot dir. For this purpose we add the following line in our chroot script.

```
(cd /; tar -cf - /usr/lib/libsocket.so.1) | (cd $chroot; tar -xpf -)
```

It is also possible; that there are some not directly related libraries or configuration-files missing. To locate these libraries you can use the `truss` tool.

This tool will give you a lot of information what’s going on, while starting up the daemon. The error normally occurs some lines before the error message is printed out and can be identified by the `-1` or `Err#2` result of the `open` command.

If you find the missing library add a line in your chroot-script, which copies the missing file(s) and rerun the script.

For example:

```
#$ chroot /opt/applic/chroot/httpd /httpd/bin/httpd -k start
httpd: bad user name wwwwrun
```

Locate the missing library using `truss`:

```
#$ truss -topen chroot /opt/applic/chroot/httpd/ /httpd/bin/httpd -k start
open("/var/ld/ld.config", O_RDONLY) = 3
open("./libc.so.1", O_RDONLY) Err#2 ENOENT
open("/mysql/lib/libc.so.1", O_RDONLY) Err#2 ENOENT
open("/usr/local/lib/libc.so.1", O_RDONLY) Err#2 ENOENT
[...]
open("/etc/netconfig", O_RDONLY) = 3
open("/dev/udp", O_RDONLY) Err#2 ENOENT
open64("/etc/.name_service_door", O_RDONLY) Err#2 ENOENT
open("/etc/nsswitch.conf", O_RDONLY) = 5
open("./nss_files.so.1", O_RDONLY) Err#2 ENOENT
```

```
open("/usr/local/lib/nss_files.so.1", O_RDONLY) Err#2 ENOENT
open("/usr/local/BerkeleyDB.3.3/lib/nss_files.so.1", O_RDONLY) Err#2 ENOENT
open("/usr/lib/nss_files.so.1", O_RDONLY) Err#2 ENOENT
httpd: bad user name wwwrun
```

In our example the library *nss_files.so.1* is missing. On our system the file *nss_files.so.1* is located in the directory */usr/lib*. To solve the problem we add the following line in our chroot script and rerun the script:

```
(cd /; tar -cf - /usr/lib/nss_files.so.1) | (cd $chroot; tar -xpf -)
```

At the end, when everything works, as it should, delete all unused content from your configuration files in your chroot. For example all not required users from the */etc/passwd* file.

4.2 Solaris MySQL Debugging Example

In case of an error, the daemon returns “file not found” or “library not found” errors:

```
$# chroot /opt/applic/chroot/mysql/ /mysql/libexec/mysqld
ld.so.1: /mysql/libexec/mysqld: fatal: libsocket.so.1: open failed: No such file or
directory
```

This indicates, that MySQL is still missing some libraries or configuration files in its chroot jail. If this occurs, check first, if there are any unresolved symbolic links in the chroot- or its subdirectories. If you find some or if the library is completely missing add a line in your chroot-script, which copies these files like into the jail and rerun the script.

In our example we have to copy the library *libsocket.so.1* into the chroot dir. For this purpose we add the following line in our chroot script.

```
(cd /; tar -cf - /usr/lib/libsocket.so.1) | (cd $chroot; tar -xpf -)
```

It is also possible; that there are some not directly related libraries or configuration-files missing. To locate these libraries you can use the `truss` tool.

This tool will give you a lot of information what's going on, while starting up the daemon. The error normally occurs some lines before the error message is printed out and can be identified by the `-1` or `Err#2` result of the open command.

If you find the missing library add a line in your chroot-script, which copies the missing file(s) and rerun the script.

For example:

```
chroot /opt/applic/chroot/mysql/ /mysql/libexec/mysqld --user=mysqlrun
Fatal error: Can't change to run as user 'mysqlrun' ;
Please check that the user exists!
040405 13:59:48 Aborting
040405 13:59:48 /mysql/libexec/mysqld: Shutdown Complete
```

Locate the missing library using `truss`:

```
$# truss -topen chroot /opt/applic/chroot/mysql /mysql/libexec/mysqld -user=mysqlrun
open("/var/ld/ld.config", O_RDONLY) = 3
open("./libc.so.1", O_RDONLY) = 3
open("./libdl.so.1", O_RDONLY) = 3
open("/usr/platform/SUNW,UltraAX-i2/lib/libc_psr.so.1", O_RDONLY) = 3
[...]
open("/usr/share/lib/zoneinfo/MET", O_RDONLY) = 3
open64("/mysql/var/razzia.lower-test", O_RDWR|O_CREAT, 0666) = 3
open64("/mysql/share/mysql/charsets/Index", O_RDONLY) = 3
open64("/mysql/share/mysql/english/errmsg.sys", O_RDONLY) = 3
open("/etc/netconfig", O_RDONLY) Err#2 ENOENT
open("/etc/netconfig", O_RDONLY) Err#2 ENOENT
open64("/etc/.name_service_door", O_RDONLY) Err#2 ENOENT
open("/etc/nsswitch.conf", O_RDONLY) = 4
open("./nss_files.so.1", O_RDONLY) Err#2 ENOENT
open("/mysql/lib/nss_files.so.1", O_RDONLY) Err#2 ENOENT
open("/usr/local/lib/nss_files.so.1", O_RDONLY) Err#2 ENOENT
```

```
open("/usr/local/BerkeleyDB.3.3/lib/nss_files.so.1", O_RDONLY) Err#2 ENOENT
open("/usr/lib/nss_files.so.1", O_RDONLY)      Err#2 ENOENT
Fatal error: Can't change to run as user 'mysqldrun' ; Please check that the user exists!
040405 14:02:53 Aborting
040405 14:02:53 /mysql/libexec/mysqld: Shutdown Complete
```

In our example the library *nss_files.so.1* is missing. On our system the file *nss_files.so.1* is located in the directory */usr/lib*. To solve the problem we add the following line in our chroot script and rerun the script:

```
(cd /; tar -cf - /usr/lib/nss_files.so.1) | (cd $chroot; tar -xpf -)
```

At the end, when everything works, as it should, delete all unused content from your configuration files in your chroot. For example all not required users from the */etc/passwd* file.