

Secure Installation of BIND

Holt Sorenson 2000-02-08

Conventions

(1...) - bibliographical references

(a..z) - information about commands

"ls -la" - type everything inside the quotes to the shell

This document was written using BIND 8 (specifically 8.2.2-p5). It is possible that some of it's recommendations will not work for any version earlier than 8.2.2-p5. It would also be wise to not use any versions of BIND earlier than 8.2.2-p5. The reasoning behind this statement is that BIND has had security issues that result in total compromise of machines that host it, and Denial of Service attacks that prevent the name server from answering queries. At the time of writing there were seven known vulnerabilities in pre 8.2.2 versions of BIND, and a vulnerability in the way DNS is deployed that affects BIND 8.2.2 when you change your nameserver from one ip address to another. (1)

Brief History

The first "DNS" system, JEEVES, was written by Paul Mockapetris. In the early eighties, the Berkeley Internet Name Daemon (BIND) was first created at the University of California at Berkeley in response to a government (DARPA) grant. It was maintained for some time at Berkeley and later became software maintained by DEC when Kevin Dunlap, an employee of DEC, worked at Berkeley. Later, Paul Vixie assumed the role of maintainer, and has recently announced that his role in maintaining BIND will cease with version 8, and that version 9 is currently in development. (2, 3, 4)

Intro

DNS performs a most critical function on internet protocol based networking infrastructure. It is critical because it tells machines what host addresses answer the names that we humans give to our machines. This implies (for better or worse) that we trust the information that the DNS server gives us. The queries it answers result in hostname to numerical IP address mapping, in situations where a compromise has not occurred. Change rooting and having the nameserver run as a user and a group that has limited or no privileges helps us stay closer to that no-compromise ideal. This makes it easier for us to trust the name server's responses. What scenarios become more likely if we chose not to use features such as change rooting and non-

privileged execution? Let us consider one, for a moment.

Lets say that Foobar, Inc. has defined in their DNS that their webserver answers on 10.0.0.22, and is named www.foobar.com. As a part of doing business, they tell their customers to visit the portion of their webserver that allows their customers to register their information with Foobar as part of a service that Foobar provides. Being security minded, Foobar's information systems personnel have protected this portion of their website with SSL (or TLS) and have purchased a certificate for their webserver. This allows their customers to (if they know how and if they know they should do this) verify the certificate that identifies Foobar's webserver. The customers can then trust that the information that they send to Foobar's webserver is going to Foobar, Inc. because they trust all the components of the process that allow them to engage in this transaction, including the certificate that identifies the Foobar, Inc.'s website.

One of Foobar, Inc's competitors, Scratch Monkeys, Inc., hires an industrial espionage professional, Mallory, to steal Foobar's customer data. Mallory learns that FBI's information systems personnel set up this server and the customer service reps at Foobar, Inc. asked their customers to register on this server. Mallory probes Foobar, Inc.'s network and finds their DNS running on a compromisable host. Mallory attacks the host and is able to edit Foobar, Inc.'s DNS files, pointing their customers to a server (192.168.17.3, note: not the same address as FBI's webserver). Mallory sets up this server so that it looks identical to their system, with the exception that the SSL certificate is not the same. However, when Mallory requested the SSL certificate with a CA (certificate authority) whose root certificate is in Foobar, Inc.'s customer's browsers, he was able to convince the CA that he was a legitimate employee of Foobar, Inc. The customer of FBI neglects to do an out of band verification of the certificate with Foobar, Inc. (how many of us really check the verity of certificates anyway ?), and since the CA root is in Foobar, Inc.'s customer's browsers, there are no warnings about the fact that they are connecting to Mallory's server. Foobar, Inc.'s customer's give up their data to Mallory, who, in turn, passes that data to Scratch Monkey's, Inc. Now Scratch Monkey's, Inc. now has all of FBI's customer's information that have registered at Mallory's server, and so they direct market Foobar, Inc.'s customers and decrease FBI's market share. The most obvious way in which this attack could be mitigated is by assuring that the attacker can't penetrate the host serving up DNS, which Foobar, Inc.'s customers trust will tell them the address that has been defined as correct for the customers to transact with for registration. As we can see from this scenario, the price of doing business across the Internet can be quite exorbitant if one isn't vigilant from a security perspective. What is more disturbing is that there are ad infinitum permutations on this scenario that allow the possibility of compromise, if informations systems personnel are not vigilant. To quote, with slight modification: "Network Security is [what] Y2k [was promised to

be] without the deadline." (5)

Implementation

In order to protect your host that will be serving up DNS, the bare minimum that should be done is to update the OS to latest patch revisions, remove all network services and software packages that are not necessary for the host to accomplish its tasks, and administrate the host through encrypted tunnels. It is also unwise to have critical services running on hosts that serve less critical or less trusted services. If you dedicate a single host to a single critical service, you're generally assured that the only thing that will get compromised, if that host is compromised, is that service. This statement, however, does not apply if that service is trusted by other network entities. If you haven't at least locked down the host, then you should stop reading this paper, take care of the minimums to lock it down, and then come back to this paper and finish setting up your DNS server.

Now that the host OS is ready, lets discuss what we're going to accomplish. We've decided that the requirements we're going place on the DNS server are that it runs as user named, group named and that it will be change rooted to a directory called `/opt/chroot`. This means that when BIND starts it will see the directory `/opt/chroot` as `/`, instead of seeing `/` (the root of the directory tree on your machine) as `/`. This is analogous to you logging into your box, and never needing to leave your home directory. In this case, everything, including the shared object libraries, configuration files, and anything else that you need to execute the commands that help you use your machine, live in your home directory or below it. If an attacker happened upon your machine and knew of an attack that was present in BIND that hadn't been fixed by BIND's maintainers, then the most privileges they would have, upon compromising BIND, are the privileges associated with the named user, and they would be stuck in a directory structure that only allowed them to fool with BIND. Since we will set up your name tables to be editable by a user other than the named user, they wouldn't be able to modify the tables, unless they could find a way to get root, from inside the chrooted jail. Since there won't be any setuid binaries of any kind (owned by any user, including root) at all inside the chrooted jail, the likelihood that they would be able to compromise your machine, or your name tables is slim, if not near impossible. One other thing to note is that you will need to allow the `var/run` and `var/log` directories to be writable by the named group. want the named.pid file to be written out. It would be wise to place the BIND directory structure on its own partition so that if named is compromised, the worst that can theoretically happen to the partition is that this partition gets filled.

The following are assumed for the configuration:

- The domain is foobar.com
- BIND will be serving up DNS for network: 10.0.0.0/24
- The primary nameserver will be: ns 10.0.0.2
- The secondary nameserver will be: ns2 10.0.0.3
- The maintenance user and group will be:
- maintusr: : 20100: 20100: : /var/named: /bin/bash
- GID 20100 = maintusr
- The name server will run as:
- named: : 20000: 20000: : /opt/bind: /bin/bash
- GID 20000 = named
- You've created the above users and groups
- You've downloaded the most current release of BIND from:
<http://www.isc.org/products/BIND/>
- You have gzip compiled and installed (ftp://ftp.gnu.org/pub/gnu/gzip)
- You've decided where you're putting BIND. The following instructions assume /opt/bind-nested inside /opt/chroot.

Some of the commands below assume you are using bash or a csh compatible shell.

Let's get started!

1. For de-archiving and compiling you need to be in an area you can write to, such as your home directory. Since neither of these operations are privileged, there is no need to be root.

de-archive the source with a command such as:

```
gzip -dc bind-src.tar.gz |tar xf -
```

or `tar xzf bind-src.tar.gz` (if you have gnu tar with your PATH set)

change directory to `src/port/linux`, or `src/port/solaris`, depending on OS.

2. Edit `Makefile.set` and change all the DEST (destination variables) following the model below (the man pages should go outside the chroot):

```
DESTBIN=/opt/bind-8.2.2-p5/bin
```

```
DESTSBIN=/opt/bind-8.2.2-p5/sbin
```

```
DESTEXEC=/opt/bind-8.2.2-p5/sbin
```

```
DESTMAN=/opt/local/man
DESTHELP=/opt/bind-8.2.2-p5/lib
DESTETC=/opt/bind-8.2.2-p5/etc
DESTRUN=/opt/bind-8.2.2-p5/var/run
```

- return to the top of the build tree src directory) and:

```
make depend
make
```

- become root and:

```
make install
```

The result of the make install that we care about will be the files placed into /opt/bind-8.2.2-p5. /usr/local/bind is also created and can be deleted as it contains the library and include files which are not needed unless you're planning further BIND related development. Since this is a production machine you want secured, it is not likely you need this. Unless you are absolutely sure you need them, delete these files.

- Next:

```
mkdir -p /opt/chroot/opt/bind-8.2.2-p5/etc
mkdir -p /opt/chroot/var/{run,log,named} /opt/chroot/lib (a)
```

- Next:

```
cd /opt/bind-8.2.2-p5
tar cf - *|(cd /opt/chroot/opt/bind-8.2.2-p5;tar xvf -)
(b) cd /opt/chroot/opt
rm -fr /opt/bind-8.2.2-p5
```

- Setup links:

```
ln -s bind-8.2.2-p5 bind (while in /opt/chroot/opt)
ln -s chroot/opt/bind bind (while in /opt)
ln -s ../../var (while in /opt/chroot/opt/bind-8.2.2-p5)
```

- Use ldd to see what shared object libraries named and named-xfer rely on (Please note that the output below could differ on your system, and you need to use what is your system's output if it does.):

```
cd ~named/sbin
ldd named named-xfer
```

Binary	Linux	Solaris
--------	-------	---------

named	<pre>libc.so.6 => /lib/libc.so.6 (0x40019000) /lib/ld-linux.so.2 => /lib/ ld-linux.so.2 (0x40000000)</pre>	<pre>libl.so.1 => /usr/lib/ libl.so.1 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</pre>
named-xfer	<pre>libc.so.6 => /lib/libc.so.6 (0x40019000) /lib/ld-linux.so.2 => /lib/ ld-linux.so.2 (0x40000000)</pre>	<pre>libl.so.1 => /usr/lib/ libl.so.1 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</pre>

Copy the files listed in the ldd output files like so:

- o Linux:

```
cp -p /lib/libc.so.6 /lib/ld-linux.so.2 /opt/chroot/lib
```

- o Solaris:

```
cp -p /usr/lib/libl.so.1 /usr/lib/libnsl.so.1 \
/usr/lib/libsocket.so.1 /usr/lib/libc.so.1 \
/usr/lib/libdl.so.1 /usr/lib/libmp.so.2 /opt/chroot/usr/lib
```

Under Solaris, make the tcp and udp devices:

```
cd /opt/chroot/dev
mknod tcp c 11 42
mknod udp c 11 41
chmod 666 tcp udp
```

9. Setup your named.conf and your zones
10. Set permissions:

```
cd /opt/chroot
find . -type f -perm -111 -exec strip {} \;
find . -type f -exec chmod ug-s {} \;
find . -type f -perm -111 -exec chmod -r {} \;
chown -R root:named opt var
chown -R maintusr:named var/named
chmod -R g-w var
chmod -R -w opt lib
chmod -R o-rx .
chmod g+w var/run var/log
touch var/log/all.log var/run/named.pid
chown named:named var/log/all.log var/run/named.pid
```
11. You can delete the following without altering the nameserver's ability to function:
 - o opt/bind/bin/*
 - o opt/bind/sbin/{dnskeygen,irpd,named-bootconf,ndc}

Outro

Setting up a secondary server will be left as an exercise to the reader. The permissions above are set to allow the BIND DNS server to access only the resources it needs inside the change root. In the event of compromise, the cracker would only be able to abuse what the server had permission to use, which equates to "not much". By adapting the steps enumerated above to your environment, you can run BIND in a change-rooted environment, as a non-root user, and with access control on your DNS tree and the zones contained in the tree. This will at minimum make attacks against your BIND server more difficult, and at maximum, help you thwart known attacks against BIND and maintain security and integrity on your BIND DNS server.

Bio

Holt Sorenson is currently employed at Counterpane Internet Security where he chases and herds datagrams and performs system administration on several Unices. Prior to working at CIS, he was employed by Utah Education Network, where he was responsible for security on the WAN that provided commodity and VBNS Internet traffic to higher education, public education, and state agencies in Utah. Comments on this article are appreciated, send them to hso@nosneros.net

Bibliography:

1. see: BID (Bugtraq ID) 933 at <http://www.securityfocus.com>
2. DNS and Bind, O'Reilly Books, O'Reilly Books, ISBN: 1565925122
3. [RFC 819](#)
4. <http://www.isc.org/products/BIND/bind-history.html>
5. Cover of Hacking Exposed (Anonymous ?), ISBN: 0072121270
6. Command reference:
 - a. see Brace Expansion in the bash man page, filename substitution in the csh and tcsh man pages
 - b. see Compound Commands in bash man page, use / to search in your other shells' man pages for parenthesis or subshell
3. Other references:
 - [BIND Homepage \(Includes Docs\)](#)
 - <http://www.linuxdoc.org/HOWTO/DNS-HOWTO.html>
 - <http://www.dns.net/dnsrd/>
 - <http://www.acmebw.com/askmr.htm>
 - [DNS definition and resources](#)

Articles and General Resources in this Section

Subscribe to the [FOCUS-Sun Mail List](#)
by Security Focus Inc.

[Secure Installation of BIND](#)
by Holt Sorensen

[Secure Installation of Apache with SSL](#)
by Dale Coddington

Relevant Links

[BIND](#)

by The Internet Software Consortium

[DNS-HOWTO](#)

by Nicolai Langfeldt

[DNS Resources Directory](#)

by DNS Resources Directory

[Ask Mr. DNS](#)

by Ask Mr. DNS

[Privacy Statement](#)

Copyright 2006, SecurityFocus