The Domain Name System: Bind 9.2.4

By Jaume Sabater, published on 27 May 2002.

Index

- Summary.
- Objectives.
- Introduction
  - What is DNS?
  - Who needs DNS?
- Requirements and technical data.
- Installation.
- Translation of names to IP addresses.
- Reverse translation.
- Secondary servers.
- Secure zone transfer
- What is a TSIG and what is it needed for?
- Creating a TSIG key.
- Comments on dynamic update, TSIG security, and ACLs.
- Risks to which an insecure Bind is exposed.
- So, what measures need to be taken?
- Recursive and non-recursive servers.
- Location of services.
- DNS record types.
- Views.
- The RNDC tool.
- Customization of logs.
- Taxonomy of a name server.
- Types of statements used in named.conf.
  - Example of log customization.
  - A table of special characters used in resource records.
  - Table of security mechanisms in named.conf.
  - Table of logging categories in Bind 9.
- Chroot.
  - Create the user.
  - Directory structure.
  - Copy the necessary files.
  - System files.
  - Logging
  - Toughening permits.
  - Installation.
  - Configuration changes.
  - Starting BIND.
- Online resources.
- The RFC.
- Revision history.

Summary.

Millions of hosts are connected to the Internet. How do you keep track of all of them when they belong to so many different countries, networks, and administrative groups? Two basic pieces of infrastructure hold all of that together: the Domain **Name System (DNS**), whose job is to know who each host is, and the Internet routing system, which is responsible for knowing how they are connected. This article refers to the portion that DNS represents in that system.

Objectives.

The objectives of a domain name server (DNS) are twofold:

1. On the one hand, translating a canonical address into an IP (**Internet Protocol**) address. For example, *linuxsilo.net* is, on the date of creation of the article,
   66.79.182.201.
2. On the other hand, translate an IP address into one or more canonical addresses. This is what is known as reverse translation.

Making a simile, the first point would be equivalent to looking in a phonebook for a person's phone number, given their name and surname, while the second would be the reverse process: given a phone number, find out which person it corresponds to.

The correlation between an IP address and a domain name doesn't have to be unique. This is due to what are known as virtual domains. In fact, it's common for one IP address to be equivalent to multiple domain names. For example, the IP address *66.79.182.201* is equivalent to *linuxsilo.net*, *www.linuxsilo.net*, *ftp.linuxsilo.net*, *pop3.linuxsilo.net,* and others. However, this does not mean that the same machine (or host) *66.79.182.201* is offering all of those services. This is possible thanks to what is known as packet **routing**, but it is not relevant to the article.

Introduction.

What is DNS?

DNS is a hierarchical system with a tree structure. The beginning is written *"."* and it is called *root*, just as in tree data structures. Under the root are the **Top Level Domains** (TLDs), the most representative examples of which are *ORG,* COM, *EDU* and *NET*, although there are many more. Like a tree, it has a root and branches that grow from it. If the reader is versed in computer science, he will recognize the DNS as a search tree and will be able to find nodes, leaf nodes and other concepts in it.

When a machine is searched, the query is executed recursively in the hierarchy, starting with the root. If you want to find the IP address of *ftp.akane.linuxsilo.net.*, the nameserver has to start asking somewhere. Start by looking at its cache. If you know the answer, because you had previously searched for it and saved it in that cache, you will answer directly. If it doesn't know, then it will remove parts of the name, starting from the left, checking if it knows anything about *akane.linuxsilo.net.*, after *linuxsilo.net.*, then *net.* and, finally, of *"."*, of which there is always information since it is located in one of the configuration files on the hard drive. It will then ask the server *"."* About *ftp.akane.linuxsilo.net*. Said server *"."* You won't know the answer, but it will help our server in your search by giving you a reference of where to keep looking. These references will take our server to the name server that knows the answer.

So, starting at *"."* We find the successive nameservers for each level in the domain name by reference. Of course, our name server will save all the information obtained throughout the process, so that you do not have to ask again for a long time.

In the analogous tree, each *"."* in the name it is a leap to another branch. And each part among the *"."* are the names of the particular nodes in the tree. We climb the tree taking the name we want (*ftp.akane.linuxsilo.net*) asking the root ("*.*") or to the server that parent from root to *ftp.akane.linuxsilo.net* about which we have information in the cache. Once the cache limits are reached, it is resolved recursively by asking the servers, chasing the references (branches) to the name.

Another concept that is not talked about as much, but is no less important, is the *in-addr.arpa* domain, which is also nested like the "normal" domains. *in-addr.arpa* allows us to get hold of the hostname when we have its address. It's worth noting here that the IP addresses are written in reverse order in the domain *in-addr.arpa*. If you have a machine's address such as *192.168.0.1*, the name server will proceed in the same way as with the ftp.akane.linuxsilo.net example. That is, it will look for the *harp servers.*, then the *in-addr.arpa servers.*, then the *192.in-addr.harp.*, then *168.192.inaddr.arpa.* and, finally, the *0.168.192.in-addr.arpa servers.* In the latter, you will find the searched record: *1.0.168.192.in-addr.arpa.*

Who needs DNS?

The DNS defines:

1. A hierarchical namespace for hosts and IP addresses.
2. A table of hosts deployed as a distributed database.
3. A translator (from English, **resolver**) or routine library that allows you to query that database.
4. Improved routing for email.
5. A mechanism for finding services in a network.
6. A protocol for exchanging name information.

To be true citizens of the Internet, sites need DNS. Keeping a local */etc/hosts* file with a map of all the hosts that users may want to contact is not feasible.

Each site maintains one or more pieces of the distributed database that enables the global service of the DNS system. Your piece of the database consists of two or more text files containing records for each of the hosts. Each record is a simple line consisting of a name (usually a host's name), a record type, and various values or data.

DNS is a client/server system. The (name) servers load the data from their DNS files into memory and use it to respond to queries from both clients on the internal network and from clients and other servers on the Internet. All of your hosts should be DNS clients, but relatively few need to be DNS servers.

If your organization is small (a few hosts on a single network), you can run a server on one of your computers or ask your ISP (**Internet Services Provider**) to provide that service on your behalf. A medium-sized site with multiple subnets should have multiple DNS servers to reduce query latency and improve productivity. A very large system can divide its DNS domains into subdomains and use a few servers for each subdomain.

Requirements and technical data.

This article will learn how to install and configure BIND (**Berkeley Internet Name Domain**) on a Linux system. A minimum knowledge of TCP/IP (**Transmission Control Protocol / Internet Protocol**) networks and Linux administration will be assumed , or at least a basic knowledge of how a system of this type works. These are the points that will be covered and the software and hardware used.

Software:

1. Debian GNU/Linux Sarge
2. Bind 9.2.4 3. DNS Utils
4. Bind Docs

Services:

1. Translation of names to IP addresses.
2. Reverse translation (from IP addresses to names).
3. Access control lists.
4. Secondary servers.
5. Secure zone transfer between primary and secondary servers (and ports).
6. Service Location (SRV Records - RFC2052, **Request For Comments**).
7. Responses parameterized according to the origin of the request (views).
8. Using the *rndc tool*.
9. Custom logs.

For this article, we will use the FQDN (**Fully Qualified Domain Name**) *linuxsilo.net* and the ns1.linuxsilo.net and ns2.linuxsilo.net nameservers. An FQDN is made up of a host and a domain name, including the top-level domain. For example, *www.linuxsilo.net* is an FQDN.

www is the host, *linuxsilo* is the second-level domain, and *net* is the highest-level domain. An FQDN always starts with the hostname and continues to go straight up to the top-level domain, so *ftp.akane.linuxsilo.net* is also an FQDN. *akane* is not an FQDN.

Installation.

It is always more than advisable to have the latest version of this type of software, as we can find in it important bugs and security bugs corrected, as well as new functionalities that facilitate our task as system administrators. The installation process on a Debian Linux distribution is as simple as running (as *root*, of course, in the same way as in the rest of the article):

```
    apt-get install bind9 bind9-doc dnsutils
```

Depending on the version of Debian, the Bind9 package will not be available (for example, version 8 is found in *Potato*), so we will need to update to a more current version (*Woody* or *Sid* at the time of writing) and proceed. The installation leaves us with a Bind with a basic configuration (in */etc/bind/*) and working, so we will only have to configure it according to our needs. We'll start with translating names to IP addresses.

The Debian *bind9* package is installed with a configuration already functional for the vast majority of terminal servers without requiring user action.

The named.conf *configuration file* of the daemon named *(name in the system of the Bind domain name server daemon) is located in /etc/bind*, so that all static configuration files related to Bind are in the same place. It is strongly recommended not to modify this configuration, especially on a GNU/Debian Linux system. However, if it is necessary to do so, possibly the best way is to use a symbolic link to the location you want to use.

The zone data files for the root servers and the **forward** and **reverse** zones for the local host are also located in */etc/bind*. The **working directory** for *named* is */var/cache/bind*. Therefore, any temporary files generated by named, such as the database files of zones that are secondary to the daemon, will be written to the */var* filesystem, which is where they belong. To make this work, the *named.conf* provided with the installation explicitly uses **fully-qualified** or **absolute pathnames** to reference the files in */etc/bind*.

Unlike previous Debian packages from Bind, *the named.conf* files and all *db.** files in the installation are considered configuration files. Therefore, if only a "cache" configuration is required for a server that is not the authorized server of any domain, the provided configuration can be executed as is. If it is necessary to change options in the *named.conf*, or even in the *init.d*, it can be done without obligation, as future updates will respect these changes, following the Debian package policy.

While the reader is free to devise the structure he likes best for the servers for which he needs to be authorized, it is suggested that all *db* files for the zones of which he is the master server be in */etc/bind* (perhaps even in a subdirectory structure, depending on the complexity). and use absolute paths in the *named.conf* file. Any secondary zones should be configured in *named.conf* as pathless filenames, so that data files end up being created in */var/cache/bind*. Throughout the article, this concept will be illustrated for a better understanding.

Translation of names to IP addresses.

The first step is to edit the */etc/bind/named.conf.options* file, where you'll change some of the default values and add everything you need to make your domain accessible from the outside.

Unless we are an Internet Service Provider, we will have been provided with one or more stable nameserver IP addresses, which we will surely want to use as **forwarders**, although it is not essential to achieve the basic objectives of this article. To do this, we must uncomment the block almost at the beginning of the file:

```
    Forwarders {
    //
0.0.0.0;     //
};
```

And leave it at something like this:

```
    forwarders {
66.79.160.3;
    };
```

Where the IPs are those corresponding to our ISP. This directive instructs our server to pass all requests for which it is not authorized or does not have the cached response to another nameserver. If you do not specify them, the DNS root servers will be used. Other interesting options from Bind (within the *options* directive and ending in semicolons) are:

1. `pid-file "/var/run/named.pid";`, which would define the location of the file containing the PID (**Process IDentificator**) of

the daemon *named*, 2. `stacksize 30M;`, which would determine a stack size of thirty megabytes,

3. `datasize 20M;`, which would specify a maximum memory size dedicated to storing data of twenty megabytes,
4. `transfer-format many-servers;`, which would cause the parallel transfer of multiple zones to the secondary servers, speeding up the process,
5. `allow-transfer { slaves; };`, which would globally limit zone transfers to secondary servers in the *slaves* list (see below for using access control lists),
6. and `"DNS server" version;`, which would hide the version of Bind that is running, for the sake of greater system security.

The areas for our domains will then be registered. If we open the */etc/bind/named.conf with a text editor,* which comes by default with the installation, we find five areas:

- The root (the point)
- localhost 127.in-
- addr.arpa
- 0.in-addr.harp
-

255.in-addr.harp

The first one introduces the root servers to our DNS server, while the other four take care of the normal and reverse translation of the localhost. From here, we open the */etc/bind/named.conf.local file* and in it we create the zone of our domain:

```
    zone "linuxsilo.net"
{ type master;
      file
"/etc/bind/db.linuxsilo.net";
allow-query { any; };       allow-
transfer { slaves; };
    };
```

The order of the zones is completely irrelevant, but it is recommended to leave them in alphabetical order for easier location in the future. Note that the name of the area does not end in *"."* (period). This is the purpose of the parameters of each zone:

1. `type master;` It means that the domain server is the primary or master of the zone. Later, when configuring secondary servers, type slave will be used *;*.
2. `file "/etc/bind/db.linuxsilo.net";` It is the file where we will specify the configuration of that area. Note that an absolute path is used, following the Debian directory policy. The contents of this file will be specified shortly.
3. `allow-query { any; };` It means that queries outside the area are allowed. This is useful and necessary, unless you want to be very paranoid about security. Information that is publicly accessible is simply offered in a technically orderly manner.
4. `allow-transfer { slaves; };` It enables the automatic transfer of this configuration to the secondary servers in the zones under our control that are specified in the *Slaves* list. The zone transfer point will be further explored.

Surely the reader has already noticed that two special words have been used, `any` and `slaves`, which require special mention. Indeed, in addition to noting the syntax similar to that of the C programming language, with which you must be extremely careful, there are two extra comments to make:

1. `any` is a reserved word in the bind syntax that stands for "any IP address", as was logical. Its use is very common and necessary. Other important reserved words are `none`, which means "no host," `localhost`, which means the local host from any of the system interfaces, and `localnets`, which represents all hosts in the networks for which the system has an interface.
2. `slaves`, on the other hand, is not a reserved word for bind, but corresponds to the concept of **an Access Control List** (ACL). These lists of IP addresses save us work because, in this way, we only have to specify them once and, since we assign them a group identifier, we can reference them in a simpler and faster way. Here's the code of the ACL used in the example which, of course, must be specified somewhere in the document before it's used:

```
        ACL "slaves" {
213.96.79.79;
        };
```

The reader will have immediately realized the great advantages of using these lists, either because the list is used in several areas, or because we have more than one slave server. Note that the identifiers of ACLs are case **sensitive**.

The contents of the linuxsilo.net zone data file are detailed below:

```
    ;
    ; BIND data file for zone linuxsilo.net
    ;
    $TTL 604800
    @ IN SOA linuxsilo.net. hostmaster.linuxsilo.net. (
      2005052401   ; Serial yyyy/mm/dd/id
           10800   ; Refresh (3 hours)
            7200   ; Retry (2 hours)
         1296000   ; Expire (15 days)
          172800 ) ; Negative Cache TTL (2 days)

    @ IN NS ns1.linuxsilo.net.
    @ IN NS ns2.linuxsilo.net.
    @ IN MX 20 mx1.linuxsilo.net.
    @ IN MX 30 mx2.linuxsilo.net.
    @IN TXT "Linux Silo Dot Net"
    @ IN HINFO "Intel Pentium IV" "Debian Linux"
    @ IN LOC 39 34 58 N 2 38 2 E 100m 10000m 20m 100m

    @ IN A 66.79.182.201
ns1 IN A 66.79.182.201 ns2
IN A 213.96.79.79 mx1 IN A
66.79.182.201 mx2 IN A
213.96.79.79 www IN A
66.79.182.201 www2 IN A
66.79.182.201 webmail IN A
66.79.182.201

    ssh.tcp SRV 0 0 22 linuxsilo.net.
    smtp.tcp SRV 0 0 25 mx1.linuxsilo.net.
    http.tcp SRV 0 3 80 linuxsilo.net.
    http.tcp SRV 0 1 80
www2.linuxsilo.net.     https.tcp SRV 1 0
443 linuxsilo.net.     pop3s.tcp SRV 0 0
995 mx1.linuxsilo.net.

    *.tcp SRV 0 0 0 .
*.udp SRV 0 0 0 .
```

Each and every one of the directives and options of these configuration files is then commented on (a semicolon, *";"*, indicates that everything to its right is a comment):

1. `$TTL 604800`: mandatory policy from version 9 of Bind (RFC1035 and RFC2308), indicates the **Time To Live** (TTL) of the information contained in the file. That is, the maximum time of validity, after which it must be refreshed or updated (to check that it has not changed). This is known as **positive/negative caching**, as specified in the RFC2308. By default, seconds are used (604800 seconds equals seven days exactly), but weeks (`$TTL 1w`), days (`$TTL 7d`), hours (`$TTL 168h`) and minutes (`$TTL 10080m`) can also be used. These abbreviations are also used in the SOA registry, which is explained below.

   Another interesting directive, although not used in the examples, is `$INCLUDE <zone-file>`, which causes *named* to include another zone file in the place where the directive is used. This allows you to store configuration parameters common to multiple subzones in a separate place in the file from the main zone.

2. `@ IN SOA linuxsilo.net. hostmaster.linuxsilo.net.`: The SOA (**Start Of Authority**) record is always behind the directives and proclaims relevant information about the authority of a domain to the name server. It is always the first resource in a zone file. The "@" *symbol* (at) is equivalent to the $ORIGIN directive (or the name of the zone if such a directive has not been used - most commonly) as the domain namespace defined by this record. This would be the skeleton of this record:

   ```
           @ IN SOA <primary-name-server> <hostmaster-email> (
             <serial-number>
             <time-to-refresh>
             <time-to-retry>
             <time-to-expire>
   <minimum-TTL> )
   ```

   The primary nameserver that is the authoritative one for this domain is used in `<primary-name-server>` and the email address of the person to be contacted about this **namespace** is replaced with `<hostmaster-email>` (note that it does not have to correspond to an address of the domain itself).

   The `<serial-number>` field is a number that is incremented each time a file in a zone is modified, so that Bind realizes that it has to reload this zone. It is recommended to use the modification date in *YYYYMMDD format*, where *YYYYY* is the year in four-digit format, *MM* is the month in two digits, and *DD* is the day of the month in two digits, followed by a two-digit number, starting with 01. In this way, up to one hundred changes can be made per day. The `<time-to-refresh>` field tells secondary (slave) servers how long they should wait before asking their primary (master) server if any changes have been made to the zone. The value of the `<serial-number>` field is used by slaves to determine if outdated information is being used that needs to be updated.

   The `<time-to-retry>` field specifies to slave servers the time interval to wait before requesting an update in the event that the primary name server is not responding. If the master server has not responded to the update request before the `time-to-expire<>` field expires, the slave will no longer act as the authorized server of that namespace (zone). The `<minimum-TTL>` field prompts other domain servers to cache information for this zone for at least the amount of time specified in it.

   Note that the field `<primary-name-server>` ends in a period, which is mandatory to enter, and which represents, as explained in the introductory section of the article, the root name server. Likewise, this point will appear in all explicit references to the domain throughout the file. When a host or subdomain, such as *ftp, is configured*, an implicit reference is made and Bind automatically adds the domain, which it pulls out of the *"@"* of the SOA record. In any case, it is possible to use implicit or explicit references interchangeably.

3. `NS ns1.linuxsilo.net.` and `NS ns2.linuxsilo.net.`: Indicate the name servers that have authority over the domain. Note that the at sign saves us from having to type the name of the full domain. In fact, the prefix, *IN* is also dispensable. This bypass is made possible by Bind taking the omitted features from the previous SOA register, i.e., *@IN.* Of course, both ways are correct.

4. `MX 20 ns1.linuxsilo.net.`: This is an MX (**Mail eXchanger**) record and indicates where to send mail destined for a namespace controlled by this zone. The digit following the word *MX* represents the priority over other MX records for the area, which would be specified on subsequent lines (`MX 30 ns2.linuxsilo.net.`), following the same format but varying that digit (increasing it as they lose priority compared to previous records). That is, the lower the preference value, the higher priority it acquires.

5. `TXT "LinuxSilo.net DNS server"`: This is a descriptive, **plain text** record for the server. It can be used freely and arbitrarily for various purposes. It will appear as a result of a query about this type of record made to the name server about this zone.

6. `HINFO "Intel Pentium IV" "Debian Linux"`: another record, also for information purposes and totally optional (**Host INFOrmation**), whose purpose is to inform about the hardware and operating system, in this order, delimited by double quotation marks and separated by a space or tab, of the machine on which the name server runs. Both this type of record (HINFO) and the previous one (TXT) can be used in each of the subdomains (not only in the main domain of the area), as will be seen below.

7. `LOC 39 34 58 N 2 38 2 E 100m 10000m 20m 100m`: server geolocation log, again optional, which is used by server location graphing tools, such as those of the Cooperative Association for Internet Data Analysis (CAIDA)) and others. Information about this type of registration can be found in the RFC1876. The coordinates (latitude, longitude and diameter of the object) are in WGS-84 format (**World Geodetic System**, 1984). The location used in the article corresponds to Palma, Mallorca, Balearic Islands, Spain.

The format to follow is as follows: `<owner><TTL><class> LOC ( d1 [m1 [s1]] {"N"|" S"} d2 [m2 [s2]] {"E"|" W"} alt["m"] [siz["m"] [hp["m"] [vp["m"]] )`. Where:

Parameter	Meaning	Unit	Values	Comment
D1	Latitude (degrees)	º	0..90	Portion in latitude degrees
M1	Latitude (minutes)	'	0..59	Latitude portion in minutes. If omitted, 0' is taken by default
S1	Latitude (seconds)	"	0..59,999	Portion in seconds of latitude. If omitted, 0" is taken by default
N/S	Latitude (hemisphere)		N/S	North/South Terrestrial Hemisphere
D2	Length (degrees)	º	0..180	Lot in Degrees of Length
m2	Length (minutes)	'	0..59	Length serving in minutes. If omitted, 0' is taken by default
S2	Length (seconds)	"	0..59,999	Length portion in seconds. If omitted, 0" is taken by default
E/W	Longitude		E/W	Longitude E=east/W=west
Alt	Altitude	m	-100000.00 .. 42849672,95	Altitude with accuracy of 0.01 m.
Yes,	Size	m	0..90000000,00	Diameter of the dial containing the indicated point. If omitted, 1 m is taken by default.
Hp	Precision horizontal	m	0..90000000,00	Horizontal accuracy in meters. If it is omitted, 10,000 m is taken by default.
Vp	Vertical Reccission	m	0..90000000,00	Vertical accuracy in meters. If omitted, 10 m is taken by default.

8. `localhost A 127.0.0.1`: A log that relates the local host to its loopback IP.
9. `linuxsilo.net. A 66.79.182.201`: A record that relates the second-level domain name (the "primary" domain name in the zone) to the IP where it is hosted. This is the most commonly used register, as any request to *linuxsilo.net* will be resolved through this register, regardless of the communications protocol used (e.g. *http://linuxsilo.net*).
10. `ns1 A 66.79.182.201`: from here begins the translation of subdomains of the domain for which we are authorized: third-level and successive domains. Note that a record must be created for each one, without the possibility of "grouping" in any way. Also, note that, as they are subdomains of the area, the suffix linuxsilo.net has been omitted ., which is implied because it does not end in *"."* (period). It is simply a question of clarity and space-saving, because the representations in both areas are - we repeat again - equally correct. Other similar records are cited, grouped below:

```
        ns2 to 213.96.79.79
             TXT "LinuxSilo.net secondary
nameserver" HINFO "Intel Pentium MMX" "Debian
Linux" www A 66.79.182.201 pop3 A
66.79.182.201 smtp A 66.79.182.201 ftp A
66.79.182.201 ts A 213.96.79.79
             TXT "LinuxSilo.net Team Speak server"
             HINFO "Intel Pentium MMX" "Debian Linux"
```

The reader will notice that two different IP addresses have been used, which would indicate a priori that, in reality, all these hosts (third-level domains) are located on only two different machines. But this does not have to be true, because you could have the same public IP but several machines serving the different ports used in these services, thanks to the action of a router.

Regarding the concept of *aliases* (`www`, `pop3`, `smtp` and `ftp` are in fact the same host) there is a controversial discussion about whether it is better to use the *CNAME* (**Canonical NAME)** or *IN A* record type. Many Bind gurus recommend not using *CNAME* records at all, although that discussion is beyond the scope of this article. In any case, it is highly recommended to follow the rule that *MX,* CNAME*,* and *SOA* records should never reference a *CNAME* record, but only something with an *"A" type record*. Therefore, it is not advisable to use:

```
        CNAME www
```

But it would be correct:

```
        web CNAME ns
```

It's also safe to assume that a *CNAME* isn't a suitable host for an email address: *webmaster@www.linuxsilo.net*, it would be incorrect given the configuration above. The way to avoid this is to use *"A"* records (and perhaps some others as well, such as the *MX* record) instead. The author of this article favors the use of *IN A* and recommends this practice.

Reverse translation.

By now, programs are already able to convert names into *linuxsilo.net* and *balearikus-party.org* to addresses to which they can connect. But a reverse zone is also required, capable of allowing the DNS to convert an address into a name. This name is used by many servers of different classes (FTP, IRC, WWW and others) to decide whether they want to "talk" to the client or not and, if so, perhaps even how much priority should be assigned to it. In order to have full access to all these services on the Internet, a reverse zone is necessary.

In the */etc/bind/named.conf* file we find several reverse zones that come by default with the installation, just below two comment lines like these:

```
    be authoritative for the localhost forward and reverse zones,
and for // broadcast zones as per RFC 1912
```

There we can find the reverse zone translation for `localhost`, `127.in-addr.arpa`, `0.in-addr.arpa` and `255.in-addr.arpa`, which does not need to be modified at all except in the first case. After them we must add our area: `38.127.217.in-addr.arpa` (remember that they are written in reverse order, as explained in the introductory section of this article):

```
    zone "38.127.217.in-addr.arpa"
{ type master;
      file "/etc/bind/db.217.127.38";
    };
```

The syntax is identical to that used in the name translation areas explained in the previous point, and the previous comments remain valid here. Let's move on to the contents of the */etc/bind/db.217.127.38 file*:

```
    ;
    ; BIND reverse data file for zone 217.127.38
    ;
    $TTL 604800
    @ IN SOA linuxsilo.net. hostmaster.linuxsilo.net. (
      2001081501   ; Serial
           10800   ; Refresh (3 hours)
            7200   ; Retry (2 hours)
         1296000   ; Expire (15 days)
          172800 ) ; Negative Cache TTL (2 days)

    @ IN NS ns1.linuxsilo.net.
           NS ns2.linuxsilo.net.
    156 IN PTR ns1.linuxsilo.net.
```

This would be what the localhost inverse zone would look like, which we will have to modify slightly from the original:

```
    ;
    ; BIND data file for local loopback
interface ;
    $TTL 604800
    @ IN SOA localhost. hostmaster.linuxsilo.net. (
      2001061501   ; Serial
          604800   ; Refresh
        86400   ; Retry
      2419200   ; Expires
       604800 ) ; Negative Cache TTL

      IN NS ns1.linuxsilo.net.
    1 IN PTR localhost.ns1.linuxsilo.net.
```

The reverse mapping of the local host address (127.0.0.1) never changes, so the times between changes are long. Note the serial number, which encodes the date: the file was last changed during the summer of 2001, when the server was created. Note also that only the master server is listed in the *localhost domain*. The value of *"@"* here is *0.0.127.in-addr.arpa.*

Again, the concepts are the same (the *"@"* - at - indicates the domain of the linuxsilo.net zone., the *"."* - period - at the end refers to the root name server and the *"SOA"* record has exactly the same structure and functionality), except for the last two lines:

1. `@ IN NS ns1.linuxsilo.net.` and `NS ns2.linuxsilo.net.`- Indicate which name servers should be asked for the reverse translation of an IP address in this zone.

2. `156 IN PTR ns1.linuxsilo.net.`: this is the record that will be used to return the name that we want to correspond to the IP address that belongs to us (be careful when creating these records, as we must refer exclusively to IP addresses that are our property or we would cause a conflict). In this case, it is indicated that the address 156 (implicitly the suffix *.38.127.217.in-addr.arpa*, indicating that it is "our" IP address *66.79.182.201*) is equivalent to the *host ns1.linuxsilo.net*.

   It is obvious that "information is missing" here, since the IP address *66.79.182.201* is actually equivalent to more hosts, as we have specified in the *file /etc/bind/db.linuxsilo.net*. This is true, but the author is of the opinion that it is redundant to add lines of the style:

   ```
       156 IN PTR ftp.linuxsilo.net.
       156 IN PTR pop3.linuxsilo.net.
       156 IN PTR smtp.linuxsilo.net.
       156 IN PTR
   www.linuxsilo.net.     [..]
   ```

   That is, it is considered more appropriate to specify a single FQDN per IP. Of course, if you had a range of IP addresses, for example from *66.79.182.201* to *217.127.38.160*, inclusive, records similar to the following would appear (they would vary depending on the specific case):

   ```
   156 IN PTR ns1.linuxsilo.net.
   ```

```
157 IN  PTR  ftp.linuxsilo.net.
158 IN  PTR  smtp.linuxsilo.net.
159 IN  PTR  ssh.linuxsilo.net.    160  IN  PTR  www.linuxsilo.net.
```

For this example, it follows that the zone *linuxsilo.net.* it is divided into five different machines, one for each of the aforementioned services (NS, FTP, SMTP, SSH and WWW, respectively).

Why doesn't reverse translation work? There are a number of "special attentions" to pay at this point that are often overlooked when setting up such a nameserver. Two common errors in reverse translations are discussed below:

1. **The reverse zone has not been delegated**. When a range of IP addresses and a domain name are requested from a service provider, the domain name is usually delegated as a rule. A delegation is the specific name server that allows you to jump from one name server to another, as explained in the introductory section of this article.

   The reverse zone must also be delegated. If the *217.127.38 network is obtained* with the domain *linuxsilo.net* through a provider, the provider must add an *NS* record for our reverse zone as well as for our direct zone. If you follow the chain from *in-addr.arpa* upwards to reach our network, you will probably find a fracture in the chain, most likely at the height of our service provider. Having found the broken link, contact your service provider and ask them to correct the error.

2. **Its subnet does not belong to a defined class**. This is a more advanced concept, but **classless subnets** are very common these days, and you probably have one if you're a small business.

   A classless subnet is what keeps the internet running today. A few years ago there was a lot of discussion about the lack of IP addresses. The brains of the Internet Engineering Task Force (**IETF**), which keeps the Internet running, squeezed their heads and found the solution to the problem, albeit at a certain cost. The price is that you get less than a "C" type subnet and some things may stop working.

   The first part of the problem is that your ISP must understand the technique used. Not all small service providers have a working knowledge of how they work, so you may need to explain it to them and be somewhat insistent (even if you make sure you understand it first). Then, they will need to prepare a reverse zone on their server whose correctness can be checked using the *dig* utility of the *dnsutils package*.

   The second and final part of the problem is that you must understand the problem and its solution. If you're not sure, pause here and find out more about it. Only then, should you configure your reverse zone for your classless network.

   But there is yet another hidden trap in this concept. Older domain name servers will not be able to follow the *CNAME* record in the translation chain and will fail in the reverse translation of your machine. This can result in a service being assigned an incorrect access class, a denial of service, or something in between. If you find yourself in this case, the only solution is for your ISP to directly insert your *PTR* record directly into your classless network zone instead of using *CNAME records*.

   Some ISPs offer a variety of alternatives to address this issue, such as web forms that will allow you to enter your back-translation record map, etc.

Secondary servers

Once the zones have been correctly configured on the main (master) server, it is necessary to prepare at least one secondary (slave) server, which will provide robustness and reliability. If the master server goes down, users will still be able to get information from the slave about the zones being represented. The slave server should be as far away from the master as possible, and both should share as few of the following characteristics as possible: power supply, local area network (LAN), ISP, city, and country. If all of them are different between the master and the slave, then you have a really good secondary server.

A slave server is simply a name server that replicates files from a master's zones. They are configured as follows:

```
    zone "balearikus-party.org"
{ type slave;
      file "sec.balearikus-
party.org";       allow-query {
any; };       masters {
66.79.182.201; };
    };

    zone "linuxsilo.net"
{ type slave;
      file
"sec.linuxsilo.net";
allow-query { any; };
masters { 66.79.182.201; };
};
```

Note that the structure is the same as for the primary server, changing only a few parameters:

1. `type slave;`: indicates that the server is a slave to this zone.

2. `file "sec.balearikus-party.org";` and `file "sec.linuxsilo.net";`: As explained in the introduction to the article, in order to follow the Debian directory policy, temporary files in zones automatically generated by the secondary server must be saved in the default directory
   /var/cache/bind, so only files are specified (without a path, or with an implicit relative path, which is the same thing). See the next point for more information on the contents of these files.

3. `allow-query { any; };`: same concept as on the primary server.

4. `masters { 66.79.182.201; };`: defines which server is the master for this zone (of which, remember, one is a slave). An ACL could have been used here, in the same way as it is done in the master's */etc/bind/named.conf*, but it has not been considered appropriate since there is a single master for both zones. However, if the reader must manage a network of name servers, where the role of master and slave is played at the same time by the same host depending on the zone, then it would be very convenient to create several ACLs, so as to facilitate maintenance and control in the assignment of master and slaves for each zone.

The other configuration options would be used identically to the master server, as long as the conditions are the same. That is, the same policies apply (for example, *options*, in which we would include the *forwarders option*) and possibilities.

Finally, we would like to highlight this aspect that should not be overlooked: reverse zones, although special, are also zones and must be transferred from the primary server to the secondary ones. At this point, the one that until now was the primary server also becomes the secondary server, since *ns2.linuxsilo.net is the* master of its reverse zone (*79.96.213.in-addr.arpa*), which it will transfer to *ns1.linuxsilo.net*, making it a slave only for that zone. In the same way, *ns1.linuxsilo.net* will act as the master of its reverse zone (*38.127.217.in-addr.arpa*), which it will transfer to *ns2.linuxsilo.net* as has been happening with the *balearikus-party.org* and *linuxsilo.net zones*. Below are the changes to the configuration files. In the *named.conf* of *ns1.linuxsilo.net*:

```
    zone "38.127.217.in-addr.arpa"
{ type master;
      file "/etc/bind/db.217.127.38";
allow-transfer { slaves; };
    };

    zone "79.96.213.in-addr.arpa"
{ type slave;
      file
"sec.db.213.96.79";
masters { 213.96.79.79; };
};
```

And in *ns2.linuxsilo.net*'s *named.conf*:

```
    zone "79.96.213.in-addr.arpa"
{ type master;
      file "/etc/bind/db.213.96.79";
allow-transfer { 66.79.182.201; };
    };

    zone "38.127.217.in-addr.arpa"
{ type slave;
      file
"sec.db.217.127.38";
masters { 66.79.182.201; };
};
```

The content of the zones would remain exactly the same. After these changes, *the file* sec.db.213.96.79 *would appear in the /var/cache/bind* directory of *ns1.linuxsilo.net* and the file sec.db.217.127.38 *would appear* in the same directory of *ns2.linuxsilo.net*, all thanks to the automatic transfer of zones that we see below.

Secure zone transfer.

The reader will have noticed that nothing has been commented on the *sec.balearikus-party.org* and *sec.linuxsilo.net files* specified in the zone directives of the secondary server. This is because we will use a procedure that will allow those files to be created in an automated way from those we create on the primary server, so that maintenance tasks are greatly facilitated.

To do this, the `allow-transfer { slaves; };` and `masters { 66.79.182.201; };` in the zones defined in the */etc/bind/named.conf* files of the primary and secondary servers, respectively. This will allow that, once the desired changes have been made to the */etc/bind/db.balearikus-party.org* or */etc/bind/db.linuxsilo.net* file, including the increase of the identification serial number of the SOA record, and having instructed the name server to reload one, several or all of the zones, these changes will be reflected in the secondary file so that the corresponding */var/cache/bind/sec.balearikus-party.org files are generated* and */var/cache/bind/sec.linuxsilo.net*.

In order for this transfer of zones to be done in a secure and controlled way, we will impose certain restrictions on the */etc/bind/named.conf* and generate keys that will ensure privacy in communication. Here are the lines we'll add in the */etc/bind/named.conf* of the primary server (*66.79.182.201* in the example):

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

"2002052101.linuxsilo.net.tsigkey.";
};     };

    Server 213.96.79.79
{ Keys {

"2002052101.linuxsilo.net.tsigkey.";
};     };
```

And these are the ones that we will add in the */etc/bind/named.conf* of the secondary:

```
    controls { inet
127.0.0.1 allow {
        127.0.0.1;
      }
keys {
        "2002052101.linuxsilo.net.tsigkey.";
      };
};

    server 66.79.182.201
{ keys {

"2002052101.linuxsilo.net.tsigkey.";
};     };
```

The meaning of both directives is explained below:

1. `controls { inet 127.0.0.1 allow { 127.0.0.1; } keys { "2002052101.linuxsilo.net.tsigkey."; } };` It is the policy that tightens control over the server through the key `2002052101.linuxsilo.net.tsigkey.` only to the local host. That is, we must have connected (usually remotely via SSH) to the server and, from there, execute the commands that control Bind's actions (usually using the *rndc utility*, which will be explained later). From this it can be deduced that both the remote transfer of zones and the control over the server (zone reloading, stopping, starting, etc.) is carried out through this encrypted key. In addition, it also follows that this restriction will not allow Bind to be controlled remotely, as mentioned above. This is the default option and the one that the author of this article recommends.

2. `Server 213.96.79.79 { keys { "2002052101.linuxsilo.net.tsigkey."; }; };` is a policy that tells the server when the key should be used. By using this clause, the server is obliged to use a certain key when communicating with a certain IP address. For each server it is convenient to specify a *server* directive, specifying the IP address of the other machine and the name of the key to be used. The example uses the same key for communication between servers and for server control from the local host - having accessed via SSH - using the *rndc utility*

Note that if the key is changed, the *rndc* tool may stop working properly. When performing this process, it is recommended for the server (*rndc stop* or */etc/init.d/bind9 stop*), to replace the keys and start it again (*/etc/init.d/bind9 start*). Before explaining how to create such a key, let's look at the real reason for its need and the problems that a security breach could cause.

About ports, Bind uses TCP 53 for transfers and UDP 53 for queries.

What is a TSIG and what is it needed for?

The DNS works on a question-answer model. If a client needs DNS information, it sends a request to the DNS server, which returns a response. Until recently, it was only possible to rely on the source IP address to discern whether or not to answer a query. But this is not exactly "ideal". Authentication based solely on the source IP address is considered insecure. Signed Transactions (TSIG)
SIGnatures) add cryptographic signatures as an authentication method in a DNS conversation. A shared secret key is used to establish trust between the parties involved.

TSIG are used to ensure that DNS information purporting to come from a certain server is actually from that server. They are primarily used for authentication in the transfer of zones between the primary and secondary name servers. It is intended to ensure that secondary servers will never be tricked into accepting a copy of a zone for which it is authorized by an imposter listening on the IP address of the primary server.

Signed transactions are defined in the RFC2845.

In the example above, the *key tsigkey.linuxsilo.net.20010922* was used to authenticate DNS traffic between the two servers, the primary (*66.79.182.201*) and the secondary (*213.96.79.79).*

Creating a TSIG Key

In the default installation of the Debian package, a pre-generated and fully functional TSIG key is provided. But it is, of course, the same for everyone who installs that package. Therefore, it is more than advisable to change it. Here's how to generate a particular key and how to use it to make zone transfer secure.

TSIG uses a shared secret key that is incorporated into an MD5 hash of the information to be signed. Bind comes with a tool for creating such keys, called *dnssec-keygen*, whose parameters are numerous (run *dnssec-keygen --help* to see the full list and *man dnsseckeygen* for the *man* page about this utility). Here are the steps to quickly create a key:

keygen for the *man* page about this utility). Here are the steps to quickly create a key:

1. By running the command `dnssec-keygen -a HMAC-MD5 -b 512 -n HOST 2002052101.linuxsilo.net.tsigkey.` A key named *2002052101.linuxsilo.net.tsigkey* is created.using the *HMAC-MD5* algorithm, 512-bit (**BInary digiT**) and *HOST* type (which is precisely the use for which it is intended).

2. Of the two files generated, `K2002052101.linuxsilo.net.tsigkey.+157+30191.key` and `K2002052101.linuxsilo.net.tsigkey.+157+30191.private`, only the second one will be used. It is used to mention that the output format of the names of the generated files is *Knnnn.+yyy+iiiii*, where *nnnn* is the name of the key, *aaa* is the numerical representation of the algorithm and *iiiii* is the mark/fingerprint of the key identifier (from English, **footprint**). Of course, the names of each generated key must be unique, that is, two keys should never share the same name, hence the unusual name that has been given to it. The key, properly speaking, is the set of characters that is after the word *Key:* in the last line of the file with a *.private suffix*.

3. The next step is to edit the */etc/bind/rndc.key* file and replace the default key with the one that has just been generated. To do this, it is enough to change the name of the default key from *"rndc.key"* to the one we have given it when we created it, in this case *2002052101.linuxsilo.net.tsigkey.*. Finally, you have to change the value of the *secret* field to the value of the generated key which, as just said above, is after the word *Key:* in the last line of the file ending in *.private*. In the example case used for the article, the key

wlnQbRQM/76rol0xGkEdm [..] MMlUFR7HpenQ==, but the reader should not take this as a reference, as it will vary in each new generation. This is the contents of the file *K2002052101.linuxsilo.net.tsigkey.+157+30191.private*:

```
        Private-key-format: v1.2
        Algorithm: 157 (HMAC_MD5)
        Key: wlnQbRQM/76rol0xGkEdm [..] MMlUFR7HpenQ==
```

This is the contents of the */etc/bind/rndc.key file* that comes by default with the installation of the Debian package:

```
        Key "rndc-key" {
algorithm hmac-md5;
          secret
"bsty5LYDsO8infm+n2JNsw==";         };
```

And this is, finally, the */etc/bind/rndc.key file* resulting from the use of the new key:

```
        key "2002052101.linuxsilo.net.tsigkey." {
algorithm hmac-md5;
          secret "wlnQbRQM/76rol0xGkEdm [..]
MMlUFR7HpenQ==";         };
```

The two files created when generating the key, ending in *.key* and *.private*, can be deleted without problems. The reader, during his tests, will notice that, if the *"."* At the end of the key name, *DNSSEC-Keygen* will automatically add it, which can create confusion about whether the name that corresponds to the generated key is with a period or without a period at the end. At this point, the author recommends using the name as it was generated, although with a simple trial and error test (checking that the zones are transferred) you can easily arrive at the correct solution.

Finally, it should be noted that a secret key is just that: secret. Therefore, it must be copied and installed on both servers in a secure way. In addition, it is strongly recommended to change the permissions of the */etc/bind/named.conf* and */etc/bind/rndc.key* files, both on the primary and secondary servers, to 600 (*chmod 600 /etc/bind/named.conf* and *chmod 600 /etc/bind/rndc.key*), so that they are only accessible by the *root user*.

The verification of a TSIG requires the possibility of writing a file temporarily. Make sure that *named* has write permissions to your directory by default (*directory* clause of the *options* directive, which in Debian is by default */var/cache/bind/*). Microsoft's implementation of TSIG does not use the RFC2845 algorithm (HMAC-MD5). Microsoft's GSS-TSIG does not meet the standard and consequently will not properly interoperate with Bind.

More information in documents RFC2535, RFC2845 and RFC2539

Feedback on dynamic update, TSIG security, and ACLs

1. It is critical that the key is kept secret, which means, for example, that:
   to. *named.conf* and *rndc.key* must not have read permissions for anyone other than *Named* or the user running *RNDC* or *NSupdate*. b. The key must not be transmitted by emails, unless they are encrypted.
   c. Anyone to whom you give this key is trustworthy: therefore, give it exclusively to those who need it, and never to people you distrust.
   d. You should consider changing passwords from time to time, after changes in personnel, or if you suspect that secrecy may have been compromised.

2. If both hosts are on the same subnet, it is more difficult to **spoofing** the IP address than it is to get a copy of the key (for example, if routers in contact with the outside are leaking monitored IPs), so an IP address ACL would be more effective.

3. It is equally valid to specify an ACL or a TSIG key. For example , `allow-update {key updater; updaters; };`, which would mean that both the TSIG and the IP address ACL are valid for updates.

4. It is not possible to require both TSIG and IP access control. Bind's developers don't think this is useful, as they focus on user-level rather than host-level control of dynamic updates (hence the emphasis on the new update policy that allows users with dynamic IP addresses to update their DNS records).

5. Dynamic updates cannot add or remove domains, only records from those domains.

6. A client host that wants to update a Bind server only needs the *nsupdate* binary and the appropriate key. No additional binaries or libraries are required.

7. *NSupdate* supports the *"-D"* parameter for **debugging** tasks.

8. *nsupdate* can also use *TCP* (**Transmission Control Protocol**) instead of *UDP* (**User Datagram Protocol**) for updates ("*-v"* parameter), which provides better performance if there are many updates to be made and greater security since *TCP* is a connection-oriented protocol. In addition, a *TCP* connection has the possibility of being routed through an SSH *(*Secure SHell) **tunnel** for added security (encryption and access control).

9. The Update Policy is a new feature in Bind 9 that allows updates to be restricted to certain specific names. By

   For example, to allow an *ADSL* (**Asymmetric Digital Subscriber Line**) or *DHCP* (**Dynamic Host Configuration Protocol**) user to update their own hostname (i.e., those to which they change IP addresses). With this update policy, you can configure a list of keys per host and allow each key to update only the host or associated zone.

Risks to which an insecure Bind exposes

Is it really necessary to worry about DNS as well? Well, a compromised DNS can expose itself to some interesting risks:

1. An attacker can obtain very interesting information if zone transfers are allowed: the complete list of hosts and routers with their IP addresses, names and, possibly, comments indicating their situation, etc.

2. **Denial of service**: If all of your DNS servers go down,
    - Your website is no longer visible (other websites cannot translate your IP address).
    - Emails can no longer be sent (some sites on the Internet with which information is exchanged will often have cached DNS records, but that won't last more than a few days).
    - An attacker could start a fake DNS server that pretends to be yours and sends fake DNS information to the Internet about your domain. That is, loss of integrity - see the next section.
3. Loss of integrity: If an attacker can change DNS data or provide (spoofing) false information to other sites (this is known as DNS **poisoning**), the situation becomes very tricky:
    - Fake your website to look like your own and capture user posts that were destined for your site, so you're talking about stealing anything from **logins** and **passwords** to credit card numbers.
    - All mail may be redirected to a **relay** server that may copy, change, or delete mail before passing it to your site.
    - If your firewall or any host accessible from the Internet uses DNS hostnames for authentication or for trust relationships, these can be completely compromised, especially if a weak packet filter is protecting the Internet servers and the Intranet. Imagine a web proxy configured to allow proxy requests only from **.midominio.com*. The attacker adds their host to the domain, so the web proxy now allows requests from it, allowing the attacker HTTP access to the Intranet. Imagine a sysadmin using SSH (great cryptographic invention), but firewall hosts have a *.shosts* relying on *admin.midominio.com*, where *admin* is the administrator's workstation. If the attacker can substitute the admin.midominio.com entry in the DNS, they have free, passwordless access to firewall hosts.

DNS has become a favorite target for hackers, as evidenced by the tools for automatic attacks and worms that use DNS flaws that appeared during the winter of 2001.

So, what measures need to be taken?

The risks of Bind can be greatly reduced with a few prevention measures:

1. Resource isolation: Use a dedicated and secured server for the Internet's DNS, do not share it with other services and, especially, do not allow remote user access. Minimizing services and users means reducing the amount of software running, and therefore the likelihood of being exposed to network attacks. The separation prevents against the possibility of other services or users locating weaknesses in the system and using them to attack Bind.
2. Redundancy: Install a secondary on a different Internet connection (branch far from your company, another ISP, etc.). If your site goes down, at least the rest of the sites won't think you've "ceased to exist," but will just think you're "unavailable," so, for example, your emails won't get lost but will go into a queue (typically up to four days).
3. Use the latest version.
4. Access Control: Restrict zone transfer to minimize the amount of information available on your network to attackers. Consider using signed transactions. Consider restricting or not allowing recursive queries.
5. Run Bind with least privileges: as a non-root user, with a very restrictive *umask* (e.g. *177*).
6. Increased resource isolation: Run Bind in a **chroot** (jail*) environment*, so that it is much more difficult for a compromised Bind daemon to damage the operating system or compromise other services.
7. Configure Bind to not report your version. Some people don't believe this measure, as it's "security by concealment," but understand that it will at least help against youngsters with scripts that scour the web for obvious targets. Defending yourself from professionals is another matter.
8. Detection: Monitor logs for unusual activity and unauthorized changes to the system using an integrity analyzer.
9. Stay continuously up to date with news and make sure you are notified of new Bind issues in a reasonable amount of time.

Recursive and non-recursive servers.

Name servers can act recursively or disallow it. If a non-recursive server has the response to a cached request from a previous transaction or is the authorized one of the domain to which the query belongs, then it provides the appropriate response. Otherwise, instead of returning an actual response, it returns a reference to the authoritative server in another domain that is better able to know the response. A client on a non-recursive server must be prepared to accept referrals and act accordingly.

While non-recursive servers may seem lazy, they usually have a good reason to get rid of the extra work. The root servers and the highest level servers are all non-recursive, but 10,000 queries per second is an excuse to be one.

A recursive server returns only real responses or error messages. It takes care of following the references by itself, relieving the client of that task. The basic procedure for translating a query is essentially the same; The only difference is that the name server is concerned with taking over the references instead of returning them to the client.

Location of services

An SRV record specifies the location of the services offered by a domain. For example, the SRV record allows you to query a remote domain directly and ask for the name of its FTP server. Until now, in most cases, you had to try your luck. To contact the FTP server of a remote domain, one expected that the system administrator of that domain would have followed the current standard (the taste rather) and had a *CNAME* for *ftp* on their DNS server.

SRV logs become very important in these types of queries and are really a better way for system administrators to move services and control their usage. However, they must be explicitly requested and analyzed by customers, so their effects will be seen gradually as time goes by.

SRV records resemble generalized MX records with fields that allow the local administrator to guide and load balance connections coming from the outside world. The format is

```
    service.proto.name[ttl] IN SRV pri wt port destination
```

where *service* is one of the services defined in the number database assigned by the IANA, *proto* can be *tcp* or *udp*, *name* is the domain to which the service refers, *pri* is a priority in the style of MX records, *wt* is the weight used to balance the load between different servers, *Port* is the port on which the service listens, and *destination* is the hostname of the server on which that service is provided. The target's A record is usually returned automatically along with the response sent to an SRV query. A value *of "0"* for the wt parameter means that no special type of load balancing is performed. A value of *"."* for the target means that the service is not running at that site.

In the *linuxsilo.net area* of the example, adapted from the RFC2052 (where SRV is defined), we have the following:

```
    ftp.tcp SRV 0 0 21 ftp.linuxsilo.net.
    ssh.tcp SRV 0 0 22 linuxsilo.net.
telnet.tcp SRV 0 0 23 linuxsilo.net.
    smtp.tcp SRV 0 0 25 smtp.linuxsilo.net.

    ; 3/4 of the connections to the primary, 1/4 to the
secondary http.tcp SRV 0 3 80 linuxsilo.net.
    http.tcp SRV 0 1 80 ns2.linuxsilo.net.
    ; to work both http://www.linuxsilo.net and http://linuxsilo.net
http.tcp.www SRV 0 3 80 linuxsilo.net.
    http.tcp.www SRV 0 1 80
ns2.linuxsilo.net.
    ; primary server on port 443, secondary - in case of failure - on another machine and
another port https.tcp SRV 1 0 443 linuxsilo.net.
    https.tcp SRV 2 0 4443 ns2.linuxsilo.net.
    https.tcp.www SRV 1 0 443 linuxsilo.net.
    https.tcp.www SRV 2 0 443 ns2.linuxsilo.net.
pop3s.tcp SRV 0 0 995 pop3.linuxsilo.net.

    *.tcp SRV 0 0 0 .     *.udp
SRV 0 0 0 .
```

This example illustrates the use of both the *wt* parameter (**weigth**) for HTTP and the priority parameter for HTTPS. Both HTTP servers will be used, splitting the work between them. The secondary server *will ns2.linuxsilo.net* only be used for HTTPS when the primary server is unavailable. All unspecified services are excluded. The fact that the daemon of, for example, *finger* does not appear in the DNS does not mean that it is not running, but only that this service cannot be located through DNS.

Microsoft uses the standard SRV records in Windows 2000, but inserts them into the DNS system in an incompatible and undocumented manner.

Types of DNS records

	Guy	Name	Function
Zone	SOA	Start Of Authority	Define a DNS representative zone
	NS	Name Server	Identify zone servers, delegate subdomains
Basic	To	IPv4 address	Translation of name to address
	AAAA	Original IPv6 address	Currently obsolete
	A6	IPv6 address	Name-to-IPv6 address translation
	PTR	Pointer	Translation from address to name
	DNAME	Redirection	Redirection for IPv6 Reverse Translations
	MX	Mail eXchanger	Control mail routing
Safety	KEY	Public Key	Public key for a DNS name
	NXT	Next	Used in conjunction with DNSSEC for negative responses
	GIS	Signature	Authenticated/Signed Zone
Optional	CNAME	Canonical Name	Nicknames or aliases for a domain
	LOC	Localization	Geographical location and extension
	RP	Responsible person	Specify the contact person for each host
	SRV	Services	Provides localization of known services
	TXT	Text	Unencrypted comments or information

Views

Views are a new feature in Bind 9 that allows internal machines to be shown a different view of the DNS name hierarchy than the one seen from the outside (meaning "inside" and "outside" with respect to the router that gives the company access to the Internet). For example, it allows you to reveal all hosts to internal users but restrict the external view to a few trusted servers. Or you could offer the same hosts in both views but provide additional (or different) records to internal users.

This type of configuration (sometimes called "**split DNS")** is becoming very popular. In the past, it was implemented by setting up separate servers for the internal and external versions of reality. Local clients pointed to distribution servers that contained the internal version of the zone, while NS records in the parent zone pointed to servers running the external version. The `view` statement in Bind 9 simplifies configuration by allowing you to have both datasets together in the same named copy. *Named* searches for matches in address lists to guess which customers should receive which data.

The `view` statement packages an access list that controls who sees the view, some options that apply to all zones in the view, and eventually the zones themselves. The syntax is:

```
    view "view-name" {
      match-clients {
address_match_list; };       view-
option; ...       zone-sentence; ...
    };
```

The `match-clients` clause controls who can see the view. Hearings are processed in sequential order, so the most restrictive ones should go first. Areas in different views can have the same name. Views are an all-or-nothing proposition; If you use them, all `zone` statements in your *named.conf* file must appear within the context of a view.

Here's the example for the *linuxsilo.net* and *balearikus-party.org* domains, created from the Bind 9 documentation that follows the split DNS schema described above. The two views define both zones, but with different registers.

```
    ACL "LAN" {
192.168.0.0/24;
    };

    View for all computers on local area network

    view "internal" { match-
clients { LAN; };
recursion yes;

      be authoritative for the localhost forward and reverse
zones, and for // broadcast zones as per RFC 1912

      prime the server with knowledge of the root servers

      zone "." {
type hint;
        file
"/etc/bind/db.root";       };

      All other default reverse zones omitted for short

      zone "38.127.217.in-addr.arpa"
{ type master;
        file "/etc/bind/db.217.127.38";
allow-transfer { slaves; };
      };

      zone "79.96.213.in-addr.arpa"
{ type slave;
        file "sec.db.213.96.79";
masters { 213.96.79.79; };
      };

      zone "0.168.192.in-addr.arpa"
{ type master;
        file
"/etc/bind/db.192.168.0";       };

      add entries for other zones below here

      zone "balearikus-party.org"
{ type master;
        file "/etc/bind/db.balearikus-party.org.internal";
      };

      zone "linuxsilo.net"
{ type master;
        file
"/etc/bind/db.linuxsilo.net.internal";
};
```

```
    };

    View for all computers outside the local area

network view "external" {

      match-clients { any; };
recursion does not;

      be authoritative for the localhost forward and reverse
zones, and for // broadcast zones as per RFC 1912

      prime the server with knowledge of the root servers

      zone "." {
type hint;
        file
"/etc/bind/db.root";       };

      All other default reverse zones omitted for short

      zone "38.127.217.in-addr.arpa"
{ type master;
        file "/etc/bind/db.217.127.38";
allow-transfer { slaves; };
      };

      zone "79.96.213.in-addr.arpa"
{ type slave;
        file
"sec.db.213.96.79";
masters { 213.96.79.79; };
};

      add entries for other zones below here

      zone "balearikus-party.org"
{ type master;
        file "/etc/bind/db.balearikus-
party.org";         allow-query { any; };
allow-transfer { slaves; };
      };

      zone "clan-bin.org"
{ type master;
        file "/etc/bind/db.clan-
bin.org";         allow-query { any;
};         allow-transfer { slaves;
};
      };
    };
```

The internal local network is *192.168.0.0*, hence an access list is used that includes any host that is on that network (network *192.168.0.0*, mask *255.255.255.0*, specified as the sum of some binaries, that is, *24*). This new situation leads us to specify a new definition of reverse zone, the one corresponding to the local network *0.168.192.in-addr.arpa*, which is shown below:

```
    ;
    ; BIND reverse data file for zone 192.168.0
    ;
    $TTL 604800
    @ IN SOA linuxsilo.net. hostmaster.linuxsilo.net. (
      2001081501   ; Serial
           10800   ; Refresh (3 hours)
            7200   ; Retry (2 hours)
         1296000   ; Expire (15 days)
          172800 ) ; Negative Cache TTL (2 days)

    @ IN NS
ns1.linuxsilo.net.     1 IN
PTR ns1.linuxsilo.net.
```

Note that the reverse zones, both those provided with the default installation for basic operation and those defined by the administrator, are now appropriately split between the two views. Instead, direct zones are duplicated, an occurrence for each view. Of course, the targeted zone files contain different registers, consistent with the view. Then the records of the files of internal direct zones are provided (the external ones remain the same, so those exposed earlier in this article are valid).

```
    ;
    ; BIND data file for zone balearikus-party.org,
internal view ;
    $TTL 604800
    @ IN SOA balearikus-party.org. hostmaster.linuxsilo.net. (
      2002051001   ; Serial yyyy/mm/dd/id
           10800   ; Refresh (3 hours)
            7200   ; Retry (2 hours)
         1296000   ; Expire (15 days)
          172800 ) ; Negative Cache TTL(2 days)

    balearikus-party.org. IN NS ns1.linuxsilo.net.
balearikus-party.org. IN MX 5 ns1.linuxsilo.net.

    localhost IN A 127.0.0.1 balearikus-party.org.
IN A 192.168.0.1

    www IN A 192.168.0.1
pop3 IN A 192.168.0.1
pop3 IN A 192.168.0.1
smtp IN A 192.168.0.1
ftp IN A 192.168.0.1

    ;
    ; BIND data file for zone linuxsilo.net,
internal view ;
    $TTL 604800
    @ IN SOA linuxsilo.net. hostmaster.linuxsilo.net. (
      2002051001   ; Serial yyyy/mm/dd/id
           10800   ; Refresh (3 hours)
            7200   ; Retry (2 hours)
         1296000   ; Expire (15 days)
          172800 ) ; Negative Cache TTL(2 days)

    NS
ns1.linuxsilo.net.
MX 5 ns1.linuxsilo.net.

    localhost A 127.0.0.1
linuxsilo.net. A 192.168.0.1

    ns1 A 192.168.0.1
ns2 A 213.96.79.79 www A
192.168.0.1 pop3 A
192.168.0.1 smtp A
192.168.0.1 ftp A
192.168.0.1 ts A
213.96.79.79 akane A
192.168.0.1

    ranma A 192.168.0.6
genma A 192.168.0.5
kasumi A 192.168.0.4
nabiki A 213.96.79.79
primetime A 192.168.0.3
```

The RNDC tool

The *rndc* command is a useful tool for manipulating *named*. The following table shows some of the options you accept. The parameters that cause the creation of files will do so in the directory specified as *the home* of *named* in the */etc/bind/named.conf* (*directory* clause, whose default value is */var/cache/bind* in Debian).

Command	Function
Help	List the available RNDC options
status	Displays the current status of the *Named* In execution
Trace	Increases the debug level by 1
Notrace	Turn off debugging
dumpdb	Dumps the DNS database to *named_dump.db*
Stats	Dump stats to *named.stats*
Reload	Reloading *named.conf* and zone files

Reload *zone*	Recharge only the specified zone
Restart	Restart *Named*, emptying the cache
querylog	Enable tracking of incoming queries

Rndc uses UDP port 953 for remote control. If the guidelines shown in this article are followed, it is not necessary for that port to be accessible from the outside - configure it on the router - because the control will always be done from the local host and zone transfers are made by TCP port 53

Log customization

Logs in Bind are configured with the `logging statement` in the *named.conf*. First, channels are defined, which are the possible destinations of the messages. Various categories of messages are then told to go to a particular channel.

Term	Meaning
channel	A place where messages can go: syslog, a file, or */dev/null*
	A class of messages that Bind can generate; for example, messages about dynamic updates or messages about responses to category queries
module	The name of the source module that generates a message place The name of a syslog
	place. DNS does not have its own destination, so the standard ones will have to be chosen. importance
	How "bad" an error message is; What Syslog refers to as priority

When a message is generated, it is assigned a category, a module, and an importance at its point of origin. It is then distributed to all channels associated with that category and module. Each channel has an importance filter that defines what level of importance a message must have to get through. The channels leading to the syslog are also filtered according to the */etc/syslog.conf rules*.

This is the skeleton of a `logging statement`:

```
    logging {
definición_de_canal;
definición_de_canal;
...
      category
nombre_categoría {
nombre_canal;
nombre_canal;          ...
      };
    };
```

A *definición_de_canal* is slightly different depending on whether the channel is a file or a syslog channel. You must choose `file` or `syslog` for each channel; a channel cannot be both at the same time.

```
    channel "nombre_del_canal" {
      file path [versions numb | unlimited] [size
sizespec];       syslog facility;       severity
importance;       print-category yes | No;
print-severity yes | No;       print-time yes | No;
    };
```

In a file, *numers* specifies how many copy versions of a file to save, and `sizespec` tells how large those files can become (e.g., `2048`, `100k`, `20m`, `15g`, `unlimited`, `default`).

In the case of syslog, *facility* specifies which place name to use when saving the message. It can be any of the standard. In practice, only *daemon* and *local0* to *local7* are reasonable choices.

All other statements in a channel definition are optional. *Importance* can take the values (in descending order) `critical`, `error`, `warning`, `notice`, `info` or `debug` (with an optional numerical level, for example `severity debug 3`). The `dynamic` value is also valid and represents the current debug level of the server.

The various `print` options add or remove message prefixes. The syslog includes the date and time and the source host in each saved message, but not the importance or category. In Bind 9, the source file (module) that generated the message is also available as a `print` option. It makes sense then to activate `print-time` only for file channels, since the syslog logs set the date and time themselves.

The following are the four default predefined channels, which should be sufficient for most cases:

Channel name	What it does
default_syslog	Command importance `info` to the syslog with the default_debug
daemon destination	Save to file *named.run*, importance placed on `Dynamic`
default_stderr	Sends messages to the standard error output of *Named*importance
`info`	

Null	All messages are discarded

The default `logging` setting for Bind 9 is:

```
    logging {
category default {
default_syslog;
default_debug;
      };
    };
```

You should take a look at the log files when you make big changes to Bind, and perhaps increase the level of debugging. Then, reconfigure it to preserve only important messages once *named* is stable. Some of the most common log messages are listed below:

- *Lame server*. If you receive this message about one of your zones, you have set something wrong. The message is relatively unimportant if it is about some area on the Internet, because it means that it is someone else's problem.
- *Bad referral*. This message indicates a lack of coordination in communication between the name servers in a zone.
- *Not authoritative for*. A slave server is not able to obtain representative information about an area. Maybe you're pointing at the wrong teacher, or maybe the teacher has had some trouble loading that area.
- *Rejected zone*. *named* rejected that area because it contained errors.
- *No NS RRs found*. A zone file does not have NS records after SOA registration. It could be that they are not or that they do not start with a tab or a blank. In the latter case, the logs are not interpreted correctly.
- *No default TTL set.*. The best way to set the default TTL is with a `$TTL clause` at the beginning of the zone file. This error message indicates that the `$TTL` is not present.
- *No root name server for class*. Your server is having trouble finding the root servers. Check your */etc/bind/db.root* file and your server's Internet connection.
- *Address already in use*. The port on which *named* wants to run is already being used by another process, probably another copy of *named*. If you don't see another copy of *named* in memory, it might have crashed, leaving the *rndc* control socket open.

Taxonomy of a name server

Server Type		Description
English	**Spanish**	
authoritative	authoritative	An official representative of an area.
Master	teacher	The main repository of data for a zone; Reads file data from disk.
Slave	slave	Obtains the master's data.
Stub	N/A	Similar to a slave, but only copies the data from the name server (not the data from the computer).
Distribution	distribution	A server that is only visible from within a domain (a "hidden server").
nonauthoritative(b)	unauthorized	Answers a query from its cache; he does not know if the data is still valid.
Caching	reservation	Save data from previous queries; it usually does not have local areas.
Forwarder	Redirect	Makes inquiries on behalf of many clients; It maintains a large cache.
Recursive	recursive	It queries on your behalf until you return a response or error.
nonrecursive	Non-recursive	It happens to another server if it is not able to respond to the query.

a. A distribution server can be visible to anyone who knows its IP address.
b. Strictly speaking, "unauthorized" is an attribute of the response to a DNS query, not a server.

Types of Statements Used in *the named.conf*

Sentence	Description
Include	Interpolates a file (e.g., trusted keys accessible only by **named**).
Options	Sets global name server settings and defaults.
Server	Specifies preserver options.
Key	Defines authentication information.
acl	Define access control lists.
zone	Defines a resource registration zone.
trusted-keys	Use pre-configured keys.
controls	Defines channels used to control the name server with **rndc**.
logging	Specify categories of logs and their destinations.
View	Defines a view of a namespace.

Example of log customization

```
    We define three log channels (important syslog
messages, medium debug, and zone load messages) //
and then assign categories to each.
    logging {
      channel
syslog_errors { syslog
local1;         severity
error;
      };
      channel moderate_debug {
        severity debug 3;    Level 3 debugging
file "debug.log";    to the debug.log file
        print-time yes;      current date to log print-category
yes entries;  print the name of the print-severity yes category;
Print the severity level
      };
      channel
no_info_messages { syslog
local2;         severity
notice;
      };
category parser {
syslog_errors;
default_syslog;
      };
      category lame-servers { null; };  Do not save this type in
the category load { no_info_messages; } logs;       category
default { default_syslog;         moderate_debug;
      };
    };
```

Table of special characters used in resource records

Character	Meaning
;	Leave a comment
@	The current domain name
()	Allows a sentence to be split into more than one line
*	Wildcard (only in the name of the field).

Table of security mechanisms in *named.conf*

Feature	Judgments	What it specifies
allow-query	options, zone	Who can query the zone or server.
allow-transfer	options, zone	Who can request zone transfers.
allow-update	zone	Who can make dynamic updates.
Blackhole	Options	Which servers should be completely ignored.
Bogus	Server	Which servers should never be consulted.
acl	several	Access control lists.

Bind 9 Logging Category Table

Category	Includes
default	Categories without an explicit channel assignment.
general	Unclassified messages.
config	Analysis and processing of configuration files.
queries/client	A short log message for each query that the server receives.
DNSSEC	DNSSEC messages.
lame-servers	Servers that are supposed to serve a zone, but are not.
statistics	Aggregated nameserver statistics.

Panic	Fatal errors (duplicates in this category).
update	Messages about dynamic updates.
Nocache	Negative cache messages.
xfer-in	Zone transfers that the server is receiving.
xfer-out	Zone transfers that the server is sending.
db/database	Messages about database operations.
packet	Received and sent package dumps(b).
Notify	Messages about the "modified zone" notification protocol.
Name me	Messages such as "... points to a CNAME".
Security	Requests approved/denied.
you	Operating system issues.
insist	Internal consistency failure checks.
maintenance	Periodic maintenance events.
load	Zone loading messages.
response-checks	Comments on malformed or invalid response packets.
resolve	DNS translation, e.g. recursive lookups for clients.
Network	Network operations.

to. Either the parent zone or the daughter zone could be the culprit; it is impossible to determine without investigating it. b. It must be a simple channel.

Chroot

This section describes some extra security-related precautions that you can take when installing BIND. It explains how to configure BIND so that it resides in a *chroot cage*, which means that it cannot view or access files outside of its own small directory tree. It also explains how to configure it to run as a non-root user.

The idea behind a chroot is quite simple: limit the access that a malicious individual can gain by exploiting BIND vulnerabilities. For the same reason it is good to run it as a non-root user (on GNU/Debian Linux from version 9.2.4-5). When BIND (or any other process) is run in a chroot cage, the process is simply unable to see any other part of the file system that is outside the cage. For example, in this section we will configure BIND in chroot mode in the */var/lib/named directory*. So, for BIND, the contents of this directory will be the root */*. Nothing outside of this directory will be accessible to you. You have most likely encountered a chroot cage before, for example when accessing a public system via FTP.

This process should be considered a complement to the usual security precautions (running the latest version, using access control lists, etc.), and never as a way to replace them.

Create the user

As mentioned in the introduction, it's not a good idea to run BIND as root. Therefore, before starting, a separate user will be created for BIND. Note that an existing generic user of the nobody type should never be used for this purpose. This process is done automatically by the Debian package installation script, but the manual procedure to achieve it is summarized below.

You need to add a line similar to this in the */etc/passwd file*:

```
bind:x:103:103::/var/cache/bind:/bin/false
```

And one similar to the following in the */etc/group file*:

```
bind:x:103:
```

In this example we are not only going to run BIND as a non-root user, but we will also be running it in a chroot environment (the default installation in Debian only covers the first aspect at present).

These lines create a user and a group called *bind* for BIND. The reader must ensure that both the UID (**User Identifier**) and the GID (**Group Identifier**) are unique in their system (both 103 in this example). The console has been left at */bin/false* because this user will never need to log in.

Directory structure

Next, you need to create a directory structure for the chroot cage in which BIND will run. It can be done anywhere in the file system; those more paranoid will even want to put it in a separate volume. */var/lib/named* will be used. Start by creating the following directory structure:

```
/var/lib/
  +--- named
        +--- dev
        +--- etc
        +--- var
              +--- run
              +--- cache
                    +--- bind
        +--- usr
              +--- sbin
```

The *mkdir command* could be used to create the aforementioned directory structure. These are the commands to execute and their parameters:

```
    # mkdir -p /var/lib/named
    # cd /var/lib/named
    # mkdir -p dev etc/bind var/run var/cache/bind usr/sbin
```

Copying the necessary files

Assuming that you have already done a standard installation of BIND 9 and are using it, you will therefore have a *named.conf* file and several zone files. These files must be moved (or copied, for greater security) inside the chroot cage, so that BIND is able to find them. *named.conf* and zone files go inside */var/lib/named/etc/bind*. For example:

```
    # cp -p /etc/bind/* /var/lib/named/etc/bind/
    # cp -a /var/cache/bind/*
/var/lib/named/var/cache/bind/ # cp /usr/sbin/named
/var/lib/named/usr/sbin/
```

In addition, it will also be necessary to copy the libraries that the */usr/sbin/named* executable needs, which can be obtained by running the *ldd /usr/sbin/named command*. With the following commands we will create the necessary subdirectories and copy the libraries:

```
    # cd /var/lib/named
    # mkdir -p usr/lib lib
    # cp /usr/lib/liblwres.so.1 usr/lib/
    # cp /usr/lib/libdns.so.5 usr/lib/
    # cp /usr/lib/libcrypto.so.0.9.6 usr/lib/
    # cp /usr/lib/libisccfg.so.0 usr/lib/
    # cp /usr/lib/libisccc.so.0 usr/lib/
    # cp /usr/lib/libisc.so.4 usr/lib/
    # cp /lib/libnsl.so.1 lib/
    # cp /lib/libpthread.so.0 lib/
    # cp /lib/libc.so.6 lib/
    # cp /lib/libdl.so.2 lib/
    # cp /lib/ld-linux.so.2 lib/
```

BIND will need to write within the */var/lib/named/var/cache/bind* and */var/lib/named/var/run* subdirectories, in the first to save the zones of which it is acting as a slave server and in the second to save statistical information about its execution. Therefore, it is pertinent to execute the following two statements:

```
    # chown -R bind:bind /var/lib/named/var/cache/bind
    # chown -R bind:bind /var/lib/named/var/run
```

System files

Once BIND is running in the chroot cage, it will not be able to access files outside the cage in any way. However, you need access to some essential files. One of the files you need inside your cage is */dev/null*. Others are */dev/zero* and */dev/random*. The following commands can be used:

```
    # mknod /var/lib/named/dev/null c 1 3
    # mknod /var/lib/named/dev/random c 1 8
    # mknod /var/lib/named/dev/zero c 1 5
    # chmod 666 /var/lib/named/dev/{null,random,zero}
```

You will also need another file in the */etc* directory inside the cage. You need to copy */etc/localtime* in there so that BIND saves the logs with correct dates. The following command would resolve the issue:

```
    # cp /etc/localtime /var/lib/named/etc/
```

Logging

BIND typically saves logs via *syslogd*, the system daemon responsible for storing logs. However, this type of logging is done by sending log entries to a special socket, */dev/log*. Because it is outside the cage, BIND will not be able to use it. Fortunately, there is a solution to this problem.

All you have to do is add the *-a /var/lib/named/dev/log parameter* to the command line that launches the syslogd. In Debian, this script is located in */etc/init.d/sysklogd*. You should look for the following lines:

```
    # Options for start/restart the daemons
    # For remote UDP logging use SYSLOGD="-r"
    #
    SYSLOGD=""
```

And change the last of the lines to this:

```
    SYSLOGD="-a /var/lib/named/dev/log"
```

Once the daemon is restarted, you should see a file in */var/lib/named/dev* called *log*, like this:

```
    srw-rw-rw- 1 root root 0 Nov 28 14:22 log
```

Finally, it should be noted that, in the case of an update of the *sysklogd* package, the changes made in said script could be lost due to a potential overwriting of the existing file. Therefore, it is recommended to be careful when upgrading. **Tightening permits**

First of all, feel free to restrict access to the entire */var/lib/named* directory to only the bind user.

```
    # chown bind:bind
/var/lib/named # chmod 700
/var/lib/named
```

If you want to increase the restrictions even further, on Linux systems you can achieve the immutability of some of the files by using the *chattr tool on* ext2 *and* ext3 *filesystems*.

```
    # cd /var/lib/named
    # chattr +i etc etc / localtime var
```

See the *man page* for more information: *man chattr*. It would be interesting to be able to do the same thing on the *dev directory* but, unfortunately, that would make it impossible for the *syslogd* to create the *dev/log* socket. You can also choose to enable the immutability bit on other files within the cage, such as those in the primary zones, in case they are not changing.

Installation

If you want to install from the fonts, it is recommended to use the bind9 Debian font package, available via the *apt-get source bind9 command*. Then it would only be necessary to execute a *./configure* and a *make* or, even easier, *dpkg-buildpackage*, which will do the two previous steps and leave us ready a series of Debian packages that will save us from having to make an *install* by hand.

Assuming that you already have a BIND 9 installation on your system, you will only need to slightly modify the daemon boot script to use these parameters:

`-u bind`, which tells BIND the user with which it should run.
`-t /var/lib/named`, which tells BIND to chroot itself inside the cage that has been prepared for it.
`-c /etc/named.conf`, which tells BIND where to find your configuration file inside the cage.

In Debian, it's very easy to change the BIND startup script, which you'll find in */etc/init.d/bind9*, to accept these new options. Just look for these lines:

```
    # for a chrooted server: "-u nobody -t
/var/lib/named" OPTS=""
```

And change the last line to the following:

```
    OPTS="-u bind -t /var/lib/named -c etc/named.conf"
```

Finally, change the executable that is called from that script from */usr/sbin/named* to */var/lib/named/usr/sbin/named*, so that it is the executable inside the cage that the script calls and not the original.

If your version of the BIND9 Debian package is 9.2.4-5 or higher, BIND will be running with the bind user and the UID/GID 103, so you can skip the steps related to user and group creation and go straight to the cage. If your package version is lower (in GNU/Debian Woody it is 9.2.44), then you will need to follow all the steps.

Configuration changes

You will also need to change or add a few options to your *named.conf* in order to keep certain directories in order. In particular, you should add (or change, if you already have them) the following policies in the options section:

```
    directory "/etc/bind";     pid-
file "/var/run/named.pid";
statistics-file
"/var/run/named.stats";
```

Since this file is going to be read by the named daemon, all the paths are obviously going to be relative to the chroot cage. At the time of writing, BIND 9 does not support many of the statistics and dump files that previous versions supported. Presumably, later versions will; if you are running one of those versions, you may need to add some additional entries for BIND to write to the */var/run directory* as well.

Booting BIND

Now all that remains is to do the most elementary step: to start the BIND demon again. To do this, you need to run the following command on a GNU/Debian Linux distribution:

```
    # /etc/init.d/bind start
```

Online Resources

- The DNS Resources Directory
- DNS HowTo
- Securing DNS with Transaction Signatures
- All About DNS
- DNS How to

RFCs

RFCs that define the DNS system are available in www.rfc-editor.org. Initial and developing ideas first appear as drafts and are later formalized as RFCs. Listed below are a set related to Bind, including those that have assumed that Bind 9 has been rewritten from scratch:

The original and definitive standards:

- 1034 - Domain Names: Concepts and Facilities.
- 1035 - Domain Names: Implementation and Specification.

Proposed Standards:

- 1995 - Incremental Zone Transfers in DNS.
- 1996 - A Mechanism for Prompt Notification of Zone Changes.
- 2136 - Dynamic Updates in the Domain Name System. 2181 - Clarifications to the DNS Specification.

New Standards Tracking RFCs:

- 2535 - Domain Name System Security Extensions.
- 2671 - Extension Mechanisms for DNS (EDNS0).
- 2672 - Non-Terminal DNS Name Redirection (DNAME). 2673 - Binary Labels in the Domain Name System.

Miscellaneous RFCs:

- 1535 - A Security Problem... with Widely Deployed DNS Software.
- 1536 - Common DNS Implementation Errors and Suggested Fixes.
- 1982 - Serial Number Arithmetic
- 2536-2541 - Various RFCs on DNSSEC.

Types of resource records:

- 1183 - New DNS RR Definitions: AFSDB, RP, X25, ISDN, RT.
- 1706 - DNS NSAP Resource Records.
- 1876 - A Means for Expressing Location Information in DNS.
- 2052 - A DNS RR for Specifying the Location of Services (SRV).
- 2168 - Resolution of Uniform Resource Identifiers using DNS.
- 2230 - Key Exchange Delegation Record for the DNS.

DNS and Internet:

- 1101 - DNS Encoding of Network Names and Other Types.
- 1123 - Requirements for Internet Hosts: Application and Support.
- 1591 - Domain Name System Structure and Delegation. 2317 - Classless in-addr.arpa Delegation.

DNS Operations:

- 1537 - Common DNS Data File Configuration Errors.
- 1912 - Common DNS Operational and Configuration Errors.
- 2182 - Selection and Operation of Secondary DNS Servers. 2219 - Use of DNS Aliases for Network Services.

Other DNS-related RFCs:

- 1464 - Using DNS to Store Arbitrary String Attributes.
- 1713 - Tools for DNS debugging.
- 1794 - DNS Support for Load Balancing.
- 2240 - A Legal Basis for Domain Name Retrieval.
- 2345 - Domain Names and Company Name Retrieval.
- 2352 - A Convention for Using Legal Names as Domain Names.