# Securing and Optimizing Linux: DNS and BIND

*A hands on guide for Linux professionals*


**Title:** Securing and Optimizing Linux: **DNS and BIND**
**Author's:** Gerhard Mourani
**Mail:** gmourani@openna.com
**Website:** http://www.openna.com/
**Page Count:** 49
**Version:** rc1.0
**Last Revised:** March 02, 2001


## Author note

This documentation comes directly from the future release of “Securing & Optimizing Linux: Red Hat Edition v2.0”. Since we’re presently looking for a new editor for our book, and this can take some time, I decided to give you something in your waiting. By the way, If you are a publisher or you know of one interested in Linux Issues, please contact us at your earliest at service@openna.com. Thank you.

The ISC group project have release a new version of DNS and BIND, which is completely different from its predecessor. BIND 9 covers security area better than BIND 8 and in this documentation you will see it. This documentation is what I called “release candidate 1.0 rc1.0” because the BIND 9 release is not completely stable at this time (development continue) and for this reason contents of this documentation will surely change for the final release, which will be in “Securing & Optimizing Linux: Red Hat Edition v2.0”. Also error may exist in this documentation and as usually if you have any comments, contributions, correction, error report, tips, etc, don’t hesitate to contact me at gmourani@openna.com.

Finally, I would like to thank all of you who are reading my book since the beginning of this story and encouraged me to continue to develop it. Also a special thanks must be given to Olafur Gudmundsson <ogud@ogud.com>, which help me to better understand the new DNSSEC feature of BIND 9.


# Contents

*Author note* ..... *2*

**LINUX ISC BIND & DNS SERVER** ..... **4**

*Recommended RPM packages to be installed for a DNS Server* ..... *5*
*Compiling - Optimizing & Installing ISC BIND & DNS* ..... *8*
*Configuring ISC BIND & DNS* ..... *11*
*Caching-Only Name Server* ..... *12*
*Primary Master Name Server* ..... *15*
*Secondary Slave Name Server* ..... *19*
*Running ISC BIND & DNS in a chroot jail* ..... *25*
*Securing ISC BIND & DNS* ..... *28*
*Optimizing ISC BIND & DNS* ..... *42*
*ISC BIND & DNS Administrative Tools* ..... *45*
*ISC BIND & DNS Users Tools* ..... *46*


# Linux ISC BIND & DNS Server

## Abstract

Once we have installed all the necessary security software in our Linux server, it's time to improve and tune the networking part of our server. Domain Name System (DNS) is one of the **MOST** important network service for IP networks communication, and for this reason, all Linux **client** machines should be configured to perform caching functions as a minimum. Setting up a caching server for client local machines will reduce the load on the site's primary server. A caching only name server will find the answer to name queries and remember the answer the next time we need it. This will shorten the waiting time the next time significantly.

A Name Server (NS) is a program that stores information about named resources and responds to queries from programs called *resolvers*, which act as client processes. The basic function of an NS is to provide information about network objects by answering queries. Linux is a perfect and secure platform to run and deploy BIND DNS server, a number of Linux DNS servers in the Internet are listed as authoritative DNS servers for Microsoft's domains. Yes, Microsoft has partially outsourced the management of its Domain Name System (DNS) servers to Linux for the job. Oops

BIND (**B**erkeley **I**nternet **N**ame **D**omain) is a widely used, free implementation of the Domain Name System for Unix and Windows NT. It provides a server, a client library, and several utility programs. It is estimated to be the DNS software in use in over 90% of the hosts on the Internet and this is the one that we will describe further down in this chapter.

For security reasons, it is very important that DNS doesn't exist between hosts on the corporate network and external hosts; it is far safer to simply use IP addresses to connect to external machines from the corporate network and vice-versa.

In our configuration and installation we'll run ISC BIND & DNS as non root-user and in a chrooted environment. We also provide you three different configurations; one for a simple Caching Name Server Only (client), one for a Slave Name Server (Secondary DNS Server) and another one for a Master Name Server (Primary DNS Server).

The simple Caching Name Server configuration will be used for your servers that don't act as a Master or Slave Name Server, and the Slave and Master configurations will be used for your servers that act as a Master Name Server and Slave Name Server. Usually one of your servers acts as Primary/Master, another one acts as Secondary/Slave and the rest act as simple Caching client Name Server.

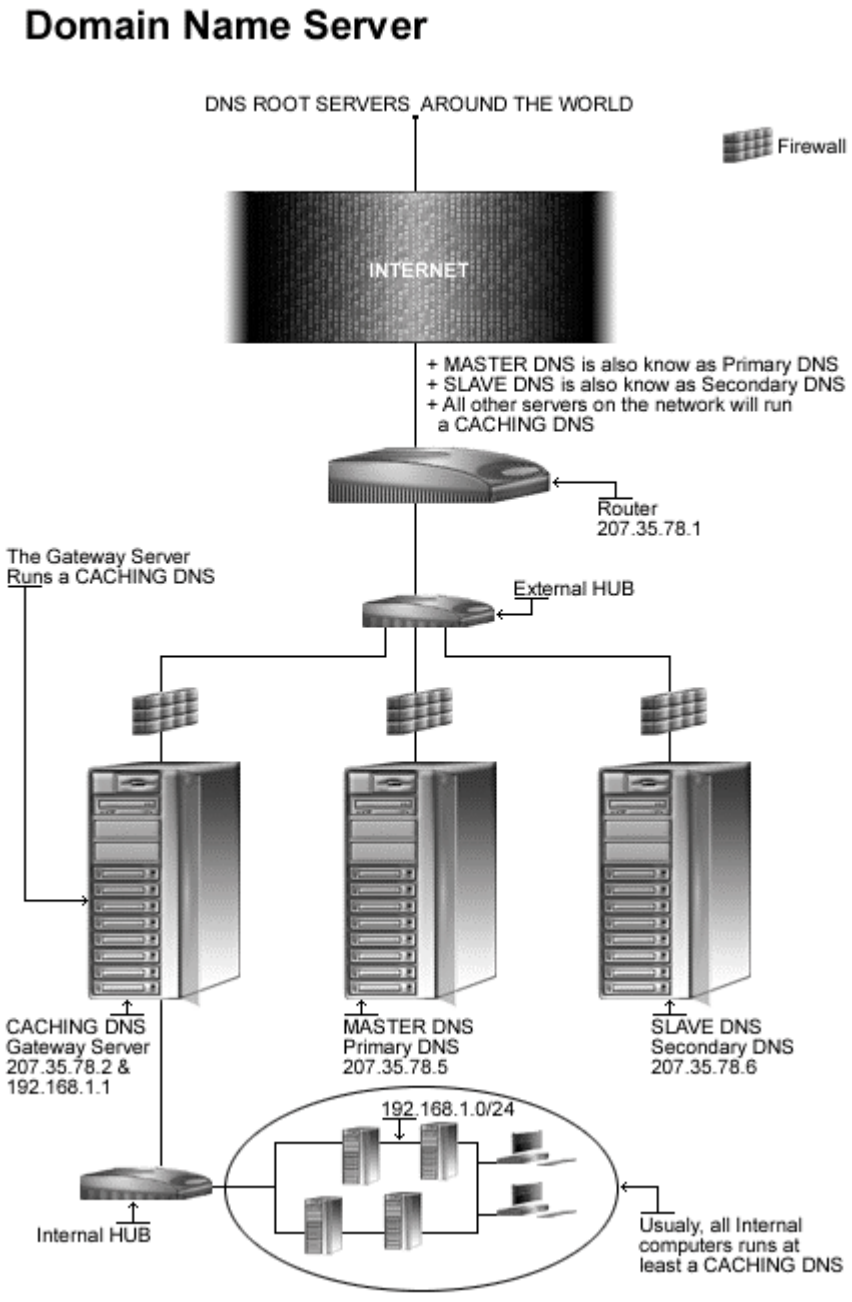

## Recommended RPM packages to be installed for a DNS Server

A minimal configuration provides the basic set of packages required by the Linux operating system. Minimal configuration is a perfect starting point for building secure operating system. Below is the list of all recommended RPM packages required to run properly your Linux server as a Primary/Master, Secondary/Slave or Caching-Only Domain Name Server (DNS).

This configuration assumes that your kernel is a monolithic kernel. Also I suppose that you will install ISC BIND & DNS by RPM package. Therefore, bind and bind-utils RPM packages are already included in the list below as you can see. All security tools are not installed, it is yours to install them as your need by RPM packages since compilers packages are not installed and included in the list.

| | | | | |
|---|---|---|---|---|
| basesystem | diffutils | initscripts | newt | shadow-utils |
| bash | e2fsprogs | iptables | pam | slang |
| bc | ed | kernel | passwd | slocate |
| bdflush | file | less | perl | sysklogd |
| bind | filesystem | libstdc++ | popt | SysVinit |
| bind-utils | fileutils | libtermcap | procps | tar |
| bzip2 | findutils | lilo | psmisc | termcap |
| chkconfig | gawk | logrotate | pwdb | textutils |
| console-tools | gdbm | losetup | qmail | tmpwatch |
| cpio | gilb | MAKEDEV | readline | utempter |
| cracklib | glibc | man | rootfiles | util-linux |
| cracklib-dicts | gmp | mingetty | rpm | vim-common |
| crontabs | gpm | mktemp | sash | vim-minimal |
| db1 | grep | mount | stat | vixie-cron |
| db2 | groff | ncompress | sed | words |
| db3 | gzip | ncurses | setup | which |
| dev | info | net-tools | sh-utils | zlib |

*Tested and fully functional on OpenNA.com.*




*This is a graphical representation of the DNS configuration we use in this book. We try to show you different settings (Caching Only DNS, Primary/Master DNS, and Secondary/Slave DNS) on different servers. Please note that lot possibilities exist, and depend of your needs, and network architecture design.*


## These installation instructions assume

Commands are Unix-compatible.
The source path `is /var/tmp` (note that other paths are possible, as personal discretion).
Installations were tested on Red Hat 7.0.
All steps in the installation will happen in super-user account "root".
Whether kernel recompilation may be required: No
Latest ICS BIND & DNS version number is `9.1.1rc3`

## Packages

The following are based on information as listed by ISC BIND & DNS as of 27/02/2001. Please regularly check at www.isc.org for the latest status.

Source code is available from:
ISC BIND & DNS Homepage: http://www.isc.org/
ISC BIND & DNS FTP Sire: 204.152.184.27
You must be sure to download: `bind-9.1.1rc3.tar.gz`

## Prerequisites

ICS BIND & DNS requires that the listed software below be already installed on your system to be able to compile successfully. If this is not the case, you must install it.

- ✓ To improve signing and verification speed of BIND9, OpenSSL library that uses hand-optimized assembly language routines should be already installed on your system.

**NOTE:** For more information on OpenSSL software, see its related chapter in this book.

## Pristine source

If you don't use RPM package to install this program, it will be difficult for you to locate all installed files into the system in the eventuality of an updated in the future. To solve the problem, it is a good idea to make a list of files on the system before you install ISC BIND & DNS, and one afterwards, and then compare them using the **`diff`** utility of Linux to find out what files are placed where.

- Simply run the following command before installing the software:
  ```
  [root@deep /root]#  find /* > DNS1
  ```
- And the following one after you install the software:
  ```
  [root@deep /root]#  find /* > DNS2
  ```
- Then use the following command to get a list of what changed:
  ```
  [root@deep /root]#  diff DNS1 DNS2 > ISC-BIND-DNS-Installed
  ```

With this procedure, if any upgrade appears, all you have to do is to read the generated list of what changed program and remove the files manually from your system before installing the new software. Related to our example above, we use the `/root` directory of the system to stock all generated list files.


## Compiling - Optimizing & Installing ISC BIND & DNS

Below are the require steps that you must absolutely make to compile and optimize the ISC BIND & DNS software before installing it into your Linux system.

Step 1
Once you get the program from the trusted main software site you must copy it to the /var/tmp directory and change to this location before expanding the archive.

- These procedures can be accomplished with the following commands:
```
[root@deep /]# cp bind-version.tar.gz /var/tmp/
[root@deep /]# cd /var/tmp/
[root@deep tmp]# tar xzpf bind-version.tar.gz
```

Step 2
In order to check that the version of ISC BIND & DNS, which you are going to install, is an original and unmodified one, please check the supplied signature with the PGP key of ISC BIND & DNS. Unfortunately, ISC BIND & DNS don't provide a MD5 signature for this verification. Only a PGP key is available on the ISC BIND & DNS website.

To get a PGP key copy of ISC BIND & DNS, please point your browser to the following URL: http://www.isc.org/. For more information about how to use this key for verification, see the GnuPG chapter in this book.

Step 3
ISC BIND & DNS cannot run as super-user root; for this reason we must create a special user with no shell privilege on the system for running ISC BIND & DNS daemon.

- To create this special ISC BIND & DNS user, use the following command:
```
[root@deep tmp]# useradd -c "Named" -u 25 -s /bin/false -r -d /var/named
named 2>/dev/null || :
```

The above command will create a null account, with no password, no valid shell, no files owned-nothing but a uid and a gid for the program.

Step 4
After that, move into the new expanded ISC BIND & DNS directory and perform the following steps before compiling and optimizing it. The modifications we bring to the ISC BIND & DNS source files below are necessary to relocate some default files as well as to fix a small bug with the software.

- To move into the new expanded ISC BIND & DNS directory, use the following command:
```
[root@deep tmp]# cd bind-9.1.1rc3/
```

Step 4.1
The first file that we must modify is called **dighost.c** located under the source directory of ISC BIND & DNS. Into this file, we will add a missing code line related to the reverse function of the program.


- Edit the **`dighost.c`** file (`vi +224 bin/dig/dighost.c`) and change the lines:

```
if (n == 0) {
        return (DNS_R_BADDOTTEDQUAD);
}
for (i = n - 1; i >= 0; i--) {
        snprintf(working, MXNAME/8, "%d.",
                 adrs[i]);
```

To read:

```
if (n == 0) {
        return (DNS_R_BADDOTTEDQUAD);
}
reverse[0] = 0;
for (i = n - 1; i >= 0; i--) {
        snprintf(working, MXNAME/8, "%d.",
                 adrs[i]);
```

Step 4.2
The second source file to modify is called **`globals.h`** and one of its functions is to specify the location of the `named.pid` and `lwresd.pid` files. We'll change the default location for these files to be compliant with our Linux operating system.

- Edit the **`globals.h`** file (`vi +101 bin/named/include/named/globals.h`) and change the lines:

```
"/run/named.pid");
```

To read:

```
"/run/named/named.pid");
```

and

```
"/run/lwresd.pid");
```

To read:

```
"/run/named/lwresd.pid");
```

Step 5
Once the required modifications have been made into the related source files of `ISC BIND & DNS` as explained previously, it is time compile and optimize it.

- To compile and optimize `ISC BIND & DNS` use the following commands:

```
CFLAGS="-O3 -funroll-loops -fomit-frame-pointer" \
./configure \
--prefix=/usr \
--sysconfdir=/etc \
--localstatedir=/var \
--mandir=/usr/share/man \
--with-openssl=/usr/include/openssl \
--with-libtool \
```


```
--disable-ipv6
```

**This tells ISC BIND & DNS to set itself up for this particular system with:**

- Build shared libraries.
- Use original OpenSSL rather than using bind-9 internal OpenSSL.
- Disable Ipv6 support.

Step 6
At this stage of our work the program is ready to be build and installed. We build ISC BIND & DNS with the 'make' command and produce a list of files on the system before we install the software, and one afterwards, then compare them using the **diff** utility to find out what files are placed where and finally install ISC BIND & DNS:

```
[root@deep bind-9.1.1rc3]# make
[root@deep bind-9.1.1rc3]# cd
[root@deep /root]# find /* > DNS1
[root@deep /root]# cd /var/tmp/bind-9.1.1rc3/
[root@deep bind-9.1.1rc3]# make install
[root@deep bind-9.1.1rc3]# cd doc/man/bin/
[root@deep bin]# install -c -m 600 named.8 /usr/share/man/man8/
[root@deep bin]# install -c -m 600 rndc.8 /usr/share/man/man8/
[root@deep bin]# install -c -m 600 lwresd.8 /usr/share/man/man8/
[root@deep bin]# install -c -m 600 nsupdate.8 /usr/share/man/man8/
[root@deep bin]# install -c -m 600 named-checkconf.1 /usr/share/man/man1/
[root@deep bin]# install -c -m 600 named-checkzone.1 /usr/share/man/man1/
[root@deep bin]# install -c -m 600 host.1 /usr/share/man/man1/
[root@deep bin]# install -c -m 600 dig.1 /usr/share/man/man1/
[root@deep bin]# install -c -m 600 rndc.conf.5 /usr/share/man/man5/
[root@deep bin]# cd ../../../
[root@deep bind-9.1.1rc3]# strip /usr/sbin/named
[root@deep bind-9.1.1rc3]# mkdir -p /var/named
[root@deep bind-9.1.1rc3]# mkdir -p /var/run/named
[root@deep bind-9.1.1rc3]# install -c -m 640 bin/rndc/rndc.conf /etc/
[root@deep bind-9.1.1rc3]# chown named.named /etc/rndc.conf
[root@deep bind-9.1.1rc3]# chown named.named /var/named/
[root@deep bind-9.1.1rc3]# chown named.named /var/run/named/
[root@deep bind-9.1.1rc3]# /sbin/ldconfig
[root@deep bind-9.1.1rc3]# cd
[root@deep /root]# find /* > DNS2
[root@deep /root]# diff DNS1 DNS2 > ISC-BIND-DNS-Installed
```

The above commands will configure the software to ensure your system has the necessary functionality and libraries to successfully compile the package, compile all source files into executable binaries, and then install the binaries and any supporting files into the appropriate locations.

Step 7
Once compilation, optimization and installation of the software have been finished, we can cleanup our workspace by deleting the program tar archive and the related source directory since there are no longer needed.

- To delete ISC BIND & DNS and its related source directory, use the following commands:

```
[root@deep /]# cd /var/tmp/
[root@deep tmp]# rm -rf bind-version/
```


```
[root@deep tmp]# rm -f bind-version.tar.gz
```

The `rm` command as used above will remove all the source files we have used to compile and install `ISC BIND & DNS`. It will also remove the `ISC BIND & DNS` compressed archive from the `/var/tmp` directory.

## Configuring `ISC BIND & DNS`

After `ISC BIND & DNS` has been built and installed successfully in your system, your next step is to configure and customize all the required parameters in your different `ISC BIND & DNS` configuration files as prudently as possible. Depending of the kind of Domain Name System you want to run in your Linux server, there is different configuration files to set up, those files are:

For running `ISC BIND & DNS` as a Caching-Only Name Server:

- ✓ `/etc/named.conf` (The `ISC BIND & DNS` Configuration File)
- ✓ `/var/named/db.127.0.0` (The `ISC BIND & DNS` reverse mapping File)
- ✓ `/var/named/db.cache` (The `ISC BIND & DNS` Root server hints File)
- ✓ `/etc/logrotate.d/named` (The `ISC BIND & DNS` Log rotation File)
- ✓ `/etc/sysconfig/named` (The `ISC BIND & DNS` System Configuration File)
- ✓ `/etc/rc.d/init.d/named` (The `ISC BIND & DNS` Initialization File)

For running `ISC BIND & DNS` as a Master/Primary Name Server:

- ✓ `/etc/named.conf` (The `ISC BIND & DNS` Configuration File)
- ✓ `/var/named/db.127.0.0` (The `ISC BIND & DNS` reverse mapping File)
- ✓ `/var/named/db.cache` (The `ISC BIND & DNS` Root server hints File)
- ✓ `/var/named/db.207.35.78` (The `ISC BIND & DNS` host names to addr mapping File)
- ✓ `/var/named/db.openna` (The `ISC BIND & DNS` addr to host names mapping File)
- ✓ `/etc/logrotate.d/named` (The `ISC BIND & DNS` Log rotation File)
- ✓ `/etc/sysconfig/named` (The `ISC BIND & DNS` System Configuration File)
- ✓ `/etc/rc.d/init.d/named` (The `ISC BIND & DNS` Initialization File)

For running `ISC BIND & DNS` as a Slave/Secondary Name Server:

- ✓ `/etc/named.conf` (The `ISC BIND & DNS` Configuration File)
- ✓ `/var/named/db.127.0.0` (The `ISC BIND & DNS` reverse mapping File)
- ✓ `/var/named/db.cache` (The `ISC BIND & DNS` Root server hints File)
- ✓ `/etc/logrotate.d/named` (The `ISC BIND & DNS` Log rotation File)
- ✓ `/etc/sysconfig/named` (The `ISC BIND & DNS` System Configuration File)
- ✓ `/etc/rc.d/init.d/named` (The `ISC BIND & DNS` Initialization File)

**WARNING:** It is important to note that some of the configuration files as shown above are the same for all type of Domain Name System and for this reason, files that are common for all configuration are described after all specific Domain Name System configurations. Please read all information contained in this chapter to be sure to not forget something.


## Caching-Only Name Server

This section applies only if you chose to install and use ISC BIND & DNS as a Caching Name Server in your system. Caching-only name servers are servers not authoritative for any domains except 0.0.127.in-addr.arpa (the localhost). A Caching-Only Name Server can look up names inside and outside your zone, as can Primary and Slave Name Servers. The difference is that when a Caching-Only Name Server initially looks up a name within your zone, it ends up asking one of the Primary or Slave Names Servers for your zone for the answer.

## /etc/named.conf: The ISC BIND & DNS Configuration File

Use this configuration file for all servers on your network that don't act as a Master or Slave Name Server. Setting up a simple Caching Server for local client machines will reduce the load on the network's primary server.

Many users on dialup connections may use this configuration along with ISC BIND & DNS for such a purpose. With this configuration for a Caching-Only Name Server, all queries from outside clients are refused. Each texts in bold are part of the configuration file that must be customized and adjusted to satisfy our needs.

- Create the **named.conf** file (touch /etc/named.conf) and add the following lines in the file. Below is what we recommend you:

```
options {
        directory "/var/named";
        allow-transfer { none; };
        allow-query { 192.168.1.0/24; localhost; };
        allow-recursion { 192.168.1.0/24; localhost; };
        forwarders { 207.35.78.5; 207.35.78.6; };
        forward only;
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};
```

**This tells named.conf file to set itself up for this particular configuration with:**

Options {};
The options statement sets up global options to be used by ISC BIND & DNS and may appear only once in a configuration file.

directory "/var/named";


The `directory` statement indicates the working directory of the server and should be an absolute path. The working directory is where all configuration files related to `ISC BIND & DNS` resides.

`allow-transfer { none; };`
The `allow-transfer` statement specifies which hosts are allowed to receive zone transfers from the Primary/Master Name Server. The default setting of `ISC BIND & DNS` is to allow transfers from all hosts. Since zone transfers requests is only required for Secondary/Slave Name Server and since the configuration we are trying to do here is for a Caching-Only Name Server, we can disable completely this directive with the parameter "`allow-transfer { none; };`". This is a security feature.

`allow-query { 192.168.1.0/24; localhost; };`
The `allow-query` statement specifies which hosts are allowed to ask ordinary questions to the Caching Name Server. The default setting in the `options` block is to allow queries from all hosts. In our configuration, we wish to allow queries from one corporate subnet only. This is a security feature.

`allow-recursion { 192.168.1.0/24; localhost; };`
The `allow-recursion` statement specifies which hosts are allowed to make recursive queries through this server. With the configuration as shown above, we allow recursive queries only from internal hosts since allowing every external hosts on the Internet to ask your name server to answer recursive queries can open you up to certain kinds of cache poisoning attacks. This is a security feature.

`forwarders { 207.35.78.5; 207.35.78.6; };`
The `forwarders` statement specifies the IP addresses to be used for forwarding. Servers that do not have direct access to the Internet use this option to create a large site-wide cache, reducing traffic over links to external name servers and to allow queries. It occurs only on those queries for which the server is not authoritative and does not have the answer in its cache. In the "`forwarders`" line, `207.35.78.5` and `207.35.78.6` are the IP addresses of the Primary (Master) and Secondary (Slave) DNS servers. They can also be the IP addresses of your ISP's DNS server and another DNS server, respectively. This is a security feature.

`forward only;`
To improve the security of your `ISC BIND & DNS` server you can stop it from even trying to contact an off-site server if their forwarder is down or doesn't respond. With the "`forward only`" option set in your `named.conf` file, the name server doesn't try to contact other servers to find out information if the forwarder doesn't give it an answer. If the `only` option is specified, the server will only query the IP addresses as specified in the `forwarders` statement and not other servers. This is a security feature.

`version "Go away!";`
The `version` statement allows us to hide the real version number of our `ISC BIND & DNS` server. This can be useful when some one from the Internet try to scan our Domain Name Server for possible vulnerable version of the software. You can change the string "`Go away!`" to whatever you want. Note doing this will not prevent attacks and may impede people trying to diagnose problems with your server. This is a security feature.

`notify no;`
`DNS` Notify is a mechanism that allows Master Name Servers to notify their Slave servers of changes to a zone's data. In response to a NOTIFY from a Master server, the Slave will check to


see that its version of the zone is the current version and, if not, initiate a transfer. The `notify` statement by default is set to "`yes`" but since the `loopback address 127.0.0.1` is the same to each system, we must avoid to transfer this localhost configuration file to Secondary/Slave Name Server.

**NOTE:** You can configure logging so that lame server messages aren't logged, which will reduce the overhead on your DNS and `syslog` servers. Lame server messages are report hosts that are believed to be name servers for the given domains, but which do not believe themselves to be such. This is often due to a configuration error on the part of that hostmaster.

You can disable "Lame server" messages by using the logging statement into your `named.conf` file:

```
logging {
        category lame-servers { null; };
};
```

## `/var/named/db.127.0.0`: The reverse mapping File

Use this configuration file for all servers on your network that don't act as a Master or Slave Name Server. The "`db.127.0.0`" file covers the loopback network by providing a reverse mapping for the loopback address on your system. Create the following file in `/var/named`.

- Create the **`db.127.0.0`** file (`touch /var/named/db.127.0.0`) and add the following lines in the file:

```
; Revision History: March 01, 2001 - root@openna.com
; Start of Authority (SOA) records.
$TTL 86400
@       IN      SOA     localhost.      root.localhost. (
                                        00      ; Serial
                                        10800   ; Refresh after 3 hours
                                        3600    ; Retry after 1 hour
                                        604800  ; Expire after 1 week
                                        86400 ) ; Minimum

        IN      NS      localhost.
1       IN      PTR     localhost.
```


## Primary Master Name Server

This section applies only if you chose to install and use ISC BIND & DNS as a Primary Name Server in your system. The Primary Master Server is the ultimate source of information about a domain. The Primary Master is an authoritative server configured to be the source of zone transfer for one or more Secondary servers. The Primary Master Server obtains data for the zone from a file on disk.

## /etc/named.conf: The ISC BIND & DNS Configuration File

Use this configuration for the server on your network that acts as a Master Name Server. In every respectable networking environment, you need to set up at least a Primary Domain Name Server for your network. We'll use "openna.com" as an example domain, and assume you are using IP network address of 207.35.78.0.

To do this, add the following lines to your /etc/named.conf file. Each texts in bold are part of the configuration file that must be customized and adjusted to satisfy our needs.

- Create the **named.conf** file (touch /etc/named.conf) and add the following lines in the file. Below is what we recommend you:

```
options {
        directory "/var/named";
        allow-transfer { 207.35.78.6; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are the master server for openna.com
zone "openna.com" {
        type master;
        file "db.openna";
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type master;
        file "db.207.35.78";
        allow-query { any; };
};
```


**This tells `named.conf` file to set itself up for this particular configuration with:**

`Options {};`
The `options` statement sets up global options to be used by `ISC BIND & DNS` and may appear only once in a configuration file.

`directory "/var/named";`
The `directory` statement indicates the working directory of the server and should be an absolute path. The working directory is where all configuration files related to `ISC BIND & DNS` resides.

`allow-transfer { 207.35.78.6; };`
The `allow-transfer` statement specifies which hosts are allowed to receive zone transfers from the Primary/Master Name Server. The default setting of `ISC BIND & DNS` is to allow transfers from all hosts. In the `allow-transfer` line as shown above, `207.35.78.6` (our Secondary/Slave Name Server) is the only IP address allowed to receive zone transfers from the Primary/Master Name Server. You should configure your server to respond to zone transfers requests only from authorized IP addresses. In most cases, you'll only authorize your known Slave servers to transfer zones from your Primary/Master Name Server. As the information provided is often used by spammers and IP spoofers. This is a security feature.

`allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };`
The `allow-query` statement specifies which hosts are allowed to ask ordinary questions to the Primary Name Server. The default setting in the `options` block is to allow queries from all hosts. In our configuration, we wish to allow queries from our subnets (`192.168.1.0/24`, `207.35.78.0/32`, and `localhost`). With this restriction, everyone form the Internet can query us for the zones that we administer and its reverse, but only internal hosts that we have specified in the "`allow-query`" statement can make other queries. Take a note that we add the "`allow-query { any; };`" option in each zone statement into the `named.conf` file to make effective this protection. This is a security feature.

`allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };`
The `allow-recursion` statement specifies which hosts are allowed to make recursive queries through this server. With the configuration as shown above, we allow recursive queries only from internal hosts since allowing every external hosts on the Internet (external hosts will have their own name servers) to ask your name server to answer recursive queries can open you up to certain kinds of cache poisoning attacks. This is a security feature.

`version "Go away!";`
The `version` statement allows us to hide the real version number of our `ISC BIND & DNS` server. This can be useful when some one from the Internet try to scan our Domain Name Server for possible vulnerable version of the software. You can change the string "`Go away!`" to whatever you want. Note doing this will not prevent attacks and may impede people trying to diagnose problems with your server. This is a security feature.

`notify no;`
DNS Notify is a mechanism that allows Master Name Servers to notify their Slave servers of changes to a zone's data. In response to a NOTIFY from a Master Server, the Slave will check to see that its version of the zone is the current version and, if not, initiate a transfer. The `notify` statement by default is set to "`yes`" but since the `loopback address 127.0.0.1` is the same to each system, we must avoid to transfer this `localhost` configuration file to Secondary/Slave Name Server.


**NOTE:** You can configure logging so that lame server messages aren't logged, which will reduce the overhead on your DNS and `syslog` servers. Lame server messages are report hosts that are believed to be name servers for the given domains, but which do not believe themselves to be such. This is often due to a configuration error on the part of that hostmaster.

You can disable "Lame server" messages by using the logging statement into your `named.conf` file:

```
logging {
        category lame-servers { null; };
};
```

## `/var/named/db.127.0.0`: The reverse mapping File

Use this configuration file for the server on your network that acts as a Master Name Server. The "`db.127.0.0`" file covers the loopback network by providing a reverse mapping for the loopback address on your system. Create the following file in `/var/named`.

- Create the **`db.127.0.0`** file (`touch /var/named/db.127.0.0`) and add the following lines in the file:

```
; Revision History: March 01, 2001 - root@openna.com
; Start of Authority (SOA) records.
$TTL 86400
@  IN  SOA  ns1.openna.com. root.openna.com.  (
                                        00      ; Serial
                                        10800   ; Refresh after 3 hours
                                        3600    ; Retry after 1 hour
                                        604800  ; Expire after 1 week
                                        86400 ) ; Minimum TTL of 1 day

; Name Server (NS) records.
        IN      NS      ns1.openna.com.
        IN      NS      ns2.openna.com.

; only One PTR record.
1       PTR     localhost.
```

## `/var/named/db.207.35.78`: The host names to addresses mapping File

Use this configuration file for the server on your network that acts as a Master Name Server. The "`db.207.35.78`" file maps host names to addresses. Create the following file in `/var/named`.

- Create the **`db.207.35.78`** file (`touch /var/named/db.207.35.78`) and add the following lines in the file:

```
; Revision History: March 01, 2001 - root@openna.com
; Start of Authority (SOA) records.
$TTL 86400
@  IN  SOA  ns1.openna.com. root.openna.com.  (
                                        00      ; Serial
                                        10800   ; Refresh after 3 hours
```


```
                                        3600    ; Retry after 1 hour
                                        604800  ; Expire after 1 week
                                        86400 ) ; Minimum TTL of 1 day

; Name Server (NS) records.
        IN      NS      ns1.openna.com.
        IN      NS      ns2.openna.com.

; Addresses Point to Canonical Names (PTR) for Reverse lookups
1       IN      PTR     router.openna.com.
2       IN      PTR     portal.openna.com.
3       IN      PTR     www.openna.com.
4       IN      PTR     smtp.openna.com.
```

## `/var/named/db.openna`: The addresses to host names mapping File

Use this configuration file for the server on your network that acts as a Master Name Server. The "`db.openna`" file maps addresses to host names. Create the following file in `/var/named`.

- Create the **`db.openna`** file (`touch /var/named/db.openna`) and add the following lines in the file:

```
; Revision History: March 01, 2001 - root@openna.com
; Start of Authority (SOA) records.
$TTL 86400
@  IN  SOA  ns1.openna.com. root.openna.com.  (
                                        00      ; Serial
                                        10800   ; Refresh after 3 hours
                                        3600    ; Retry after 1 hour
                                        604800  ; Expire after 1 week
                                        86400 ) ; Minimum TTL of 1 day

; Name Server (NS) records.
        IN      NS      ns1.openna.com.
        IN      NS      ns2.openna.com.

; Mail Exchange (MX) records.
        MX      0       smtp.openna.com.

; Address (A) records.
localhost       IN      A       127.0.0.1
router          IN      A       207.35.78.1
portal          IN      A       207.35.78.2
www             IN      A       207.35.78.3
smtp            IN      A       207.35.78.4
ns1             IN      A       207.35.78.5
ns2             IN      A       207.35.78.6
```


## Secondary Slave Name Server

This section applies only if you chose to install and use ISC BIND & DNS as a Secondary Name Server in your system. The purpose of a Slave Name Server is to share the load with the Master Name Server, or handle the entire load if the Master Name Server is down. A Slave Name Server, which is an authoritative server, loads its data over the network from another Name Server (usually the Master Name Server, but it can load from another Slave Name Server too). This process is called a zone transfer. Slave servers provide necessary redundancy on the network.

## /etc/named.conf: The ISC BIND & DNS Configuration File

Use this configuration for the server on your network that acts as a Slave Name Server. You must modify the "named.conf" file on the Slave Name Server host. Change every occurrence of primary to secondary except for "0.0.127.in-addr.arpa" and add a masters line with the IP address of the Master Server as shown below. Each texts in bold are part of the configuration file that must be customized and adjusted to satisfy our needs.

- Create the **named.conf** file (touch /etc/named.conf) and add the following lines in the file. Below is what we recommend you:

```
options {
        directory "/var/named";
        allow-transfer { none; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are a slave server for openna.com
zone "openna.com" {
        type slave;
        file "db.openna";
        masters { 207.35.78.5; };
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type slave;
        file "db.207.35.78";
        masters { 207.35.78.5; };
        allow-query { any; };
};
```


The above `named.conf` file tells the Secondary Name Server that it is a Slave Server for the zone "`openna.com`" and should track the version of this zone that is being kept on the host "`207.35.78.5`", which is the Master Name Server in the network.

**NOTE:** A Slave Name Server doesn't need to retrieve all of its database (`db`) files over the network because these `db` files "`db.127.0.0`" and "`db.cache`" are the same as on a Primary Master, so you can keep a local copy of these files on the Slave Name Server.

You can configure logging so that lame server messages aren't logged, which will reduce the overhead on your `DNS` and `syslog` servers. Lame server messages are report hosts that are believed to be name servers for the given domains, but which do not believe themselves to be such. This is often due to a configuration error on the part of that hostmaster.

You can disable "Lame server" messages by using the logging statement into your `named.conf` file:

```
logging {
        category lame-servers { null; };
};
```

## `/var/named/db.127.0.0`: The reverse mapping File

Use this configuration file for the server on your network that acts as a Slave Name Server. The "`db.127.0.0`" file covers the loopback network by providing a reverse mapping for the loopback address on your system. Create the following file in `/var/named`.

- Create the **`db.127.0.0`** file (`touch /var/named/db.127.0.0`) and add the following lines in the file:

```
; Revision History: March 01, 2001 - root@openna.com
; Start of Authority (SOA) records.
$TTL 86400
@  IN  SOA  ns1.openna.com. root.openna.com.  (
                                        00      ; Serial
                                        10800   ; Refresh after 3 hours
                                        3600    ; Retry after 1 hour
                                        604800  ; Expire after 1 week
                                        86400 ) ; Minimum TTL of 1 day

; Name Server (NS) records.
        IN      NS      ns1.openna.com.
        IN      NS      ns2.openna.com.

; only One PTR record.
1       PTR     localhost.
```

## `/var/named/db.cache`: The Root server hints File

This section applies for all type of Name Server (Caching, Master or Slave) that you may want to install in your system. The **`db.cache`** file is also know as the "Root server hints file" and tells


your server (Caching, Master or Slave) where the servers for the "root" zone are, you must get a copy of **db.cache** file and copy this file into the /var/named directory.

Step 1
Use the following commands on another Unix computer in your organization to query a new **db.cache** file for your Name Servers or pick one from your Linux CD-ROM source distribution:

- To query a new db.cache file, use the following command:
  **[root@deep]# dig @a.root-servers.net . ns > db.cache**

- To query a new db.cache file by IP address, use the following command:
  **[root@deep]# dig @198.41.0.4 . ns > db.cache**

**NOTE:** The root name servers do not change very often, but they do change. A good practice is to update your db.cache file every month or two.

Step 2
Don't forget to copy the db.cache file to the /var/named directory on your Name Server after retrieving it over the Internet.

**WARNING:** Internal addresses like 192.168.1/24 are not included in the DNS configuration files for security reasons. Once again, it is very important that DNS doesn't exist between hosts on the corporate network and external hosts. You have been warned.

## /etc/logrotate.d/named: The ISC BIND & DNS Log rotation File

This section applies for all type of Name Server (Caching, Master or Slave) that you may want to install in your system. This file allows the Domain Name Server to automatically rotate its log files each week. Configure your /etc/logrotate.d/named file to rotate each week your ISC BIND & DNS log files automatically.

- Create the **named** file (touch /etc/logrotate.d/named) and add the lines:

```
/var/log/named.log {
    missingok
    postrotate
    /bin/kill -HUP `cat /var/run/named.pid 2> /dev/null` 2> /dev/null ||
true
    endscript
}
```

## /etc/sysconfig/named: The ISC BIND & DNS System Configuration File

This section applies for all type of Name Server (Caching, Master or Slave) that you may want to install in your system. The /etc/sysconfig/named file is used to specify ISC BIND & DNS system configuration information, such as if ISC BIND & DNS should run in a chroot environment, and if additional options are required to be passed to named daemon at startup.


- Create the **named** file (`touch /etc/sysconfig/named`) and add the following lines:

```
# Currently, you can use the following options:
#ROOTDIR=""
#OPTIONS=""
```

The "`ROOTDIR=""`" option instructs `ISC BIND & DNS` where its root directory should be located, this line is useful when you want to run `ISC BIND & DNS` in an chroot jail environment for more security. From now, this line must be commented out since we'll see later in this chapter how to run `ISC BIND & DNS` in a chroot environment and how to use this option.

As usually with many daemons under Unix, we can add special options to the command line before starting the daemons. With the new system V feature of Linux most of command line options can now be specified in config files like the above. The "`OPTIONS=""`" parameter in the `/etc/sysconfig/named` file is for this such use for `ISC BIND & DNS`. We can for example add the "`-d`" option for debug level of `ISC BIND & DNS` but in many case we don't need to use it.

## `/etc/rc.d/init.d/named`: The `ISC BIND & DNS` Initialization File

This section applies for all type of Name Server (Caching, Master or Slave) that you may want to install in your system. The `/etc/rc.d/init.d/named` script file is responsible to automatically start and stop the `ISC BIND & DNS` daemon on your server. Loading the `named` daemon, as a standalone daemon will eliminate load time and will even reduce swapping since non-library code will be shared.

Step 1
Create the **named** script file (`touch /etc/rc.d/init.d/named`) and add the following lines inside it:

```
#!/bin/bash
#
# named           This shell script takes care of starting and stopping
#                 named (BIND DNS server).
#
# chkconfig: - 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 /etc/sysconfig/named ] && . /etc/sysconfig/named

[ -f /usr/sbin/named ] || exit 0

[ -f "${ROOTDIR}"/etc/named.conf ] || exit 0

RETVAL=0
```


```
start() {
        # Start daemons.
        echo -n "Starting named: "
        if [ -n "${ROOTDIR}" -a "x${ROOTDIR}" != "x/" ]; then
                OPTIONS="${OPTIONS} -t ${ROOTDIR}"
        fi
        daemon named -u named ${OPTIONS}
        RETVAL=$?
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/named
        echo
        return $RETVAL
}
stop() {
        # Stop daemons.
        echo -n "Shutting down named: "
        killproc named
        RETVAL=$?
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/named
        echo
        return $RETVAL
}
rhstatus() {
        /usr/sbin/rndc status
        return $?
}
restart() {
        stop
        start
}
reload() {
        /usr/sbin/rndc reload
        return $?
}
probe() {
        /usr/sbin/rndc reload >/dev/null 2>&1 || echo start
        return $?
}

# See how we were called.
case "$1" in
        start)
                start
                ;;
        stop)
                stop
                ;;
        status)
                rhstatus
                ;;
    restart)
        restart
        ;;
    condrestart)
        [ -f /var/lock/subsys/named ] && restart
        ;;
    reload)
        reload
        ;;
    probe)
```


```
            probe
            ;;
      *)
            echo "Usage: named {start|stop|status|restart|condrestart|reload|probe}"
            exit 1
esac

exit $?
```

Step 2
Once the named script file has been created, it is important to make it executable, change its default permissions, create the necessary links and start it. Making this file executable will allow the system to run it, changing its default permission is to allow only the root user to change this file for security reason, and creation of the symbolic links will let the process control initialization of Linux which is in charge of starting all the normal and authorized processes that need to run at boot time on your system to start the program automatically for you at each reboot.

- To make this script executable and to change its default permissions, use the commands:
```
[root@deep /]# chmod 700 /etc/rc.d/init.d/named
[root@deep /]# chown 0.0 /etc/rc.d/init.d/named
```

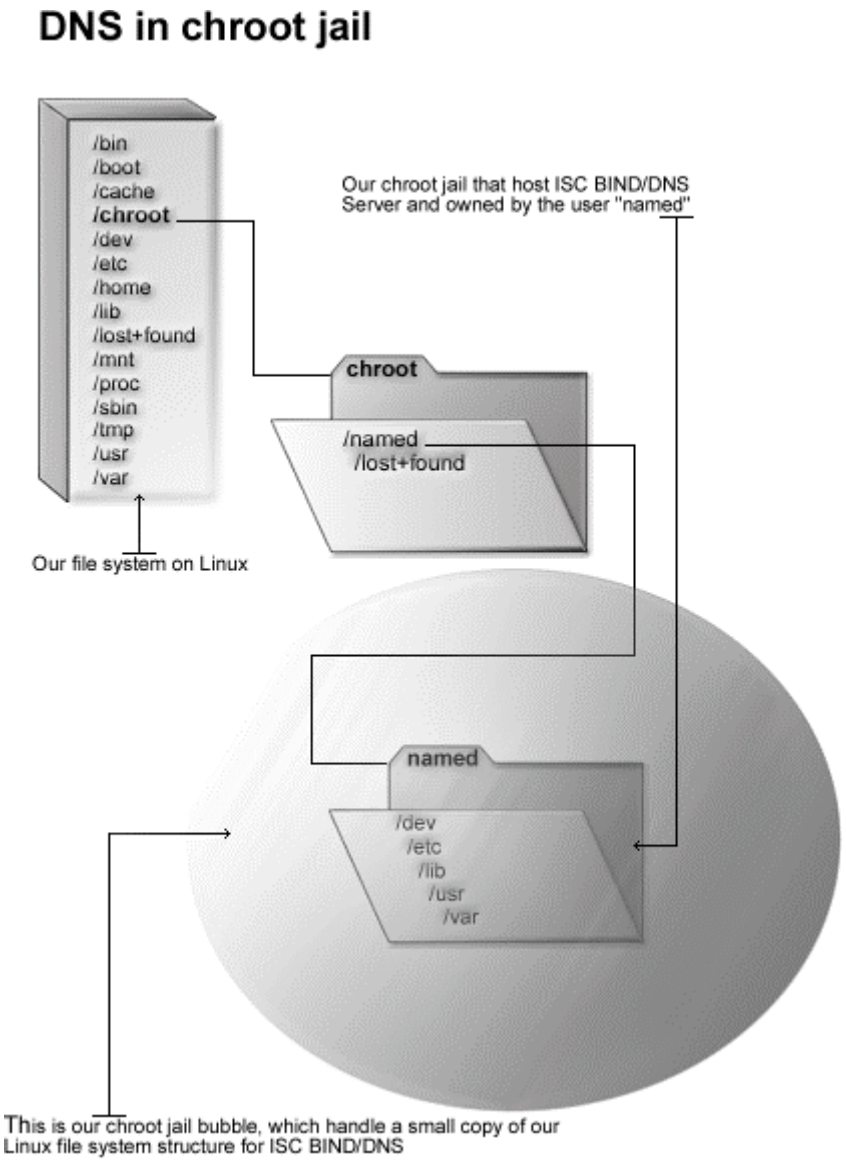
- To create the symbolic rc.d links for ISC BIND & DNS, use the following commands:
```
[root@deep /]# chkconfig --add named
[root@deep /]# chkconfig --level 2345 named on
```

- To start ISC BIND & DNS software manually, use the following command:
```
[root@deep /]# /etc/rc.d/init.d/named start
Starting named:                 [OK]
```

**NOTE:** All the configuration files required for each software described in this book has been provided by us as a gzipped file, floppy.tgz for your convenience. This can be downloaded from this web address: http://www.openna.com/books/floppy.tgz. You can unpack this to any location on your local machine, say for example /var/tmp, assuming you have done this your directory structure will be /var/tmp/floppy. Within this floppy directory each configuration file has its own directory for respective software. You can either cut and paste this directly if you are faithfully following our instructions from the beginning or manually edit these to modify to your needs. This facility is there though as a convenience but please don't forget ultimately it will be your responsibility to check, verify, etc. before you use them whether modified or as it is.

## Running ISC BIND & DNS in a chroot jail

This part focuses on preventing ISC BIND & DNS from being used as a point of break-in to the system hosting it. Since ISC BIND & DNS performs a relatively large and complex function, the potential for bugs that affect security is rather high with this software. In fact, there have been exploitable bugs in the past that allowed a remote attacker to obtain root access to hosts running ISC BIND & DNS.




To minimize this risk, ISC BIND & DNS can be run **as a non-root user**, which will limit any damage to what can be done as a normal user with a local shell. Of course, this is not enough for the security requirements of most DNS servers, so an additional step can be taken - that is, **running ISC BIND & DNS in a chroot jail.**


The main benefit of a chroot jail is that the jail will limit the portion of the file system the DNS daemon program can see to the root directory of the jail. Additionally, since the jail only needs to support DNS, the programs related to ISC BIND & DNS available in the jail can be extremely limited. Most importantly, there is no need for setuid-root programs, which can be used to gain root access and break out of the jail.

## Necessary steps to run ISC BIND & DNS software in a chroot jail:

What you're essentially doing is creating a skeleton root file system with enough components necessary (directories, files, etc.) to allow Unix to do a chroot when the ISC BIND & DNS daemon start. Contrarily to its predecessor (Bind8), Bind9 is far more easily to setup in a chroot jail environment. Now there is no need to copy shared library dependencies of named binary as well as binaries programs to the jail. All you have to do is to copy its configuration file with its zone files and instruct its daemon process to chroot to the appropriated chroot directory before starting.

Step 1
The first step to do for running ISC BIND & DNS in a chroot jail will be to set up the chroot environment, and create the root directory of the jail. We've chosen /chroot/named for this purpose because we want to put this on its own separate file system to prevent file system attacks. Early in our Linux installation procedure we created a special partition /chroot for this exact purpose.

```
[root@deep /]# /etc/rc.d/init.d/named stop ← Only if named daemon already run.
Shutting down named:              [OK]

[root@deep /]# mkdir -p /chroot/named
[root@deep /]# mkdir -p /chroot/named/etc
[root@deep /]# mkdir -p /chroot/named/var/run/named
[root@deep /]# mkdir -p /chroot/named/var/named
[root@deep /]# chown -R named.named /chroot/named/var/run/named/
[root@deep /]# chown -R named.named /chroot/named/var/named/
```

We need all of the above directories because, from the point of the chroot, we're sitting at "/" and anything above this directory is inaccessible.

**WARNING:** The owner of the /chroot/named/var/named directory and all files into this directory must be owned by the process called "named".

Step 2
After that, we must move the main configuration files of ISC BIND & DNS into the appropriate places in the chroot jail. This includes the named.conf file and all zone files.

```
[root@deep /]# mv /etc/named.conf /chroot/named/etc/
[root@deep /]# cd /var/named; mv * /chroot/named/var/named/
[root@deep /]# chown named.named /chroot/named/etc/named.conf
[root@deep /]# chown -R named.named /chroot/named/var/named/*
```


Step 3
You will also need the **`/etc/localtime`** file in your chroot jail structure so that log entries are adjusted for your local time zone properly.

```
[root@deep /]# cp /etc/localtime /chroot/named/etc/
```

Step 4
Now we must set the `named.conf` file in the chroot jail directory immutable for better security.

- This procedure can be accomplished with the following commands:
  ```
  [root@deep /]# cd /chroot/named/etc/
  [root@deep etc]# chattr +i named.conf
  ```

**WARNING:** Don't forget to remove the immutable bit on these files if you have some modifications to bring to them with the command "`chattr -i`".

Step 5
Once the required files to run `ISC BIND & DNS` in the chroot jail environment have been relocated, we can remove the unnecessary directories related to `ISC BIND & DNS` from the system since the ones we'll work with now on a daily basis are located under the chroot directory. These directories are `/var/named` and `/var/run/named`.

```
[root@deep /]# rm -rf /var/named/
[root@deep /]# rm -rf /var/run/named/
```

Step 6
After that, it is time to instruct `ISC BIND & DNS` to start in the chroot jail environment. The `/etc/sysconfig/named` file is used for this purpose.

- Edit the **`named`** file (`vi /etc/sysconfig/named`) and change the following lines:

  ```
  # Currently, you can use the following options:
  #ROOTDIR=""
  #OPTIONS=""
  ```

  To read:

  ```
  # Currently, you can use the following options:
  ROOTDIR="/chroot/named/"
  ```

The "`ROOTDIR="/chroot/named/"`" option instructs `ISC BIND & DNS` where the chroot directory is located. Therefore the `named` daemon read this line in the `/etc/sysconfig/named` file and chroot to the specified directory before starting.

Step 7
Finally, we must test the new chrooted jail configuration of our `ISC BIND & DNS` server.


- Start the new chrooted jail ISC BIND & DNS with the following command:
```
[root@deep /]# /etc/rc.d/init.d/named start
Starting named:                            [OK]
```

- If you don't get any errors, do a **ps ax | grep named** and see if we're running:
```
[root@deep /]# ps ax | grep named
4278 ?        S      0:00 named -u named -t /chroot/named/
4279 ?        S      0:00 named -u named -t /chroot/named/
4280 ?        S      0:00 named -u named -t /chroot/named/
4281 ?        S      0:00 named -u named -t /chroot/named/
4282 ?        S      0:00 named -u named -t /chroot/named/
```

If so, lets check to make sure it's chrooted by picking out one of its process numbers and doing **ls -la /proc/that_process_number/root/**.

```
[root@deep /]# ls -la /proc/4278/root/
```

If you see something like:
```
total 4
drwxrwxr-x    4 root     root         1024 Feb 22 16:23 .
drwxr-xr-x    4 root     root         1024 Feb 22 14:33 ..
drwxrwxr-x    2 root     root         1024 Feb 22 15:52 etc
drwxrwxr-x    4 root     root         1024 Feb 22 14:34 var
```

Congratulations! Your ISC BIND & DNS in chroot jail is working.

## Securing ISC BIND & DNS

This section deals especially with actions we can make to improve and tighten security under ISC BIND & DNS. The interesting points here are that we refer to the features available within the base installed program and not to any new additional software.

### TSIG based transaction security with BIND

The new BIND9 allow us to create transaction keys and use **T**ransaction **SIG**natures (TSIG) with ISC BIND & DNS (TSIG is used for signed DNS requests). This means that if the server receives a message signed by this key, it can verify the signature. If the signature succeeds, the same key signs the response. This new feature of BIND will allow us to have a better control about which can make a zone transfer, notify, and recursive query messages on the DNS server. It might be most useful for dynamic update too. Below, we show you all the required steps to generate this key and how to use it in your named.conf file.

Step 1
The first step will be to generate shared keys for each pair of hosts. This shared secret will be shared between Primary Domain Name Server and Secondary Domain Name Server and an arbitrary key name must be chosen like in our example "ns1-ns2". It is also important that the key name be the same on both hosts.

- To generate shared keys, use the following command:
```
[root@deep /]# dnssec-keygen -a hmac-md5 -b 128 -n HOST ns1-ns2
Kns1-ns2.+157+49406
```


Step 2
The above command will generate a 128 bit (16 byte) HMAC-MD5 key and the result will be in a file called "Kns1-ns2.+157+49406.private" with a base-64 encoded string following the word "Key:", which must be extracted from the file and used as a shared secret.

- Edit the **Kns1-ns2.+157+49406.private** file (vi Kns1-ns2.+157+49406.private), and extract the base-64 encoded string:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: ps1jy3f7czVa1VNZkYaLfw==
```

The string "ps1jy3f7czVa1VNZkYaLfw==" in the above example is the part of this file that must be extracted and used as the shared secret.

Step 3
Once the required base-64 encoded string has been extracted from the generated file, we can remove the files from our system and copy the shared secret to both machines via a secure transport mechanism like ssh, telephone, etc.

- To remove the generated files from the system, use the following commands:

```
[root@deep /]# rm -f Kns1-ns2.+157+49406.key
[root@deep /]# rm -f Kns1-ns2.+157+49406.private
```

Step 4
After that, it is time to inform the servers (Primary & Secondary) of the Key's existence by adding to each server's named.conf file the following parameters.

- Edit the **named.conf** file (vi /chroot/named/etc/named.conf) on both DNS servers, and add the following lines:

```
key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};
```

Once the above lines have been added, your named.conf file on both DNS servers (Primary & Secondary) should look something like:

For Primary/Master server:

```
options {
        directory "/var/named";
        allow-transfer { 207.35.78.6; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
```


```
};

key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are the master server for openna.com
zone "openna.com" {
        type master;
        file "db.openna";
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type master;
        file "db.207.35.78";
        allow-query { any; };
};
```

For Secondary/Slave server:

```
options {
        directory "/var/named";
        allow-transfer { none; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};
```


```
        // We are a slave server for openna.com
        zone "openna.com" {
                type slave;
                file "db.openna";
                masters { 207.35.78.5; };
                allow-query { any; };
        };

        zone "78.35.207.in-addr.arpa" {
                type slave;
                file "db.207.35.78";
                masters { 207.35.78.5; };
                allow-query { any; };
};
```

Step 5
One of the last steps is to instruct the both servers (Primary & Secondary) to Use the Key. The servers must be told when keys are to be used. Adding another parameter into the `named.conf` file on both DNS servers does this. Into this parameter, on `ns1` we add the IP address of `ns2` and on `ns2` we add the IP address of `ns1`.

- Edit the **`named.conf`** file (`vi /chroot/named/etc/named.conf`) on both `DNS` servers, and add the following lines:

```
server x.x.x.x {
  keys { ns1-ns2 ;};
};
```

Where `x.x.x.x` is the IP address.

Once the above lines have been added, your `named.conf` file on both `DNS` servers (Primary & Secondary) should look something like:

For Primary/Master server:

```
        options {
                directory "/var/named";
                allow-transfer { 207.35.78.6; };
                allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
                allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
                version "Go away!";
        };

        logging {
                 category lame-servers { null; };
        };

        key ns1-ns2 {
          algorithm hmac-md5;
          secret "ps1jy3f7czVa1VNZkYaLfw==";
        };

        server 207.35.78.6 {
          keys { ns1-ns2 ;};
        };
```


```
// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are the master server for openna.com
zone "openna.com" {
        type master;
        file "db.openna";
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type master;
        file "db.207.35.78";
        allow-query { any; };
};
```

For Secondary/Slave server:

```
options {
        directory "/var/named";
        allow-transfer { none; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};

server 207.35.78.5 {
  keys { ns1-ns2 ;};
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are a slave server for openna.com
zone "openna.com" {
```


```
                type slave;
                file "db.openna";
                masters { 207.35.78.5; };
                allow-query { any; };
        };

        zone "78.35.207.in-addr.arpa" {
                type slave;
                file "db.207.35.78";
                masters { 207.35.78.5; };
                allow-query { any; };
};
```

Step 6
Finally, since `named.conf` file now handle a secret key, it is recommended that either `named.conf` (on Primary & Secondary servers) be non-world readable.

- This procedure can be accomplished with the following command on Primary server:
  `[root@deep /]# **chmod 600 /chroot/named/etc/named.conf**`

- This procedure can be accomplished with the following command on Secondary server:
  `[root@deep /]# **chmod 600 /chroot/named/etc/named.conf**`

Restart your DNS server on both sides for the changes to take effect.

- Restart ISC BIND & DNS with the following command on both DNS servers:
  ```
  [root@deep /]# /etc/rc.d/init.d/named restart
  Shutting down named:                       [OK]
  Starting named:                            [OK]
  ```

### Using TSIG key based access control to make a zone transfer

Once the TSIG feature has been configured and enabled in your DNS server, we can use it to improve security in the system. One improvement can be made with the `allow-transfer` statement of ISC BIND & DNS. Usually, we configure our Primary/Master Domain Name Server to respond to zone transfers requests only from authorized IP addresses. In most cases, we'll only authorize our known Secondary/Slave Domain Name Servers. The same technique as described here can also be used for dynamic update, notify, and recursive query messages too.

With BIND9, we do that within a zone phrase in the Primary Name Server with a directive like "`allow-transfer { 207.35.78.6; };`", but with the share of keys between `ns1` and `ns2` like we're doing previously, we have extended the possibility of our `named.conf` file to allow TSIG keys and can use this feature to modify the `allow-transfer` directive, which will improve security of zone transfer between `ns1` and `ns2`.

- To use TSIG key based access control to make a zone transfer between Primary DNS & Secondary DNS, edit your **`named.conf`** file on the Primary/Master Domain Name Server (`vi /chroot/named/etc/named.conf`) and change the line:

  ```
  allow-transfer { 207.35.78.6; };
  ```


To Read:

```
        allow-transfer { key ns1-ns2; };
```

This allows zone transfer to succeed only if a key named “ns1-ns2” signed the request, which only your Primary & Secondary DNS known and handle in their `named.conf` file.

Once the above line has been modified, your `named.conf` file on the Primary/Master server should look something like:

```
options {
        directory "/var/named";
        allow-transfer { key ns1-ns2; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};

server 207.35.78.6 {
  keys { ns1-ns2 ;};
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are the master server for openna.com
zone "openna.com" {
        type master;
        file "db.openna";
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type master;
        file "db.207.35.78";
        allow-query { any; };
};
```


**WARNING:** If you use BIND9's dynamic update functionality, you'll also want to restrict zone updates to authorized IP addresses and you'd probably do this in the zone phrase. Note that if you don't specify an `allow-update` option, updates are not allowed for that zone, so you'll only need to do this if you actually use dynamic update.

```
zone "openna.com" {
        type master;
        file "db.openna";
        allow-update { key ns1-ns2; };
        allow-query { any; };
};
```

### Using encryption algorithm for the name server control utility `rndc`

The BIND9 utility for controlling the name server, `rndc`, has its own configuration file `/etc/rndc.conf`, which also required a TSIG key to work. The name server must be configured to accept `rndc` connections and to recognize the key specified in the `rndc.conf` file, using the controls statement in `named.conf`. Below I show you the procedures to do before usinf `rndc` on your system.

Step 1
The first step will be to generate shared keys. This shared secret key will be included into `/etc/rndc.conf` file and `/chroot/named/etc/named.conf` file.

- To generate a random shared key, use the following command:
  `[root@deep /]#` **`dnssec-keygen -a hmac-md5 -b 128 -n user rndc`**
  `Krndc.+157+36471`

Step 2
The above command will generate a 128 bit (16 byte) HMAC-MD5 key and the result will be in a file called "`Krndc.+157+36471.private`" with a base-64 encoded string following the word "`Key:`", which must be extracted from the file and used as a shared secret.

- Edit the **`Krndc.+157+36471.private`** file (vi `Krndc.+157+36471.private`), and extract the base-64 encoded string:

  ```
  Private-key-format: v1.2
  Algorithm: 157 (HMAC_MD5)
  Key: 9kMjEIB5ikRJ6NSwtXWWVg==
  ```

The string "`9kMjEIB5ikRJ6NSwtXWWVg==`" in the above example is the part of this file that must be extracted and used as the shared secret.

Step 3
Once the required base-64 encoded string has been extracted from the generated file, we can remove the files from our system and copy the shared secret to both `rndc.conf` and `named.conf` files.

- To remove the generated files from the system, use the following commands:


```
[root@deep /]# rm -f Krndc.+157+36471.key
[root@deep /]# rm -f Krndc.+157+36471.private
```

Step 4
After that, we must edit the `rndc.conf` file and configure it with the key.

- Edit the **`rndc.conf`** file (`vi /etc/rndc.conf`), and add the following lines:

```
options {
        default-server  localhost;
        default-key     "localkey";
};

server localhost {
        key     "localkey";
};

key "localkey" {
        algorithm       hmac-md5;
        secret "9kMjEIB5ikRJ6NSwtXWWVg==";
};
```

In the above example, `rndc` will by default use the server at localhost (`127.0.0.1`) and the key called `localkey`. Commands to the localhost server will use the `localkey` key. The key statement indicates that `localkey` uses the `HMAC-MD5` algorithm and its secret clause contains the base-64 encoding of the `HMAC-MD5` secret enclosed in double quotes.

Step 5
Also don't forget to edit the `named.conf` file and configure it with the key.

- Edit the **`named.conf`** file (`vi /chroot/named/etc/named.conf`), and add the lines:

```
options {
        directory "/var/named";
        allow-transfer { key ns1-ns2; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};

server 207.35.78.6 {
  keys { ns1-ns2 ;};
};

key localkey {
  algorithm hmac-md5;
```


```
  secret "JpWopXTHbRel32xLP9x7rg==";
};

controls {
  inet 127.0.0.1 allow { 127.0.0.1; } keys { localkey; };
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are the master server for openna.com
zone "openna.com" {
        type master;
        file "db.openna";
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type master;
        file "db.207.35.78";
        allow-query { any; };
};
```

In the above example, `rndc` connection will only be accepted at localhost (`127.0.0.1`).

Step 6
Finally, it is important to restart your DNS server for the changes to take effect.

- Restart ISC BIND & DNS with the following command:
```
[root@deep /]# /etc/rc.d/init.d/named restart
Shutting down named:                       [OK]
Starting named:                            [OK]
```

### DNSSEC Cryptographic authentication of DNS information

The BIND9 release of ISC BIND & DNS includes and support validation of DNSSEC (**DNS Sec**urity) signatures in responses but should still be considered experimental. The DNSSEC feature of BIND9 is used for signed zones, what DNSSEC do is to make sure that the DNS communication taking place is with the right server, and that the information have not been tampered with during the transport. This allow protection of Internet-wide DNS transfers, cache pollution, and will protect you from someone trying to spoof your DNS servers.

But be aware that DNSSEC is NOT for every kind of Name Server. DNSSEC verify that the data received by a resolver is the same as the data published. For it to do anything, your resolver must be configured to verify data. Signing a localhost zone like for Caching-Only or Secondary/Slave Name Server is not useful, since it's not traveling over an insecure network. Signing data in general doesn't help you; it guarantees that anyone that gets data from your server can verify its correctness, if they've configured their resolver to do so.


Each zone (domain) in the DNS will need to have a key pair. The zone's public key will be included in its resource records. The zone's private key will be kept securely by the administrator of the zone, and never given to anyone outside your organization. Below, I show you all the required steps for the creation and use of DNSSEC signed zones.

In our example we assume that you want to use the DNSSEC feature for your Primary/Master Name Server with your parent zone (i.e. .COM) over the Internet. All commands listed below are assumed to be made in the **/chroot/named/var/named** directory since the DNSSEC tools require that the generated key files will be in the working directory.

Step 1
As usually in cryptography area, the first step will be to generate a key pair. The generated zone keys here, will produce a private and public key to be used to sign records for the related zones in question and the zone keys must have the same name as the zone like in our example "openna.com". The resulted public keys should later be inserted into the related zone file with the **$INCLUDE** statements.

- To generate a 1024 bit DSA key for the openna.com zone, use the following command:
```
[root@deep /]# cd /chroot/named/var/named/
[root@deep named]# dnssec-keygen -a DSA -b 1024 -n ZONE openna.com
Kopenna.com.+003+28448
```

The above command will generate a 1024 bit DSA key for the openna.com zone and two output files will be produced: "Kopenna.com.+003+28448.key" and "Kopenna.com.+003+28448.private". The private key will be used to generate signatures, and the public key will be used for signature verification.

Step 2
Once the zone keys have been generated as shown previously, a keyset must be built and transmit to the administrator of the parent zone in question to sign the keys with its own zone key. It is important that when building a keyset, at least the following information be included in the generation of the key: the TTL (**T**ime **T**o **L**ive) of the keyset must be specified, and the desired signature validity period of the parent's signature may also be specified.

- To generate a keyset containing the previous key, use the following command:
```
[root@deep named]# dnssec-makekeyset -t 3600 -e +864000 \
Kopenna.com.+003+28448
keyset-openna.com.
```

The above command generates a keyset containing the previous key with a TTL of 3600 and a signature validity period of 10 days (864000) starting from now into an output file called "keyset-openna.com.". This file should be absolutely transmitted to the parent to be signed. It includes the keys, as well as signatures over the keyset generated by the zone keys themselves, which are used to prove ownership of the private keys and encode the desired validity period.

Step 3
After that, the administrator on the parent zone (in our case .COM since our zone is openna.com) should receive the keyset files for each of your secure zone (in our example: keyset-openna.com.) and must sign the keys with its own private key. This is the step that


permits others on the net to determine that resource records that they receive in your zone are really from you.

- The administrator of your parent zone will sign the keyset with its zone keys by using something like the following command:
  ```
  [root@internic named]# dnssec-signkey keyset-openna.com. \
  KA.COM.+003+31877
  signedkey-openna.com.
  ```

One output file called "`signedkey-openna.com.`" will be produced. This file should be both transmitted back to the destinataire and retained. It will include all keys from the keyset file and signatures generated by this zone's zone keys.

**WARNING:** Take a note that in our example "`KA.COM.+003+31877`" is the key for the "`A.COM`" zone file, which is our parent zone. Olafur Gudmundsson <ogud@ogud.com> has informed me that `.COM` is not there yet, but what you SHOULD do is to contact your registrar and notify them that you MUST have your key set signed by `.COM` ASAP and when they expect that to happen. Verisign Global Registry has indicated that they want to start signing `.COM` sometime this year, but check with them what the current plans are.

To resume a little bit our procedures from now:

- ✓ We have generated a key pair for our zone file in step 1.
- ✓ We have build and transmit a keyset to our parent zone for signing in step 2.
- ✓ Administrator in the parent zone signs our keyset with its private key.
- ✓ Administrator in the parent zone transmits back our ketset after singing it.

Step 4
Ok, from now if we recall what we have said before is that the public keys should later be inserted into the related zone file with the **`$INCLUDE`** statements, then at this step, we must insert the public key (`Kopenna.com.+003+28448.key`) into the related zone file, which is in our example the zone file called `db.openna` located under `/chroot/named/var/named` directory.

- Edit the **`db.openna`** zone file (`vi /chroot/named/var/named/db.openna`), and add the following line to your default zone file:

```
; Revision History: March 01, 2001 - root@openna.com
; Start of Authority (SOA) records.
$TTL 86400
@  IN  SOA  ns1.openna.com. root.openna.com.  (
                                        00      ; Serial
                                        10800   ; Refresh after 3 hours
                                        3600    ; Retry after 1 hour
                                        604800  ; Expire after 1 week
                                        86400 ) ; Minimum TTL of 1 day

$INCLUDE Kopenna.com.+003+28448.key

; Name Server (NS) records.
        IN      NS      ns1.openna.com.
        IN      NS      ns2.openna.com.
```


```
; Mail Exchange (MX) records.
        MX      0       smtp.openna.com.

; Address (A) records.
localhost       IN      A       127.0.0.1
router          IN      A       207.35.78.1
portal          IN      A       207.35.78.2
www             IN      A       207.35.78.3
smtp            IN      A       207.35.78.4
ns1             IN      A       207.35.78.5
ns2             IN      A       207.35.78.6
```

Don't forget to restart your DNS server for the change to take effect.

- Restart ISC BIND & DNS with the following command:
```
[root@deep /]# /etc/rc.d/init.d/named restart
Shutting down named:                       [OK]
Starting named:                            [OK]
```

**NOTE:** Please, check if everything looks right under your log files (`/var/log/messages`) before continuing to the step below. This is important to be sure that there is not something wrong with your configuration.

Step 5
Once the keyset has been signed and approved by the parent zone (`.COM`), the final step will be to sign our zone. The result will produce one output file called "`db.openna.signed`". This file should be referenced by `named.conf` as the input file for the zone instead of the default one called "`db.openna`".

- To sign the zone file, use the following command:
```
[root@deep named]# dnssec-signzone -o openna.com db.openna
db.openna.signed
```

- One last requirement will be to change the owner of the `db.openna.signed` file to be the user under which our `named` daemon runs. In our case the `named` daemon runs under a name, which is also called "`named`":
```
[root@deep named]# chown named.named db.openna.signed
```

**NOTE:** If a zone doesn't publish a key, then BIND will accept any plausible-looking records, without a digital signature, just like in the original DNS. This provides compatibility with existing DNS zones, allowing Secure DNS to be gradually introduced throughout the Internet.

Step 6
The result of signing the zone will produce one output file called "`db.openna.signed`". Recall that this file should be referenced by `named.conf` as the input file for the zone.


- Edit the **`named.conf`** file (`vi /chroot/named/etc/named.conf`), and change the following line:

```
options {
        directory "/var/named";
        allow-transfer { key ns1-ns2; };
        allow-query { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        allow-recursion { 192.168.1.0/24; 207.35.78.0/32; localhost; };
        version "Go away!";
};

logging {
         category lame-servers { null; };
};

key ns1-ns2 {
  algorithm hmac-md5;
  secret "ps1jy3f7czVa1VNZkYaLfw==";
};

server 207.35.78.6 {
  keys { ns1-ns2 ;};
};

key localkey {
  algorithm hmac-md5;
  secret "JpWopXTHbRel32xLP9x7rg==";
};

controls {
  inet 127.0.0.1 allow { 127.0.0.1; } keys { localkey; };
};

// Root server hints
zone "." { type hint; file "db.cache"; };

// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
        type master;
        file "db.127.0.0";
        notify no;
};

// We are the master server for openna.com
zone "openna.com" {
        type master;
        file "db.openna.signed";
        allow-query { any; };
};

zone "78.35.207.in-addr.arpa" {
        type master;
        file "db.207.35.78";
        allow-query { any; };
};
```

Step 7
Don't forget to restart your DNS server for the changes to take effect.


- Restart ISC BIND & DNS with the following command on both DNS servers:

```
[root@deep /]# /etc/rc.d/init.d/named restart
Shutting down named:                       [OK]
Starting named:                            [OK]
```

## Optimizing ISC BIND & DNS

This section deals especially with actions we can make to improve and tighten performance of ISC BIND & DNS. Take a note that we refer to the features available within the base installed program.

### The BIND9 Lightweight Resolver

The new release of Bind comes with a new daemon program called "lwresd". The lwresd daemon is essentially a Caching-Only Name Server that answers requests using the lightweight resolver protocol rather than the DNS protocol. Because it needs to run on each host, it is designed to require no or minimal configuration. In our configuration we'll run lwresd in a chrooted environment.

On all Caching-Only Name Server that you may have in your network, it can be interesting to run this daemon "lwresd" instead of the full "named" daemon. If we remember, Caching-Only Name Server is server not authoritative for any domains except 0.0.127.in-addr.arpa. It can look up names inside and outside your zone, as can Primary and Slave Name Servers but the difference is that when it initially looks up a name within your zone, it ends up asking one of the Primary or Slave Names Servers for your zone for the answer and nothing else. Therefore we can run the "lwresd" daemon in this kind of Name Server and everything will run, as we want.

Below, I show you the required steps to run your Caching-Only Name Server with the "lwresd" daemon instead of the "named" daemon in a chrooted environment.

Step 1
By default, applications using the lightweight resolver library will make UDP requests to the IPv4 loopback address (127.0.0.1) on port **921**. Therefore it is important to add a new firewall rule lines related to lwresd into your firewall script file to allow communication on this port (**921**).

- Edit the **firewall** file (vi /etc/rc.d/init.d/firewall), and add the following lines:

```
# LWRESD server (921)
# ------------------

# A lightweight resolver library for Caching-Only Name Server

iptables -A INPUT  -i $EXTERNAL_INTERFACE -p udp  \
         --source-port $UNPRIVPORTS \
         -d $IPADDR --destination-port 921 -j ACCEPT

iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp  \
         -s $IPADDR --source-port 921 \
         --destination-port $UNPRIVPORTS -j ACCEPT
```


Step 2
By default, the `lwresd` daemon listens on the loopback address (`127.0.0.1`). With a firewall on the system it is important to instruct the `lwresd` daemon to listen to the external interface of the server. This can be made with an "`lwserver`" statement lines in the `/etc/resolv.conf` file.

- Edit the **`resolv.conf`** file (`vi /etc/resolv.conf`), and add the following line:

  ```
  lwserver 207.35.78.2
  ```

Where `207.35.78.2` is the IP address of the external interface in the firewall script file.

Step 3
Since `lwresd` will run in a chroot jail environment, we must copy the `/etc/resolv.conf` file to our chrooted environment for the `lwresd` daemon to be able to find the `resolv.conf` file and start.

- To copy the `resolv.conf` file to your chroot jail, use the following command:
  `[root@deep /]#` **`cp /etc/resolv.conf /chroot/named/etc/`**

Step 4
Now, we must create an initialization script file for the `lwresd` daemon to automatically start and stop on your server.

- Create the **`lwresd`** script file (`touch /etc/rc.d/init.d/lwresd`) and add the following lines inside it:

```
#!/bin/bash
#
# lwresd           This shell script takes care of starting and stopping
#                 lwresd (The lightweight resolver library).
#
# chkconfig: - 55 45
# description: lwresd is essentially a Caching-Only Name Server \
# that answers requests using the lightweight resolver \
# protocol rather than the DNS protocol.
# 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 /etc/sysconfig/named ] && . /etc/sysconfig/named

[ -f /usr/sbin/lwresd ] || exit 0

[ -f "${ROOTDIR}"/etc/resolv.conf ] || exit 0

RETVAL=0
```


```
start() {
        # Start daemons.
        echo -n "Starting lwresd: "
        if [ -n "${ROOTDIR}" -a "x${ROOTDIR}" != "x/" ]; then
                OPTIONS="${OPTIONS} -t ${ROOTDIR}"
        fi
        daemon lwresd -u named ${OPTIONS}
        RETVAL=$?
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/lwresd
        echo
        return $RETVAL
}
stop() {
        # Stop daemons.
        echo -n "Shutting down lwresd: "
        killproc lwresd
        RETVAL=$?
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/lwresd
        echo
        return $RETVAL
}
restart() {
        stop
        start
}

# See how we were called.
case "$1" in
        start)
                start
                ;;
        stop)
                stop
                ;;
        restart)
                restart
                ;;
        *)
                echo "Usage: lwresd {start|stop|restart}"
                exit 1
esac

exit $?
```

Step 5

Once the `lwresd` script file has been created, it is important to make it executable, change its default permissions, create the necessary links and start it. Making this file executable will allow the system to run it, changing its default permission is to allow only the root user to change this file for security reason, and creation of the symbolic links will let the process control initialization of Linux which is in charge of starting all the normal and authorized processes that need to run at boot time on your system to start the program automatically for you at each reboot.

- To make this script executable and to change its default permissions, use the commands:
  `[root@deep /]# **chmod 700 /etc/rc.d/init.d/lwresd**`
  `[root@deep /]# **chown 0.0 /etc/rc.d/init.d/lwresd**`


- To create the symbolic `rc.d` links for `lwresd`, use the following commands:
```
[root@deep /]# chkconfig --add lwresd
[root@deep /]# chkconfig --level 2345 lwresd on
```

Step 6
Finally, since we run now the `lwresd` instead of `named` daemon in our Caching-Only Name Server, it is important to deactivate and uninstall the `named` initialization script file in our system.

- These procedures can be accomplished with the following commands:
```
[root@deep /]# chkconfig --del named
[root@deep /]# chkconfig --level 2345 named off
[root@deep /]# rm -f /etc/rc.d/init.d/named
```

Step 7
Now it is time to start your DNS server with the `lwresd` daemon.

- To start `lwressd` manually, use the following command:
```
[root@deep /]# /etc/rc.d/init.d/lwresd start
Starting lwresd:                           [OK]
```

## Further documentation
For more details, there are several manual pages you can read:

```
$ man named-checkconf (1)        - Configuration file syntax checking tool
$ man named-checkzone (1)        - Zone validity checking tool
$ man host (1)                                  - DNS lookup utility
$ man dig (1)                              - DNS lookup utility
$ man rndc.conf (5)                        - rndc configuration file
$ man named (8)                                 - Internet domain name server
$ man rndc (8)                                  - name server control utility
$ man lwresd (8)                           - lightweight resolver daemon
$ man nsupdate (8)                         - Dynamic DNS update utility
```

## ISC BIND & DNS Administrative Tools
The commands listed below are some that we use often, but many more exist. Check the manual pages of ISC BIND & DNS and documentation for more details and information.

### dig
The `dig` command DNS lookup utility (**d**omain **i**nformation **g**roper) is a tool for interrogating DNS name servers by performing DNS lookups and displays the answers that are returned from. It can also be used to update your `db.cache` file by telling your server where the servers for the "root" zone are. `Dig` is a useful tool to use when you want to troubleshoot DNS problems.

- Use the following command to query an address:
```
[root@deep /]# dig @www.openna.com

; <<>> DiG 9.1.0 <<>> @www.openna.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 20994
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```


```
;; QUESTION SECTION:
;.                              IN      NS

;; Query time: 3 msec
;; SERVER: 207.35.78.5#53(ns1.openna.com)
;; WHEN: Fri Feb 23 19:16:51 2001
;; MSG SIZE  rcvd: 17
```

Where **@www.openna.com** is the address of the server. Many options exist for this toll and I recommend you to read the `dig` manual page `dig(1)` for a complete list.

### rndc

The `rndc` command utility allows the system administrator to control the operation of a name server. It replace the `ndc(8)` utility that was provided in old `BIND8` releases. You can use this tool to reload configuration file and zones, schedule immediate maintenance for a zone, write server statistics, toggle query logging, stop the `DNS` server, and many other thing. The `rndc` tool prints a short summary of the supported commands and the available options if invoked without command line options.

- Type `rndc` on your terminal to get a short summary of all available and supported commands:

```
[root@deep /]# rndc
Usage: rndc [-c config] [-s server] [-p port] [-y key] [-z zone] [-v
view]
        command [command ...]

command is one of the following:

  reload        Reload configuration file and zones.
  reload zone [class [view]]
                Reload a single zone.
  refresh zone [class [view]]
                Schedule immediate maintenance for a zone.
  stats         Write server statistics to the statistics file.
  querylog      Toggle query logging.
  dumpdb        Dump cache(s) to the dump file (named_dump.db).
  stop          Save pending updates to master files and stop the server.
  halt          Stop the server without saving pending updates.
  *status       Display ps(1) status of named.
  *trace        Increment debugging level by one.
  *notrace      Set debugging level to 0.
  *restart      Restart the server.

* == not yet implemented
Version: 9.1.0
```

## ISC BIND & DNS Users Tools

The commands listed below are some that we use often, but many more exist. Check the manual pages of `ISC BIND & DNS` and documentation for more details and information.

### nslookup

The `nslookup` program allows the user to query Internet domain name servers interactively or non-interactively. In interactive mode the user can query name servers for information about


various hosts and domains, and print a list of hosts in a domain. In non-interactive mode the user can just print the name and request information for a host or domain.

Interactive mode has a lot of options and commands; it is recommended that you see the manual page for `nslookup`.

- To enter under `nslookup` Interactive mode, use the command:

```
[root@deep /]# nslookup
> www.openna.com
Server:         207.35.78.5
Address:        207.35.78.5#53

Name:   www.openna.com
Address: 207.35.78.3
> exit
```

- To run in non-interactive mode, use the command:

```
[root@deep /]# nslookup www.openna.com
Server:         207.35.78.5
Address:        207.35.78.5#53

Name:   www.openna.com
Address: 207.35.78.3
```

Where <`www.openna.com`> is the host name or Internet address of the name server to be looked up.

### `host`

The `host` tool is a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa. When no arguments or options are given, `host` prints a short summary of its command line arguments and options.

- To prints `host` command line arguments and options, use the command:

```
[root@deep /]# host
Usage: host [-aCdlrTwv] [-c class] [-n] [-N ndots] [-t type] [-W time]
            [-R number] hostname [server]
       -a is equivalent to -v -t *
       -c specifies query class for non-IN data
       -C compares SOA records on authorative nameservers
       -d is equivalent to -v
       -l lists all hosts in a domain, using AXFR
       -n Use the nibble form of IPv6 reverse lookup
       -N changes the number of dots allowed before root lookup is done
       -r disables recursive processing
       -R specifies number of retries for UDP packets
       -t specifies the query type
       -T enables TCP/IP mode
       -v enables verbose output
       -w specifies to wait forever for a reply
       -W specifies how long to wait for a reply
```

- To look up host names using domain server, use the command:

```
[root@deep /]# host openna.com
```


```
openna.com. has address 207.35.78.3
```

## List of installed ISC BIND & DNS files on your system

> /etc/rndc.conf
> /usr/bin/dig
> /usr/bin/host
> /usr/bin/nslookup
> /usr/bin/nsupdate
> /usr/bin/isc-config.sh
> /usr/include/isc
> /usr/include/isc/assertions.h
> /usr/include/isc/base64.h
> /usr/include/isc/bitstring.h
> /usr/include/isc/boolean.h
> /usr/include/isc/buffer.h
> /usr/include/isc/bufferlist.h
> /usr/include/isc/commandline.h
> /usr/include/isc/entropy.h
> /usr/include/isc/error.h
> /usr/include/isc/event.h
> /usr/include/isc/eventclass.h
> /usr/include/isc/file.h
> /usr/include/isc/formatcheck.h
> /usr/include/isc/fsaccess.h
> /usr/include/isc/heap.h
> /usr/include/isc/hex.h
> /usr/include/isc/hmacmd5.h
> /usr/include/isc/interfaceiter.h
> /usr/include/isc/lang.h
> /usr/include/isc/lex.h
> /usr/include/isc/lfsr.h
> /usr/include/isc/lib.h
> /usr/include/isc/list.h
> /usr/include/isc/log.h
> /usr/include/isc/magic.h
> /usr/include/isc/md5.h
> /usr/include/isc/mem.h
> /usr/include/isc/msgcat.h
> /usr/include/isc/msgs.h
> /usr/include/isc/mutexblock.h
> /usr/include/isc/netaddr.h
> /usr/include/isc/ondestroy.h
> /usr/include/isc/os.h
> /usr/include/isc/print.h
> /usr/include/isc/quota.h
> /usr/include/isc/random.h
> /usr/include/isc/ratelimiter.h
> /usr/include/isc/refcount.h
> /usr/include/isc/region.h
> /usr/include/isc/resource.h
> /usr/include/isc/result.h
> /usr/include/isc/resultclass.h
> /usr/include/isc/rwlock.h
> /usr/include/isc/serial.h
> /usr/include/isc/sha1.h
> /usr/include/isc/sockaddr.h
> /usr/include/isc/socket.h
> /usr/include/isc/stdio.h
> /usr/include/isc/string.h
> /usr/include/isc/symtab.h
> /usr/include/isc/task.h
> /usr/include/isc/taskpool.h
> /usr/include/isc/timer.h
> /usr/include/dns/keyflags.h
> /usr/include/dns/keytable.h
> /usr/include/dns/keyvalues.h
> /usr/include/dns/lib.h
> /usr/include/dns/log.h
> /usr/include/dns/master.h
> /usr/include/dns/masterdump.h
> /usr/include/dns/message.h
> /usr/include/dns/name.h
> /usr/include/dns/namedconf.h
> /usr/include/dns/ncache.h
> /usr/include/dns/nxt.h
> /usr/include/dns/peer.h
> /usr/include/dns/rbt.h
> /usr/include/dns/rcode.h
> /usr/include/dns/rdata.h
> /usr/include/dns/rdataclass.h
> /usr/include/dns/rdatalist.h
> /usr/include/dns/rdataset.h
> /usr/include/dns/rdatasetiter.h
> /usr/include/dns/rdataslab.h
> /usr/include/dns/rdatatype.h
> /usr/include/dns/request.h
> /usr/include/dns/resolver.h
> /usr/include/dns/result.h
> /usr/include/dns/rootns.h
> /usr/include/dns/sdb.h
> /usr/include/dns/secalg.h
> /usr/include/dns/secproto.h
> /usr/include/dns/ssu.h
> /usr/include/dns/tcpmsg.h
> /usr/include/dns/time.h
> /usr/include/dns/tkey.h
> /usr/include/dns/tsig.h
> /usr/include/dns/ttl.h
> /usr/include/dns/types.h
> /usr/include/dns/validator.h
> /usr/include/dns/view.h
> /usr/include/dns/xfrin.h
> /usr/include/dns/zone.h
> /usr/include/dns/zt.h
> /usr/include/dns/enumclass.h
> /usr/include/dns/enumtype.h
> /usr/include/dns/rdatastruct.h
> /usr/include/dst
> /usr/include/dst/dst.h
> /usr/include/dst/lib.h
> /usr/include/dst/result.h
> /usr/include/lwres
> /usr/include/lwres/context.h
> /usr/include/lwres/lwbuffer.h
> /usr/include/lwres/lwpacket.h
> /usr/include/lwres/lwres.h
> /usr/include/lwres/result.h
> /usr/include/lwres/int.h
> /usr/include/lwres/lang.h
> /usr/include/lwres/list.h
> /usr/include/lwres/net.h
> /usr/include/lwres/ipv6.h
> /usr/include/lwres/netdb.h


> /usr/include/isc/types.h
> /usr/include/isc/util.h
> /usr/include/isc/platform.h
> /usr/include/isc/app.h
> /usr/include/isc/dir.h
> /usr/include/isc/int.h
> /usr/include/isc/net.h
> /usr/include/isc/netdb.h
> /usr/include/isc/offset.h
> /usr/include/isc/stdtime.h
> /usr/include/isc/time.h
> /usr/include/isc/condition.h
> /usr/include/isc/mutex.h
> /usr/include/isc/once.h
> /usr/include/isc/thread.h
> /usr/include/dns
> /usr/include/dns/a6.h
> /usr/include/dns/acl.h
> /usr/include/dns/adb.h
> /usr/include/dns/byaddr.h
> /usr/include/dns/cache.h
> /usr/include/dns/callbacks.h
> /usr/include/dns/cert.h
> /usr/include/dns/compress.h
> /usr/include/dns/confacl.h
> /usr/include/dns/confcache.h
> /usr/include/dns/confcommon.h
> /usr/include/dns/confctl.h
> /usr/include/dns/confctx.h
> /usr/include/dns/confip.h
> /usr/include/dns/confkeys.h
> /usr/include/dns/conflog.h
> /usr/include/dns/conflsn.h
> /usr/include/dns/conflwres.h
> /usr/include/dns/confparser.h
> /usr/include/dns/confresolv.h
> /usr/include/dns/confrrset.h
> /usr/include/dns/confview.h
> /usr/include/dns/confzone.h
> /usr/include/dns/db.h
> /usr/include/dns/dbiterator.h
> /usr/include/dns/dbtable.h
> /usr/include/dns/diff.h
> /usr/include/dns/dispatch.h
> /usr/include/dns/dnssec.h
> /usr/include/dns/events.h
> /usr/include/dns/fixedname.h
> /usr/include/dns/journal.h
> /usr/include/lwres/platform.h
> /usr/include/omapi
> /usr/include/omapi/compatibility.h
> /usr/include/omapi/lib.h
> /usr/include/omapi/omapi.h
> /usr/include/omapi/private.h
> /usr/include/omapi/result.h
> /usr/include/omapi/types.h
> /usr/lib/libisc.so.3.0.0
> /usr/lib/libisc.so.3
> /usr/lib/libisc.so
> /usr/lib/libisc.la
> /usr/lib/libisc.a
> /usr/lib/libdns.so.4.0.0
> /usr/lib/libdns.so.4
> /usr/lib/libdns.so
> /usr/lib/libdns.la
> /usr/lib/libdns.a
> /usr/lib/liblwres.so.1.1.0
> /usr/lib/liblwres.so.1
> /usr/lib/liblwres.so
> /usr/lib/liblwres.la
> /usr/lib/liblwres.a
> /usr/lib/libomapi.so.3.0.0
> /usr/lib/libomapi.so.3
> /usr/lib/libomapi.so
> /usr/lib/libomapi.la
> /usr/lib/libomapi.a
> /usr/sbin/named
> /usr/sbin/lwresd
> /usr/sbin/rndc
> /usr/sbin/dnssec-keygen
> /usr/sbin/dnssec-makekeyset
> /usr/sbin/dnssec-signkey
> /usr/sbin/dnssec-signzone
> /usr/sbin/named-checkconf
> /usr/sbin/named-checkzone
> /usr/share/man/man1/named-checkconf.1
> /usr/share/man/man1/named-checkzone.1
> /usr/share/man/man1/host.1
> /usr/share/man/man1/dig.1
> /usr/share/man/man5/rndc.conf.5
> /usr/share/man/man8/named.8
> /usr/share/man/man8/rndc.8
> /usr/share/man/man8/lwresd.8
> /usr/share/man/man8/nsupdate.8
> /var/named
> /var/run/named