Chroot−BIND HOWTO

Table of Contents

Chroot−BIND HOWTO

Scott Wunsch, `scott at wunsch.org`

v1.0, 13 March 2000

This document describes installing the BIND 8 nameserver to run in a chroot jail and as a non−root user, to provide added security and minimise the potential effects of a security compromise.

5.The End

6.Appendix − Document Distribution Policy

1.Introduction

This is the Chroot−BIND HOWTO; see Where? for t he master site, which contains the latest copy. It is assumed that you already know how to configure and use BIND (the Berkeley Internet Name Domain). If not, I would recommend that you read the DNS HOWTO first. It is also assumed that you have a basic familiarity with compiling and installing software on your UNIX−like system.

1.1 What?

This document describes some extra security precautions that you can take when you install BIND. It explains how to configure BIND so that it resides in a ``chroot jail'', meaning that it cannot see or access files outside its own little directory tree. We shall also configure it to run as a non−root user.

The idea behind chroot is fairly simple. When you run BIND (or any other process) in a chroot jail, the process is simply unable to see any part of the filesystem outside the jail. For example, in this document, we'll set BIND up to run chrooted to the directory `/chroot/named`. Well, to BIND, the contents of this directory will appear to be `/`, the root directory. Nothing outside this directory will be accessible to it. You've probably encounted a chroot jail before, if you've ever ftped into a public system.

1.2 Why?

The idea behind running BIND in a chroot jail is to limit the amount of access any malicious individual could gain by exploiting vulnerabilities in BIND. It is for the same reason that we run BIND as a non−root user.

This should be considered as a supplement to the normal security precautions (running the latest version, using access control, etc.), not a replacement for them.

1.3 Where?

The latest version of this document is always available from the web site of the Linux/Open Source Users of Regina, Sask., at http://www.losurs.org/docs/howto/Chroot−BIND.html.

BIND is available from the Internet Software Consortium at http://www.isc.org/bind.html. As of this writing, the current version of BIND is 8.2.2_P5.

1.4 How?

I wrote this document based on my experiences in setting BIND up in a chroot environment. In my case, I already had an existing BIND installation in the form of a package that came with my Linux distribution. I'll assume that most of you are probably in the same situation, and will simply be transferring over and modifying the configuration files from your existing BIND installation, and then removing the package before installing the new one. Don't remove the package yet, though; we may want some files from it first.

If this is not the case for you, you should still be able to follow this document. The only difference is that, where I refer to copying an existing file, you first have to create it yourself. The DNS HOWTO may be helpful for this.

1.5 Disclaimer

These steps worked for me, on my system. Your mileage may vary. This is but one way to approach this; there are other ways to set the same thing up (although the general approach will be the same).

My BIND experience to date has been installing on Linux servers. However, most of the instructions in this document should be easily applicable to other flavours of UNIX as well, and I shall try to point out differences of which I am aware.

2.Preparing the Jail

2.1 Creating a User

As mentioned in the introduction, it's not a good idea to run BIND as root. So, before we begin, let's create a separate user for BIND. Note that you should never use an existing user like `nobody` for this purpose.

This requires adding a line something like the following to `/etc/passwd`:

```
named:x:200:200:Nameserver:/chroot/named:/bin/false
```

And one like this to `/etc/group`:

```
named:x:200:
```

This creates a user and group called `named` for BIND. Make sure that the UID and GID (both 200 in this example) are unique on your system. The shell is set to `/bin/false` because this user will never need to log in.

2.2 Directory Structure

Now, we must set up the directory structure that we will use for the chroot jail in which BIND will live. This can be anywhere on your filesystem; the truly paranoid may even want to put it on a separate volume. I shall assume that you will use `/chroot/named`. Let's start by creating the following directory structure:

```
/chroot
  +-- named
       +-- bin
       +-- dev
       +-- etc
       |    +-- namedb
       +-- lib
       +-- var
            +-- run
```

2.3 Placing the BIND Data

Assuming that you have already done a conventional installation of BIND and are using it, you will already have an existing `named.conf` and zone files. These files must now be moved (or copied, to be safe) into the chroot jail, so that BIND can get at them. `named.conf` goes in `/chroot/named/etc`, and the zone files can go in `/chroot/named/etc/namedb`. For example:

```
# cp -p /etc/named.conf /chroot/named/etc/

# cp -a /var/named/* /chroot/named/etc/namedb/
```

BIND will likely need to write to the `namedb` directory, and probably some of the files in it. For example, if your DNS serves as a slave for a zone, it will have to update that zone file. Also, BIND can dump statistical information, and does so in this directory. For that reason, you should probably make the `named` user the owner of this directory and its contents:

```
# chown −R named:named /chroot/named/etc/namedb
```

BIND will also need to write to the `/var/run` directory, to put its pidfile and ndc socket there, so let's allow it to do so:

```
# chown named:named /chroot/named/var/run
```

2.4 System Support Files

Once BIND is running in the chroot jail, it will not be able to access files outside the jail **at all**. However, it needs to access a few key files, such as the system's C library. Exactly what libraries are required will depend on your flavour of UNIX. For most modern Linux systems, the following commands will be sufficient to put the necessary libraries in place:

```
# cd /chroot/named/lib
# cp −p /lib/libc−2.*.so .
# ln −s libc−2.*.so libc.so.6
# cp −p /lib/ld−2.*.so .
# ln −s ld−2.*.so ld−linux.so.2
```

As an alternative, you could simply build statically−linked versions of the BIND binaries to put in your chroot jail.

BIND needs one more system file in its jail: good ol' `/dev/null`. Again, the exact command necessary to create this device node may vary from system to system; check your `/dev/MAKEDEV` script to be sure. For most Linux systems, we can use the following command:

```
# mknod /chroot/named/dev/null c 1 3
```

Finally, you need a couple extra files in the `/etc` directory inside the jail. In particular, you must copy `/etc/localtime` in there so that BIND logs things with the right time on them, and you must make a simple `group` file with the `named` group in it. The following two commands will take care of this:

```
# cp /etc/localtime /chroot/named/etc/

# echo 'named:x:200:' > /chroot/named/etc/group
```

Keep in mind that the GID, 200 in this example, must match the one you defined in the real `/etc/group` above.

2.5 Logging

Unlike a conventional jailbird, BIND can't just scribble its log entries on the walls :−). Normally, BIND logs through `syslogd`, the system logging daemon. However, this type of logging is performed by sending the log entries to the special socket `/dev/log`. Since this is outside the jail, BIND can't use it any more. Fortuantely, there are a couple options to work around this.

The Ideal Solution

The ideal solution to this dilemma requires a reasonably recent version of `syslogd` which supports the `−a` switch introduced by OpenBSD. Check the manpage for your `syslogd(8)` to see if you have such a version.

If you do, all you have to do is add the switch ``` ``−a/chroot/named/dev/log'' ``` to the command line when you launch `syslogd`. On systems which use a full SysV−init (which includes most Linux distributions), this is typically done in the file `/etc/rc.d/init.d/syslog`. For example, on my Red Hat Linux system, I changed the line

```
daemon syslogd −m 0
```

to

```
daemon syslogd −m 0 −a /chroot/named/dev/log
```

The simply restart `syslogd`, either by killing it and launching it again, or by using the SysV−init script to do it for you:

```
# /etc/rc.d/init.d/syslog stop
# /etc/rc.d/init.d/syslog start
```

Once it's been restarted, you should see a ``file'' in `/chroot/named/dev` called `log`, that looks something like this:

```
srw−rw−rw−   1 root     root            0 Mar 13 20:58 log
```

The Other Solutions

If you have an older `syslogd`, then you'll have to find another way to do your logging. There are a couple programs out there, such as `holelogd`, which are designed to help by acting as a ``proxy'' and accepting log entries from the chrooted BIND and passing them out to the regular `/dev/log` socket.

Alteratively, you can simply configure BIND to log to files instead of going through syslog. See the BIND documentation for more details if you choose to go this route.

3.Compiling BIND

You should be able to find the BIND source by visiting http://www.isc.org/bind.html. You need the `bind−src.tar.gz` package. Be sure to get the latest version!

3.1 Modifying Paths

Things can get a bit confusing at this point, because different parts of the BIND package will be referring to the same directories by different names (depending on whether or not they're running inside the jail). I'll try not to confuse you **too** much :−).

The main directory that we have to worry about here is `/var/run`, because its contents are required for both the main `named` daemon (inside the jail), and the `ndc` utility (on the outside). We'll start by setting everything up to find this directory from the outside world. To do this, we need to modify `src/port/linux/Makefile.set` (substitute your port's directory if you're not running Linux), and change the line

```
DESTRUN=/var/run
```

to

```
DESTRUN=/chroot/named/var/run
```

While you're in there, you may want to change the other destination paths from `/usr` to `/usr/local`.

Now everything should be able to find that directory... except the `named` daemon itself, to which it's still just `/var/run` inside the jail. We can get around this by making a small change in the `named` source. In the file `src/bin/named/named.h`, find the line

```
#include "pathnames.h"
```

and add the following line immediately after it

```
#define _PATH_NDCSOCK    "/var/run/ndc"
```

This way, `named` will ignore our definition of `DESTRUN` over in `Makefile.set` and use the correct location (from its perspective in the chroot jail). You will notice some warnings about redefinitions of _PATH_NDCSOCK when you do the build; just ignore them.

3.2 Doing the Build

You should now be able to compile BIND as normal, following the instructions in the `INSTALL` file. At this stage, we only want to compile BIND, not install it. Don't go too far when following the `INSTALL` file. Essentially, it's just `make clean`, `make depend`, and `make`.

4.Installing Your Shiny New BIND

I should mention that if you have an existing installation of BIND, such as from an RPM, you should probably remove it before installing the new one. On Red Hat systems, this probably means removing the packages `bind` and `bind−utils`, and possibly `bind−devel` and `caching−nameserver`, if you have them.

You may want to save a copy of the init script (e.g., `/etc/rc.d/init.d/named`), if any, before doing so; it'll be useful later on.

4.1 Installing the Tools Outside the Jail

This is the easy part :−). Just run `make install` and let it take care of it for you. You may want to `chmod 000 /usr/local/sbin/named` afterwards, to make sure you don't accidentally run the non−chrooted copy of BIND. (This is `/usr/sbin/named` if you didn't tell it to go in `/usr/local/sbin` like I suggested.)

4.2 Installing the Binaries in the Jail

Only two parts of the package have to live inside the chroot jail: the main `named` daemon itself, and `named−xfer`, which it uses for zone transfers. You can simply copy them in from the source tree:

```
# cp src/bin/named/named /chroot/named/bin

# cp src/bin/named-xfer/named-xfer /chroot/named/bin
```

4.3 Setting up the Init Script

If you have an existing init script from your distribution, it would probably be best simply to modify it to run `/chroot/named/bin/named`, with the appropriate switches. The switches are... *(drumroll please...)*

- `−u named`, which tells BIND to run as the user `named`, rather than `root`.

- `-g named`, to run BIND under the group `named` too, rather than `root` or `wheel`.
- `-t /chroot/named`, which tells BIND to chroot itself to the jail that we've set up.

The following is the init script I use with my Red Hat 6.0 system. As you can see, it is almost exactly the same as the way it shipped from Red Hat.

```
#!/bin/sh
#
# named           This shell script takes care of starting and stopping
#                 named (BIND DNS server).
#
# chkconfig: 345 55 45
# description: named (BIND) is a Domain Name Server (DNS) \
# that is used to resolve host names to IP addresses.
# probe: true

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /chroot/named/bin/named ] || exit 0

[ -f /chroot/named/etc/named.conf ] || exit 0

# See how we were called.
case "$1" in
  start)
        # Start daemons.
        echo -n "Starting named: "
        daemon /chroot/named/bin/named -u named -g named -t /chroot/named
        echo
        touch /var/lock/subsys/named
        ;;
  stop)
        # Stop daemons.
        echo -n "Shutting down named: "
        killproc named
        rm -f /var/lock/subsys/named
        echo
        ;;
  status)
        /usr/local/sbin/ndc status
        exit $?
        ;;
  restart)
        /usr/local/sbin/ndc restart
        exit $?
        ;;
  reload)
        /usr/local/sbin/ndc reload
        exit $?
        ;;
  probe)
        # named knows how to reload intelligently; we don't want linuxconf
```

```
        # to offer to restart every time
        /usr/local/sbin/ndc reload >/dev/null 2>&1 || echo start
        exit 0
        ;;
  *)
        echo "Usage: named {start|stop|status|restart}"
        exit 1
esac

exit 0
```

4.4 Configuration Changes

You will also have to add or change a few options in your `named.conf` to keep the various directories straight. In particular, you should add (or change, if you already have them) the following directives in the `options` section:

```
directory "/etc/namedb";
pid-file "/var/run/named.pid";
named-xfer "/bin/named-xfer";
```

Since this file is being read by the `named` daemon, all the paths are of course relative to the chroot jail.

5.The End

5.1 Launching BIND

Everything should be set up, and you should be ready to put your new, more secure BIND into action. Assuming you set up a SysV−style init script, you can simply launch it as:

```
# /etc/rc.d/init.d/named start
```

Make sure you kill any old versions of BIND still running before doing this.

If you take a look at your logs, you should find the initialisation messages that BIND spits out when it loads. (If not, there's a problem with your logging configuration that you need to fix.) Amongst those messages, BIND should tell you that it chrooted successfully, and that it is running as the user and group `named`. If not, you have a problem.

5.2 That's It!

You can go take a nap now ;−).

6.Appendix − Document Distribution Policy
