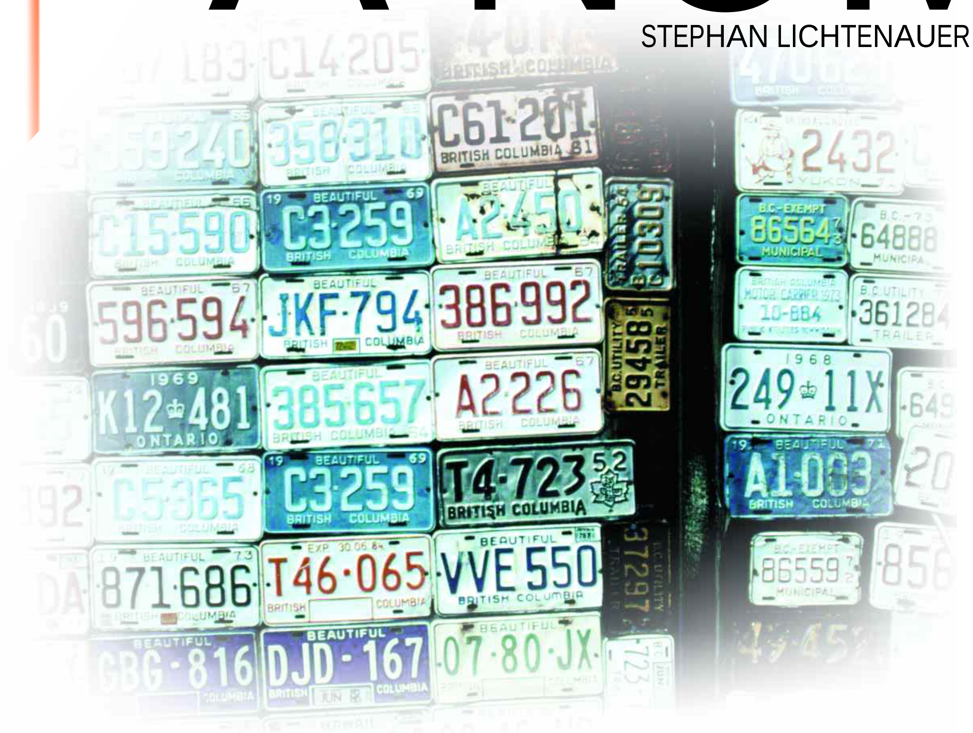
Configuration and operation of DNS servers

NOT JUST A NUMBER

STEPHAN LICHTENAUER



The nameserver BIND is practically standard under Unix and Linux. Unfortunately, it is very sparsely documented. The man pages for example are at best useful as a reference. And yet well-maintained name servers are essential for users of all Internet services in any organisation.

Name allocation and resolution in the Internet and other IP-based networks has a long history. Since 32-bit addresses, by which network nodes are actually addressed, are hard to remember, computers quickly began to be given names. At first people made do by setting up a file, HOSTS.TXT, which allocated a name to each IP address in the network.

This file, which is still in use today (and is called in Linux */etc/hosts*), contains an IP address, an allocated host name and optional alternative aliases by which this computer can also be accessed. In the early days of the ARPANET, with a few dozen computers, that was adequate. Even today in some simple intranets this is still a workable solution, but not in the modern hierarchical structure of the Internet (the successor to ARPANET). So pretty soon the search was on for a solution which, firstly, makes the most of the advantages of this hierarchy and secondly, makes it unnecessary to maintain separate, but nevertheless consistent, host files on each computer. This search resulted in several Requests For Comments (RFCs), including RFC 1035 (Domain Names – Implementation and Specification) and RFC 1034 (Domain Names – Concepts and Facilities).

The Domain Name System

To make it possible to manage the millions of computers of the Internet a hierarchical name structure was introduced. The root of this name is a period ".", followed by one of the global top level domains laid down by the IANA (Internet Assigned Numbers Authority), for example *com*, *edu*, *org*, *uk* or *is*. For each of these name domains, in turn, various organisations assign subordinate domains. Thus, for example, Nominet is responsible for all names in the *uk* domain.

If you register a name, you can yourself create a hierarchy with as many additional subdomains as you like, i.e. subordinate names. The ".", which specifies the root, is left out in everyday use, which means that for example *penguin.production.jaguar.com* uniquely identifies a computer called *penguin*, which is part of the subdomain *production*, which in turn are subordinate to *jaguar* and the top level domain *com*.

When TCP/IP has been installed on a computer (which is the case for all computers running Unix or Linux computers) then at least one name server must be specified. This name server will resolve the host names into IP addresses. Often, particularly in the case of dial-up connections using the PPP protocol, the name server is assigned dynamically. Either way, a name server must be known to the computer because only a name server can convert host names into IP addresses, and only IP addresses can be used for communication across the network.

If an application wants to resolve a host name to determine the associated address, the procedure that is followed is governed by the file */etc/host.conf*, in which the search sequence is defined. Normally, the file */etc/hosts* will be searched first. After that, if no matching name has been found there, the name server is contacted. The name server then either processes the enquiry itself, if its database holds the data for the name domain in question, or it passes it on to the next server in the hierarchy.

Let's look at an example. If the server responsible for *jaguar.com* receives an enquiry for *nathan.rover.com*, it will pass on the enquiry to the server for the entire top level domain *com*. On the other hand it could resolve *venus.production.jaguar.com* by itself. The *com* server knows the address of the *rover.com* name server and delivers this to the enquirer, which can then repeat the enquiry to it.

The BIND8 IP-Name server packet

The Internet Software Consortium (ISC) designed and implemented the domain name system which has been the standard used to date. This system is called BIND (Berkeley Internet Demon). Although Berkeley is actually using BSD as its operating system, BIND is now used on practically all important platforms and included in almost all Unix systems and Linux distributions. With the change from BIND4 to BIND8, the current version, a major alteration (resulting in a certain amount of simplification) of the configuration occurred.

In BIND slang a domain is referred to as a zone. The server responsible for the zone has the database containing the master data. Any available secondary servers, which intervene in the event of a failure or overload of the primary master server, have a copy of this data (the slave zone.) When there is any change in the configuration of the zone the slaves are automatically provided with the new domain data by the master.

The server consists of a daemon process called *named*, which is usually started or stopped on configuration in System V style by a boot script (usually */sbin/init.d/named*.) If the configuration is changed, the daemon process must be persuaded using *kill -HUP* to do a new read-in of the files. Any error messages and the results of communication with the other name servers, which must be informed of the changes, are found in the system log. So that these messages are not overlooked, it is recommended that this file is always displayed on a console using the command *tail -f /var/log/messages*.

Configuration of *named*

BIND8 has a central configuration file */etc/named.conf*, which, apart from general parameters, determines the zones that are controlled and their associated zone configuration files (Listing 2).

Slave zones

Operating a secondary name server – in other words, controlling a slave zone – is not difficult. Once the brief zone definition has been made in */etc/named.conf*, which defines the master server, and entering the name of a file in which BIND is to store the data, the job is done. All necessary database updates are fetched automatically by *named*, as long as the associated primary server is correctly configured.

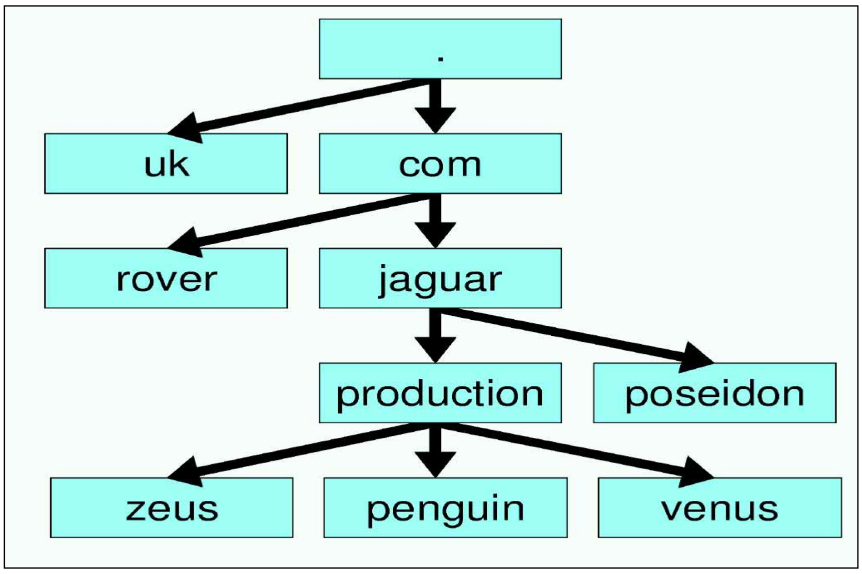


Fig. 1: The position of penguin.production.jaguar.com in the name hierarchy of the Internet

Zone files

For those domains for which your name server is configured as the primary server, however, more care is needed. In particular the divided, hierarchical architecture of the Internet domain concept does not permit trial and error methods of configuration. You must bear in mind that you have no access whatsoever to most servers which have zone data defined by you, whether it is a secondary server (for example your ISP's name server), or some other server which after a query has temporarily stored your data in its cache. You yourself can to a large extent define the lifetime of such invalid details through the time-to-live of the records defined by you (we will show you how later.)

Usually, you will define the root zone and at least two master zones. For each of the master zones there are two database files; one for resolution of names into addresses and one for the reverse procedure of resolving addresses into associated names (reverse lookup). Apart from this there is a file with the addresses of the root name servers which control the data on the top level domains and which your name server may need to contact as a last resort. This file should be checked at regular intervals to make sure it is up to date; it can be obtained from

Listing 1: HOSTS.TXT-Example file

```
192.168.1.1  poseidon.qthree.uk  mysql
192.168.1.2  venus.qthree.uk     ftp mail www
192.168.1.3  client1.qthree.uk
192.168.1.4  client2.qthree.uk
```

ftp://rs.internic.net/domain/named.root. Copy this file into the directory with your zone files (usually – and as stated in the sample configuration file – this is */var/named*) and give it the name which you have also specified in *named.conf* (here, *root.hint*).

As the starting point for your own zone files, if your */etc/hosts* file is large, you can use the tool *h2n*. This tool converts */etc/hosts* files into BIND zone databases. You can execute this program at regular intervals if your */etc/hosts* file always contains the latest data. Usually, however, zone databanks are managed by hand.

Listing 2: Example of a /etc/named.conf file.

```
/* Sample configuration for BIND 8.1 or new
 * install as /etc/named.conf
 *
 * Author: Stephan Lichtenauer
 * Note: All IP addresses/host names have been found
 */

#
# General server parameters
#
options {
       # Directory in which the zone databanks are stored
       directory "/var/named";
       # by default in case of errors in the master zone files
# the server will be stopped
       check-names master warn;

       pid-file "/var/named/slave/named.pid";

       datasize default;
       stacksize default;
       coresize default;
       files unlimited;
       recursion yes;
       multiple-cnames no;

       # by default at Port 53 there is a listen-out for all
available
       # interfaces, following commands could
       # specify this more precisely:
       #listen-on { 5.6.7.8; };
       #listen-on port 1234 { !1.2.3.4; 1.2/16; };
       query-source port 53;
};

#
# Logging options for various problems:
#
logging {
       category lame-servers { null; };
       category cname { null; };
};

#
# Pre-defined "Access Control Lists" (ACL):
# "any"       Lets any hosts in
# "none"      Prohibits all hosts
# "localhost"   Allows connections from this computer
# "localnets"   Allows connections from LANs (192.168.0.0/16)
#
# Define own ACL:
acl secondaries { 193.158.2.17; 152.133.12.18; };

#
# With the »server« instruction, other servers can be assigned
# certain properties.
#
# A server marked as »bogus« is never queried
server 193.158.24.28 { bogus yes; }
# if the other server has also installed at least BIND 8.1,
# zones can be transferred more compactly.
server 193.158.24.29 { transfer-format many-answers; }
```

```
#
# Defining the root zone
#
zone "." IN {
       type hint;
       file "root.hint";
};

#
# Defining the »localhost« zone
#
zone »localhost« IN {
       type master;
       file "localhost.zone";
       check-names fail;      // errors here would be fatal
       allow-update { none; }; // of local interest only
};

#
# Defining of reverse lookup for local host (addresses into names)
#
zone »0.0.127.in-addr.arpa« IN {
       type master;
       file "0.0.127.zone";
       check-names fail;
       allow-update { none; };
};

#
# Defining reverse lookup for an address zone
#
zone "36.158.193.in-addr.arpa" IN {
       type master;
       file "36.158.193.zone";
       check-names fail;
       allow-update { none; };
       allow-query { any; };
       allow-transfer { secondaries; };
       notify yes;
};

#
# A master zone
#
zone "jaguar.com" IN {
       type master;
       file "jaguar.com.zone";
       # Restrict zone transfers, to make work harder for
       # spies
       allow-transfer { secondaries; };
       allow-update { none; };
       allow-query { any; };
       notify yes;
};

#
# A slave zone
#
zone "rover.com" IN {
       type slave;
       file "slave/db.rover.com";
       masters { 194.238.99.128; };
};
```

As an example, let’s set up the file for the domain jaguar.com. According to our details in */etc/named.conf* this must be stored under */var/named/jaguar.com.zone* (Listing 3).

The ”SOA” record represents the start of the database file. *jaguar.com.* defines the described domain. Take note at this point of the dot on the end, which stands for the root name domain. You must always write this dot afetr all fully qualified names, otherwise *named* assumes the name has yet to be completed and appends the current domain.

poseidon.jaguar.com. (again with a dot at the end) stands for the current computer, on which *named* is running. *root.poseidon* gives the email address of the DNS administrator, with the dot standing for the otherwise usual ”@”. Since this time the name does not end with a dot, BIND completes the entry, making it *root.poseidon.jaguar.com.*, which represents the mailing address ”root@poseidon.jaguar.com”.

So that the other name servers storing your data (either as secondaries or in their cache) can check that they are up to date, you must specify a serial number for the data record, which you increment with each amendment. The concrete format is up to you; often the current date is used (as in this case 7 Jan 2000 is represented as 20000107.)

The refresh value states in seconds how often the secondary name servers should ask for updates (in this instance, ten hours). If the primary server should fail to answer this request in *retry* (in this case: 1800) seconds a new attempt will start. If within the period defined by *expire* no response is received from the primaries, the secondary server stops answering requests for this domain on the basis that no answer is still better than a wrong one.

TTL (time to live) is sent with all answers and shows how long the data record will remain valid and can remain in the cache. Choose this value with care, as with large values changes (and corrections for typing errors) take a very long time to spread through the network.

The following data records are each named according to the third column of the zone file (Listing 3). The two lines following the SOA record list the name servers (NS) for the domain. The first is the computer on which the master zone is located, then follow all the secondaries (just one in this case.)

The file then continues with the MX records. These state the addresses of the MaileXchanges, in other words, the mail servers. The number before the address is the priority value, representing a sort of inverse priority of this server. An SMTP server which wants to send an email first tries to connect to the server with the lowest priority value. Only if this fails will it look further down the list according to the nearest priorities.

A-records define the mapping of the host names onto IP addresses. Thus *poseidon* for example is completed, making it into *poseidon.jaguar.com*. If the request matches this name, 193.158.36.58 is returned as the associated address.

Listing 3: The zone file */var/named/jaguar.com.zone*

```
jaguar.com.       IN SOA poseidon.jaguar.com. root.poseidon
                                ( 20000107         ; serial
                                  36000            ; refresh
                                  1800             ; retry
                                  3600000          ; expire
                                  86400 )          ; time to live
jaguar.com.       IN     NS     poseidon.jaguar.com.
                  IN     NS     pns.bt.uk.

jaguar.com.       IN     MX 1    193.158.36.59
                  IN     MX 2    193.158.36.60

localhost         IN     A      127.0.0.1
poseidon          IN     A      193.158.36.58
phoenix           IN     A      193.158.36.59
venus             IN     A      193.158.36.60

ftp               IN     CNAME   phoenix.jaguar.com.
www               IN     CNAME   poseidon.jaguar.com.
ns                IN     CNAME   poseidon.jaguar.com.
news              IN     CNAME   venus.jaguar.com.
irc               IN     CNAME   venus.jaguar.com.

jaguar.com.         IN SOA poseidon.jaguar.com. root.poseidon
                                ( 20000107         ; serial
                                  36000            ; refresh
                                  1800             ; retry
                                  3600000          ; expire
                                  86400 )          ; time to live
```

CNAME data records make aliases available. ”news” – since, without a dot at the end after completion it becomes *news.jaguar.com* – is translated into venus.jaguar.com and the A-record associated with this host name is searched for and evaluated.

The zone for the *localhost* (Listing 4), which has to be included in every configuration, corresponds to the same syntax as the file for *jaguar.com*, except that the scope is considerably more manageable. However, a few small abbreviations are used: With *$ORIGIN*, *localhost*. is named as a macro for the current domain, to which the @ symbols then refer.

Reverse Look-ups

Some programs, such as for example *telnet*, try to find out the host names associated with IP addresses. These reverse lookups are resolved by BIND using *in-addr.arpa* zone files (Listing 5). In our file */etc/named.conf* we have defined a zone *36.158.193.in-addr.arpa IN …* , containing the addresses 193.158.36.0/24 (thus as a maximum in the domains 193.158.36.0 to 193.158.36.255). For historical reasons, IP-addresses for reverse-lookups are also written backwards (so no printing errors…) and must end in *in-addr.arpa* (and in the zone file, of course, with *in-addr.arpa.*).

The SOA-Record has the usual format (where one can also see the possible, self-explanatory and very practical abbreviations for units of time), only here the reverse lookup is defined. For this reason the zone name is *36.158.193.in-addr.arpa*. With NS, again, the primary and secondary name servers

Listing 4: The zone file /var/named/localhost.zone

```
# /var/named/localhost.zone contains the allocation
# of the loopback names and addresses

$ORIGIN localhost.
                                    42     ; serial
                                    3H     ; refresh
                                    15M    ; retry
                                    1W     ; expiry
                                    1D )   ; minimum

                      1D IN NS       @
                      1D IN A        127.0.0.1
```

Listing 5: The zone file */var/named/36.158.193.zone*

```
# /var/named/36.158.193.zone contains the allocation
# of host names to IP-addresses

36.158.193.in-addr.arpa.      IN SOA  poseidon.jaguar.com. root.poseidon
(
                                     20000107      ; serial
                                     3H            ; refresh
                                     15M           ; retry
                                     1W            ; expiry
                                     1D )          ; TTL

36.158.193.in-addr.arpa.          IN NS   poseidon.jaguar.com.
                                  IN NS   pns.bt.uk.

58.36.158.193.in-addr.arpa.          IN PTR  poseidon.jaguar.com.
59.36.158.193.in-addr.arpa.          IN PTR  phoenix.jaguar.com.
60.36.158.193.in-addr.arpa.          IN PTR  venus.jaguar.com.
```

Listing 6: The zone file */var/named/0.0.127.zone*

```
# /var/named/0.0.127.zone contains the allocation
# of local host to the address 127.0.0.1

0.0.127.in-addr.arpa.         IN SOA  poseidon.jaguar.com. root.poseidon
(
                                     43            ; serial
                                     3H            ; refresh
                                     15M           ; retry
                                     1W            ; expiry
                                     1D )          ; minimum

                      IN NS   poseidon.jaguar.com.
1                     IN PTR  localhost.
```

are defined, then come the PTR data records. These are the counterpart to the A-Records of forward resolution and allocate the host names to the IP addresses. Great care must be taken here to ensure consistency between the A- and the associated PTR-records. Together with the reverse-mapping file for the 127.0.0.0-address zone, this makes configuration complete. Since the 1 in the last line is not fully qualified and does not end with a dot, it is automatically completed to make *1.0.0.127.in-addr.arpa*.

Troubleshooting

As already mentioned, the fact that data is distributed to all possible configuration files and in the next step to all possible computers does not make it easy to find and to correct any errors. The most frequent error – apart from not rebooting the *named*-daemon – is forgetting, after making modifications, to increment the serial numbers of the zones, so that the connected computers do not notice that something has changed. You must also bear in mind that due to caching it may take some time before your amendments spread through the network (so at this point think of a reasonable TTL value.) In the event of problems with other zones the best thing to do is use whois or finger to find the contact information on the administrator responsible and speak to them.

You will find many errors as soon as you look at the system log (*/var/log/messages*) after a new read-in of the configuration. Syntax errors are also normal, if named quits in such a situation (this should not be the case if you have specified in */etc/named.conf check-names master warn*). Check whether your fully qualified names in the zone files end in a dot (thus for example *poseidon.qthree.uk*. If you write *poseidon.qthree.uk*, this will be completed as *poseidon.qthree.uk.qthree.uk.*, which is probably not what you want).

If applications such as *telnet*, which perform reverse look-ups, run very slowly, reverse look-up is probably not correctly configured. Test this with the tool *nslookup*, found in both Unix and Linux, and which acts as an all-purpose tool in the toolbox of the BIND administrator. In the following example the allocation of addresses to names does function, but the reverse is not true (193.158.24.68 is the address of the name server tested):

```
root@qthree.uk ~ # nslookup poseidon.ieee.↩
com 193.158.24.68
Server:  ieee.com
Address:  193.158.24.68

Name:    poseidon.ieee.com
Address:  193.158.24.69

root@qthree.uk ~ # nslookup 193.158.24.69
Server:  ns.phade.com
Address:  195.35.22.1

** ns.phade.com can't find 193.158.24.69: No↩
n-existent host/domain
```

nslookup will be able to help you in most cases. There is also *dnswalk*, which searches configurations for common errors such as inconsistent A- and PTR-data records. Don't forget to notify changes to the IP-address of your name server to the competent authority (Nominet, INTERNIC etc.).

Lame or *Missing Delegations* are also very common: In the first case a name server which is higher up in the hierarchy, when queried, delivers the address of the server which is supposedly responsible, but which is in fact completely ignorant of this good fotune. In the latter, reverse case, the server simply does not bring back the address of the one responsible. In order to avoid this it is necessary to have good co-operation with your ISP. And don't forget to check from time to time that your root file is up to date (in the example in this article this is *root.hint*). You can, of course, automate this with cron (but then make sure that whatever happens, you don't mail the log outputs from *named*). ■