;login:

THE MAGAZINE OF USENIX & SAGE

July 2001 • Volume 26 • Number 4

inside:

SETTING UP BIND8 IN A CHANGE-ROOTED ENVIRONMENT ON SOLARIS

by Timo Sivonen

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

setting up BIND8 in a change-rooted environment on solaris

by Timo Sivonen

Timo Sivonen is a consultant working for Greenwich Technology Partners. His interests include operating systems, security management and model railroads.

tljs@greenwichtech.com

Introduction

Probably the majority of all BIND installations in the world are configured to run named as super-user. This implies full access to the operating system, which probably is the most dangerous configuration possible. Any vulnerability in BIND, such as a buffer overflow bug, will put the entire underlying operating system at risk. However, there are two protection mechanisms in BIND8 to limit the consequences of a buffer overflow: running named in a change-rooted environment and changing the owner of the process to other than super-user. Although the measures in the following text will not eliminate the potential and real security vulnerabilities in BIND, the goal is to do what can be done to contain the damage caused by an attack.

Changing the root directory of a process to something other than the system root ('/') is a very effective method to limit the execution environment to the bare minimum. The rationale is to withdraw all vulnerable system tables and databases (e.g., /etc/passwd) from the reach of the service process. If the password database does not exist, it cannot be exploited. A change-rooted execution environment also permits file and directory permissions more rigorous than otherwise possible. Figure 1 shows the directory structure of our setup.

Simply moving the process to a change-rooted environment is not enough. Although only the super-user can execute the chroot() system call, the super-user can also cancel the effect of chroot() and break out from the change-rooted area. Simon Burr[1] has put together an excellent Web page on how to attack against chroot(). Consequently, we must follow the principle of least privilege and change the user owning named from root to, for example, an unprivileged bind account. Of course, there must not be any set-user-id executables in the change-rooted area.

Since BIND-8.1.2, one has been able to use a command-line option to instruct named to change-root itself immediately after it has started up, and another option to change the user-id the process is running on. Sun Microsystems has been distrib-

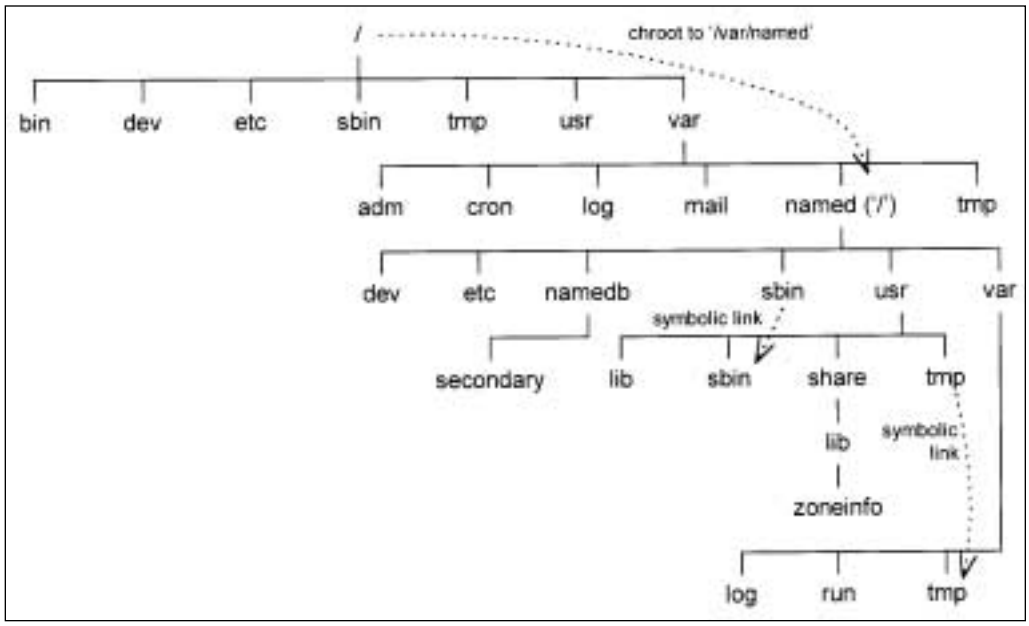

Figure 1. Directory hierarchy

uting BIND8—version 8.1.2—since Solaris 7, but sadly, Sun's version (in.named) has neither feature available. This is strikingly disturbing since change-root is never completely safe if the process is owned by root.

In the following text, we will take a generic approach and describe how to perform change-root first and then execute named in the restricted environment. This approach has two advantages:

- The process won't have any open file descriptors outside the change-rooted area.
- Since you do not rely on chroot() internal to a program, you can use the same method to change-root other unix services as well. There is also Wietse Venema's very useful chrootuid[2] utility that not only changes the root directory but also launches the process with the given user-id.

Obviously, there has been little improvement in system security if one runs BIND in a restricted environment yet the operating system has all the default network services such as telnet, FTP, XDMCP and others available to the world. SANS' Step-by-Step Guide[3] is a good source for instructions how to build a hardened Solaris system.

Getting Started

In the following text we will assume that you have either built BIND-8.2.3 from the source,[4] or you have obtained the binaries from Sunfreeware.com.[5] You can install the binaries and the configuration files to the default locations, since we will have to copy those to the change-rooted hierarchy.

The first step is to decide where to put our BIND hierarchy and create bind, the BIND account, in /etc/passwd. The BIND directory will be the home directory of the account.

It is customary to have all BIND-related files and directories under /var/named but you can also use some other location if your /var is limited in space. We will need approximately 7MB of disk space for the base setup, and you probably want to reserve some additional space for your logs, zone, and dump files.

Edit /etc/passwd, /etc/shadow, and /etc/group to set up the bind account. In the following text we will discuss the entries that must be added in each file.

```
/etc/passwd:
bind:x:65533:65533:Berkeley Internet Name Daemon:/var/named:/nonexistent
```

bind does not have to have any specific user or group id, as long as those do not conflict with any existing accounts. Since bind is meant to be a restricted account with minimum access rights and no login, it is a good practice to assign a first available id in descending order from 65535, the highest possible user-id. Note that the home directory of the account points to the root of the BIND hierarchy. This is not absolutely necessary yet it is quite handy.

```
/etc/shadow:
bind:NP:6445::::::
```

The password field for bind in /etc/shadow must contain NP to disable logins.

```
/etc/group:
bind::65533:bind
```

To be complete we also want to set up a group for bind. You should select the highest available group id for the account. The only member of the group is bind itself.

Files and Directories

Careful construction of the change-rooted directory hierarchy is critical to our setup. The guidelines described in the manual page of in.ftpd[6] offer a good starting point.

1. Create the /var/named directory.

```
# mkdir /var/named
# chown root:root /var/named
# chmod 551 /var/named
# ls -ld /var/named
dr-xr-x--x 4 root root 512 Feb 11 14:04 /var/named
```

Create the /var/named directory and set its owner and group to root. Set the directory permissions to mode 551 to permit read and directory access for the owner and group and directory-only access for everyone else (including bind).

2. Change your default directory to /var/named and create the first-level directories under named/.

Here we create the top level of our change-rooted environment. There will be only a few directories in our structure named will need write access to. For example, there has to be a read-write directory for named to store the transferred zone files.

```
# cd /var/named
# mkdir dev etc namedb usr var
# ln -s usr/sbin .
# chown root:root *
# chmod 551 *
# chmod 751 namedb
# ls -la
total 12
dr-xr-x--x      4 root   root     512    Feb 11 14:04      .
drwxr-xr-x      4 root   root     512    Feb 11 14:04      ..
dr-xr-x--x      4 root   root     512    Feb 11 14:04      dev
dr-xr-x--x      4 root   root     512    Feb 11 14:04      etc
drwxr-x--x      4 root   root     512    Feb 11 14:04      namedb
lrwxrwxrwx      1 root   root      10    Feb 11 14:04      sbin -> usr/sbin
dr-xr-x--x      4 root   root     512    Feb 11 14:04      usr
dr-xr-x--x      4 root   root     512    Feb 11 14:04      var
```

Our change-rooted sbin/ directory is a symbolic link to the change-rooted usr/sbin/. The purpose of this step is to store all executables and the corresponding shared libraries into a single hierarchy. Consequently, there will be no need to ever change the usr/ hierarchy after it has been set up. This serves the purpose of creating the change-rooted environment in the first place.

3. Change your directory to dev/ and create the necessary device nodes.

The devices are specific to your operating system and possibly even the version you are running. Therefore you should verify the major and the minor device numbers with 'ls -lL' from the system /dev directory before proceeding with creating the devices. Moreover, five out of seven devices we need are specific to Solaris: the only generic devices are dev/null and dev/zero.

```
# cd dev
# mknod conslog c 21 0
# mknod log c 21 5
# mknod null c 13 2
# mknod syscon c 0 0
# mknod tcp c 11 42
```

```
# mknod udp c 11 41
# mknod zero c 13 12
# chown root:sys *
# chown root:tty syscon
# chmod 666 conslog null tcp udp zero
# chmod 640 log
# chmod 620 syscon
# ls -la
total 4
dr-xr-x--x      2 root  root       512     Feb 11   15:06    .
dr-xr-x--x      8 root  root       512     Feb 11   14:59    ..
crw-rw-rw-      1 root  sys     21,   0     Feb 11   15:05    conslog
crw-r-----      1 root  sys     21,   5     Feb 11   15:05     log
crw-rw-rw-      1 root  sys     13,   2     Feb 11   15:05    null
crw--w----      1 root  tty      0,   0     Feb 11   15:05    syscon
crw-rw-rw-      1 root  sys     11, 42     Feb 11   15:05    tcp
crw-rw-rw-      1 root  sys     11, 41     Feb 11   15:05    udp
crw-rw-rw-      1 root  sys     13, 12     Feb 11   15:05    zero
```

We need to set up the dev/ directory for the proper operation of the network and syslog. Since BIND uses TCP and UDP, both dev/tcp and dev/udp device files must be present. These files should be readable and writable by the world (mode 666).

The files dev/conslog, dev/log, and dev/syscon control logging to the system console and syslog. BIND must be able to write to these devices for logging to work properly. One significant advantage of having these devices in the change-rooted area is the ability to collect log messages from BIND to syslog.

Traditionally the default log device, /dev/log, has been a UNIX domain socket. The Solaris /dev/log is a STREAMS device that nicely circumvents the boundaries of the change-rooted area. As a result, you don't have to specify additional log sockets for each change-rooted hierarchy: creating a private dev/log in your dev directory is enough.

Another benefit from using dev/log and syslog for logging is to move all log information beyond the reach of named as soon as a new entry is generated. This makes it considerably more difficult for an attacker to tamper with the logfiles.

The null device is there for garbage collection purposes. Any program, including BIND, may want to redirect its input or output to /dev/null, and an attempt to open a non-existent device would surely generate an error message.

The zero device is required by ld.so to set up shared libraries properly. If this device is missing you won't be able to start any change-rooted programs that utilize shared libraries.

4. Create the necessary system files in etc/.
Unlike the real /etc, our change-rooted etc/ will have only the minimum number of files, i.e., passwd, group, and netconfig. You have to change your default directory to ../etc/ to create these files.

In the following text we will use the ex line editor to create the system files. This is because it is easier to show all the right keystrokes for a line editor than for vi or textedit. Of course, you can always choose to use your favorite editor but you cannot use admintool since it can only modify the default system databases.

```
# cd ../etc
# ex passwd
```

```
"passwd" [New file]
:a
root:*:0:1:Super-User:/:/nonexistent
bind:*:65533:65533:Berkeley Internet Name Daemon:/:/nonexistent
.
:w
"passwd" [New file] 2 lines, 101 characters
:q
```

Our minimal passwd file will contain only two accounts, root and bind. These two are needed since we will start named as the super-user, but the owner of the process will change to bind as soon as possible. There will not be other user accounts.

```
# ex group
"group" [New file]
:a
root::0:root
other::1:root
sys::3:root
tty::7:root
bind::65533:bind
.
:w
"group" [New file] 4 lines, 54 characters
:q
```

Again, group will contain only the most necessary group entries. We have to include sys and tty, since we copied the ownerships and permissions of our change-rooted devices from the real /dev.

```
# ex netconfig
"netconfig" [New file]
:a
udp     tpi_clts        v  inet  udp  /dev/udp    -
tcp     tpi_cots_ord    v  inet  tcp  /dev/tcp    -
.
:w
"netconfig" [New file] 2 lines, 120 characters
:q
```

Our netconfig can be very minimal, since we need only UDP and TCP. This is one area where the manual page of in.ftpd and empirical data disagree, since the manual page instructs to include ticotsord into the list of networks and copy straddr.so to usr/lib/:

```
ticotsord tpi_cots_ord  v  loopback  -  /dev/ticotsord    straddr.so
```

Note that you would also have to create the corresponding device in the dev/ directory to use ticotsord:

```
# mknod /var/named/dev/ticotsord c 105 1
# chown root:sys /var/named/dev/ticotsord
# chmod 666 /var/named/dev/ticotsord
# ls -l /var/named/dev/ticotsord
crw-rw-rw-  1  root  sys   105,  1  Feb   11 15:05  /var/named/dev/ticotsord
```

Taking these steps and setting up ticotsord does not seem to yield anything, since named works fine without it. Moreover, you should not install a service or a facility into the change-rooted environment if you don't know what it does and especially if it is not really needed by the process you are setting up in the first place. The bottom line is that

there seems to be neither benefit nor harm in setting up ticotsord, which basically renders it unneeded in our change-rooted environment.

Finally, we have to set the proper ownership and permissions for individual files. All files in the directory must be owned by root and set read-only to everyone.

```
# chown root:root *
# chmod 444 *
# ls -la
total 10
dr-xr-x--x  2  root  root  512  Feb 11 15:05  .
dr-xr-x--x  8  root  root  512  Feb 11 15:05  ..
-r--r--r--  1  root  root   75  Feb 11 15:39  group
-r--r--r--  1  root  root  313  Feb 11 15:52  netconfig
-r--r--r--  1  root  root  101  Feb 11 15:10  passwd
```

Note that there is no etc/shadow. It is not required since the change-rooted environment has no login and therefore no passwords to store. Again, this differs from the guidelines for in.ftpd.

5. Set up usr/.
The usr/ hierarchy will accommodate the executables, the required shared libraries and time-zone information. Note that Solaris BIND uses /usr/tmp and BIND 8.2.3 uses /var/tmp. We will create a symbolic link for usr/tmp/ anyway, and the real location for our temp directory will be var/tmp/.

```
# cd ../usr
# mkdir -p lib sbin share/lib
# chown -R root:root *
# chmod -R 551 *
# ln -s ../var/tmp .
# ls -la
total 6
dr-xr-x--x        2 root   root     512  Feb 11 15:55        lib
dr-xr-x--x        2 root   root     512  Feb 11 15:55        sbin
dr-xr-x--x        3 root   root     512  Feb 11 15:55        share
lrwxrwxrwx        2 root   root      10  Feb 11 15:55        tmp -> ../var/tmp
```

The entire change-rooted usr/ hierarchy must be read-only. There are no files or directories in this subtree that would ever change once it has been constructed, and the permissions must be set accordingly.

The best choice to protect the change-rooted usr/ is to have it on a read-only media. This could also be a separate (loopback) file system that has been mounted read-only. One alternative, offered by 4.4BSD,[7] i.e., FreeBSD, is to set the "system immutable" flag for all the files and directories in the change-rooted usr/. An immutable file may not be changed, moved, or deleted. However, the kernel should run in a secure mode to prevent anyone from turning this flag off.

6. Copy the BIND executables to usr/sbin/.
The default installation directory for BIND 8 named and named-xfer is /usr/local/sbin. You only have to change your default directory to usr/sbin/ and copy the executables. These must be owned by root, and the permissions must be execute-only to the world.

```
# cd sbin
# cp -p /usr/local/sbin/named /usr/local/sbin/named-xfer .
# chown root:root *
# chmod 111 *
```

```
# ls -la
total 10
dr-xr-x--x  2 root    root        512    Feb 11 15:05  .
dr-xr-x--x  8 root    root        512    Feb 11 15:05  ..
--x--x--x   1 root    root     619400    Feb 11 14:52  named
--x--x--x   1 root    root     329200    Feb 11 14:52  named-xfer
```

Since we place named-xfer in the change-rooted usr/sbin/, as opposed to the default /usr/local/sbin, we have to specify the new location in named.conf. Otherwise named will not be able to find named-xfer and transfer zones.

7. Set up the shared libraries.
We have to copy the required shared objects—libraries—and ld.so.1, the run-time link editor, into our usr/lib/ in order to be able to execute BIND change-rooted. We will use the ldd command to find out what shared libraries are needed:

```
# ldd ./named
        libnsl.so.1 =>       /usr/lib/libnsl.so.1
        libsocket.so.1 =>    /usr/lib/libsocket.so.1
        libc.so.1 =>         /usr/lib/libc.so.1
        libdl.so.1 =>        /usr/lib/libdl.so.1
        libmp.so.2 =>        /usr/lib/libmp.so.2
        /usr/platform/SUNW,Ultra-5_10/lib/libc_psr.so.1
```

Now we have to copy these libraries and ld.so.1 in our usr/lib/. Since Solaris maintains the specific name (the shared library itself) and a generic name (a symbolic link to the specific shared library) of each shared object, we will use tar to copy these to our environment properly. However, we can simply copy the files that have only the specific name, i.e., the link loader ld.so.1, architecture-specific libc_psr.so.1, and nss_files.so.1.

```
# cd ../lib
# ( cd /usr/lib ; tar cf - libc.so* libdl.so* libmp.so* \
libnsl.so* libsocket.so* ) | tar xvBf -
# cp -p /usr/platform/SUNW,Ultra-5_10/lib/libc_psr.so.1 .
# cp -p /usr/lib/ld.so.1 .
# cp -p /usr/lib/nss_files.so.1 .
# chown root:root *
# chmod 111 ld.so.1
# chmod 555 lib* nss*
# ls -la
total 4094
dr-xr-x--x      2 root    root        512    Feb 6 07:51   .
dr-xr-x--x      4 root    root        512    Feb 6 08:03   ..
---x--x--x      1 root    root     181840    Dec 9 01:05   ld.so.1
lrwxrwxrwx      1 root    root         11    Feb 6 05:51   libc.so ->         ./libc.so.1
-r-xr-xr-x      1 root    root    1015768    Dec 9 01:03   libc.so.1
-r-xr-xr-x      1 root    root      16932    Dec 9 01:03   libc_psr.so.1
lrwxrwxrwx      1 root    root         12    Feb 6 05:51   libdl.so ->        ./libdl.so.1
-r-xr-xr-x      1 root    root       4304    Dec 9 01:05   libdl.so.1
lrwxrwxrwx      1 root    root         12    Feb 6 05:51   libmp.so ->      ./libmp.so.2
-r-xr-xr-x      1 root    root       6732    Jul 15 1997   libmp.so.1
-r-xr-xr-x      1 root    root      19304    Jul 15 1997   libmp.so.2
lrwxrwxrwx      1 root    root         13    Feb 6 05:51   libnsl.so -.       /libnsl.so.1
-r-xr-xr-x      1 root    root     726968    Dec 9 01:03   libnsl.so.1
lrwxrwxrwx      1 root    root         16    Feb 6 05:51   libsocket.so  /libsocket.so.1
-r-xr-xr-x      1 root    root      53656    Jul 16 1997   libsocket.so.1
-r-xr-xr-x      1 root    root      27000    Jul 16 1997   nss_files.so.1
```

The most interesting shared object in our list is nss_files.so.1. It is required by the Name Service Switch, as documented in the manual page of nsswitch.conf.[8] Solaris has three possible databases to store user and password information, i.e., files in the /etc directory, NIS, and NIS+. Since our change-rooted setup uses its local passwd database only, we do not need other nss_* methods from /usr/lib, such as nss_nisplus.so.1. Note that we do not need nss_dns.so.1 either, since the operation of named does not require resolver.[9]

8. Set up the time zones.
The time-zone data files from /usr/share/lib/zoneinfo are needed to report timestamps correctly in logfiles. We only have to copy this hierarchy to our change-rooted usr/share/lib/ and ensure that the ownerships and permissions are properly set.

```
# cd ../share/lib
# ( cd /usr/share/lib ; tar cf - zoneinfo ) | tar xvBf -
# chown -R root:root .
# find . -type f -print | xargs chmod 444
# find . -type d -print | xargs chmod 551
# ls -la
total 8
dr-xr-x--x   3 root     root         512   Feb 6 05:47      .
dr-xr-x--x   3 root     root         512   Feb 6 05:47      ..
dr-xr-x--x  10 root     root        1536   Nov 25 1998      zoneinfo
```

9. Set up var/.
Named must be able to write into the subdirectories of var/. It uses the log, run, and tmp directories for its logfiles, control channel and process id storage, respectively.

Note that var/log/ will not be needed if you decide to set up named to use syslog exclusively. We will discuss the details of this configuration later in the text.

The control channel of named, var/run/ndc.d/ndc is a UNIX domain socket owned by the super-user. It is created before named changes its user-id to bind, and obviously, it is writable by the super-user only. You can use this to your advantage since var/run/ can be set to writable by root, read-only by the group, and unavailable by everyone else.

Finally, we have to keep var/tmp/ for named and named-xfer to write their temporary files. These may potentially all use the available disk space, but at least the risk has been contained in a single directory that can be monitored regularly. You may also want to consider limiting the disk usage of bind by setting up a file system quota.

```
# cd ../../../var
# mkdir -p log run/ndc.d tmp
# chown -R root:bind log
# chown -R root:root run tmp
# chmod 771 log
# chmod 750 run
# chmod 1777 tmp
# ls -la
total 10
dr-xr-x--x      5 root     root       512   Feb 11 06:00      .
dr-xr-x--x      8 root     root       512   Feb 11 07:08      ..
drwxrwx--x      2 root     bind       512   Feb 11 15:06      log
drwxr-x---      3 root     root       512   Feb 11 15:37      run
drwxrwxrwt      2 root     root       512   Feb 11 15:00      tmp
```

10. Configure BIND.

First, we have to change our default directory to /var/named/namedb/ and create the secondary/ subdirectory. This directory will host all the zone files transferred from nameservers that this host is secondary for.

```
# cd ../namedb
# mkdir secondary
# chown root:bind secondary
# chmod 2770 secondary
# ls -ld secondary
drwxrws---  2 root    bind      512   Feb 20 10:26  secondary
```

Set the owner and the group of the subdirectory to root and bind, respectively. Set the permissions to read-write to the owner and the group, set-group-id to the group, and inaccessible to the others. The purpose of the set-group-id mode is to force the group ownership of the files in secondary/ to bind, regardless of the creator of a file.

Now we have our files and directories set, and we can proceed by creating named.conf, the BIND configuration file. In the following example we will only set up a cache-only nameserver that runs in our change-rooted environment. Note that the setup of master and slave nameservers and how to run your DNS are not in the scope of this document: you should consult Albitz & Liu[10] on those topics.

We have to use the options statement to specify the directory and file locations. All absolute pathnames in the following example are in respect to the change-rooted environment. In other words, although the real working directory of BIND is /var/named/namedb, after chroot() the named process sees /var/named as the root directory ('/') and the path to the working directory will be /namedb.

Note that all named-created files will be stored into the change-rooted var/tmp, i.e., /var/named/var/tmp. These files will have bind as the owner and the group.

```
options {
        directory "/namedb";
        named-xfer "/sbin/named-xfer";
        pid-file "/var/tmp/named.pid";
        dump-file "/var/tmp/named_dump.db";
        memstatistics-file "/var/tmp/named.memstats";
        statistics-file "/var/tmp/named.stats";
};
```

We have to set up the named control channel in a safe place. The preferred location, which is subject to file system security on your host, is a UNIX domain socket in the change-rooted var/run/ndc.d. The controls statement sets the owner, and the group of the directory, where the control socket resides (e.g., var/run/ndc.d), to root and read-write permissions to the owner only.

This behavior really is a workaround, since Solaris cannot set permissions for UNIX domain sockets. The permissions are handled properly, for example, on FreeBSD.

```
controls {
        unix "/var/run/ndc.d/ndc" perm 0600 owner 0 group 0;
};
```

We will need only the reverse map for localhost (127.0.0.1) and the list of root nameservers for a cache-only nameserver.

```
zone "0.0.127.in-addr.arpa" in {
      type master;
      file "db.127.0.0";
};
zone "." in {
      type hint;
      file "db.cache";
};
```

If your nameserver acted as a slave to other nameservers, you should store the zone files in the secondary/ subdirectory. As with the files defined in the options statement, the zone files will have bind as the owner and the group. The following fictitious zone is here as an example only:

```
zone "sub.domain.net" in {
      type slave;
      file "secondary/bak.sub.domain.net";
      masters { 169.254.27.16; };
};
```

You can save all log information into the change-rooted var/log/ directory. In our example we maintain three versions and roll the logfile over if its size exceeds 200KB.

```
logging {
      channel local_log {
          file "/var/log/named.log" versions 3 size 200k;
          severity notice;
          print-category yes;
          print-severity yes;
          print-time yes;
      };
      category lame-servers { null; };
      category default { local_log; };
};
```

Storing named log information to the change-rooted area has two problems. From the forensics point of view, it is generally not considered a good idea to leave audit information within the reach of possibly compromised server process. If the change-rooted area is compromised, there is a danger that the attacker could tamper with the log entries.

The management aspect to logging speaks for centralized log control. Since Solaris syslog can collect log information from a change-rooted dev/log device, we only need to decide which log facility to use. In the following example we will use LOG_LOCAL0:

```
# cd /etc
# ex syslog.conf
"syslog.conf" 34 lines, 981 characters
:a
local0.info                     /var/log/named.log
.
:w
"syslog.conf" 36 lines, 1049 characters
:q
# touch /var/log/named.log
# chown root:root /var/log/named.log
# ls -l /var/log/named.log
-rw-r--r--   1 root     root   10453    Feb 23 05:29  /var/log/named.log
```

The last step is to define the channel and the category for logging. Note that we cannot use the built-in default_syslog channel since once a channel has been defined, it cannot be altered. We can still define our own channel and modify the default category to forward all logging to syslog:

```
logging {
      channel local_syslog {
          syslog local0;
          severity notice;
          print-category yes;
          print-severity yes;
          print-time yes;
      };
      category default {
          local_syslog;
      };
};
```

Starting Up

Now that we have our directory hierarchy, files, and devices configured, we can start named change-rooted. This simply means entering the following command:

```
# /usr/sbin/chroot /var/named /sbin/named -u bind -c /namedb/named.conf
# ps -ef | grep bind | grep -v grep
  bind 27621  1  0 17:07:12 ?  0:00 /sbin/named -u bind -c /namedb/named.conf
# tail /var/adm/messages
Feb 11 17:07:12 london.greenwichtech.com named[27620]: starting
        (/namedb/named.conf).
```

You can use ndc to control and stop named, but you have to specify the pathname to the ndc control socket:

```
# ndc -c /var/named/var/run/ndc.d/ndc stop
```

We want named to start automatically during system boot, which means adding the aforementioned command in the RC scripts. Instead of modifying /etc/init.d/inetsvc, the default location to start named in RC, we should keep the original Solaris script intact and write our own RC script that will be executed immediately after inetsvc, aka, /etc/rc2.d/S72inetsvc. Edit /etc/init.d/named as follows:

```
# cd /etc/init.d/
# ex named
"named" [New file]
:a
#!/bin/sh

BINDD="/var/named"
PIDF="$BINDD/var/tmp/named.pid"
NDCHANNEL="$BINDD/var/run/ndc.d/ndc"

case "$1" in
'start')
      if [ -x /usr/sbin/chroot -a \
                -x "$BINDD/sbin/named" -a \
                -f "$BINDD/namedb/named.conf" \
           ]; then
          cd "$BINDD" > /dev/null 2>&1
          if [ $? -ne 0 ]; then
              echo "${0}: ${BINDD}: cannot cd." >&2
```

```
            exit 1
        else
            /usr/sbin/chroot "$BINDD" \
                /sbin/named -u bind -c /namedb/named.conf
            if [ $? -eq 0 ]; then
                echo 'chroot-named starting''
            fi
        fi
    fi
    ;;
'stop')
    if [ -x /usr/local/sbin/ndc -a \
            -r "$NDCHANNEL" -a \
            -w "$NDCHANNEL" \
        ]; then
        /usr/local/sbin/ndc -c "$NDCHANNEL" stop
    fi
    ;;
*)
    echo "Usage: $0 { start | stop }"
    exit 1
    ;;
esac
exit 0
:w
"named" [New file] 39 lines, 715 characters
:q
```

Our RC script checks that the executables and the configuration file are available and then proceeds with starting up named. To stop named, the script checks that the name daemon control program, ndc, and the control channel are available and then requests named to shut itself down.

Please note that although chroot() changes the root directory of your process, it does not change the default directory. To change-root cleanly we should change our working directory to the change-rooted area before calling chroot(). Secondly, the Solaris fchroot() system call can be used to reverse the operation of change-root if there is an open file descriptor to a directory before chroot() was called.

The owner and the group of the named script must be root and sys, respectively. The permissions of the file must be read-write and execute to the owner and read-only to the group and the others. After you have set the ownership and the permissions you must create the symbolic links for the RC directories:

```
# chown root:sys named
# chmod 744 named
# cd ../rcS.d
# ln -s ../init.d/named K42named
# cd ../rc0.d
# ln -s ../init.d/named K42named
# cd ../rc1.d
# ln -s ../init.d/named K42named
# cd ../rc2.d
# ln -s ../init.d/named S72named
```

Now we are all set, and you can start and stop BIND with the RC script. named will start automatically when you reboot the system next time.

Version Control

One common management problem with a change-rooted area is maintaining consistent versions of your binaries and other executables between the change-rooted environment and the rest of the system. For example, when the vendor releases a patch for BIND or the C library, you want to be sure that your change-rooted area(s) will also get updated. On the other hand, you do not want to update the change-rooted executables automatically, without evaluating at least the dependencies first.

You may want to consider your standard UNIX environment as "development" or "staging" and your change-rooted areas as "production." Often there are strict procedures that define how applications and services move from development to production. Accordingly, when you install a vendor patch, it goes to the development/staging area. You may want to set up a secondary change-rooted area for testing purposes and update your production area only after you have ensured that the patched executable or library won't break anything.

To avoid inconsistency in change-rooted production, you probably want to monitor not only integrity of the change-rooted files but also compare them with their counterparts elsewhere in the system. You may not implicitly remember which production files you have to update afterwards, but at least you will be notified.

Conclusions

The text has described how to set up and run BIND 8 change-rooted on Solaris and more specifically, on Solaris-2.6. This is the most protective setup available on UNIX, and the same methodology can be applied to change-rooting other UNIX services as well. The author has been running change-rooted BIND 8.2.3 on Solaris and FreeBSD since the release of the recent CERT advisory.[11]

Generally speaking, it is unwise to run any super-user-owned network service on your machine, especially if you haven't taken any steps to protect the operating system from an attack. From the practical point of view, there is no reason why you should not change-root your named, since, requiring few external resources, it is almost an ideal program to be confined.

The most significant threat to running named or any other UNIX process change-rooted is the possibility that the attacker might be able to revert back to the system root. For this reason a change-rooted process is never completely safe if it is owned by the super-user.

REFERENCES

1. S. Burr, How to break out of a chroot() jail, *http://www.bpfh.net/simes/computing/chroot-break.html*, January 31, 2001.

2. W. Venema, Chrootuid-1.2, *ftp://coast.cs.purdue.edu/pub/tools/unix/sysutils/chrootuid/*, August 16, 1993.

3. H. Pomeranz et al., *Solaris Security: Step-by-Step*, Version 1.0, The SANS Institute, 1999.

4. Internet Software Consortium, BIND 8.2.3 source package, *ftp://ftp.isc.org/isc/bind/src/8.2.3/bind-src.tar.gz*, January 26, 2001.

5. Sunfreeware.com, BIND 8.2.3 Package for SPARC/Solaris8, *ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/bind-8.2.3-sol8-sparc-local.gz*, January 29, 2001.

6. Sun Microsystems, Inc., *in.ftpd – Internet File Transfer Protocol server*, Solaris System Administrator's Reference Guide, March 4, 1997.

7. M. K. McKusick, et al., *The Design and Implementation of the 4.4BSD Operating System*, Addison-Wesley, 1996.

8. Sun Microsystems, Inc., *nsswitch.conf – configuration file for the name service switch*, Solaris System Administrator's Reference Guide, April 28, 1997.

9. Sun Microsystems, Inc., *resolver – resolver routines*, Solaris Programmer's Reference Guide, December 30, 1996.

10. P. Albitz, C. Liu, *DNS and BIND*, 3rd Edition, O'Reilly & Associates, September 1998.

11. CERT® Coordination Center, CA-2001-02: Multiple Vulnerabilities in BIND, *http://www.cert.org/advisories/*, January 29, 2001.