DNS

DAW2

Index

Introduction

The DNS (Domain Name System) domain name service is basically responsible for the translation of canonical system names to their IP addresses and also for what is known as reverse resolution, which consists of starting from the IP to obtain the names associated with it since a system can have more than one name.

Let's briefly review the history of this basic service for the operation of today's networks and the Internet.

Initially, in the Internet's precursor network, ARPAnet, the translation between system names and their IPs was carried out by means of a text file called HOSTS.TXT which was maintained centrally and then the network systems obtained a copy on which to make their queries. This mechanism worked well as long as the number of systems connected to the network was maintained in dozens of systems, but as the number of new systems increased, keeping the file HOSTS.TXT updated became more difficult, to the point of not being operational. Thus, in 1984 the first version of DNS was published, which has been adopted as the service for name resolution in TCP/IP-based networks. In any case, in implementations of *NIX type systems we still have a file similar to HOSTS.TXT, the *\/etc/hosts* where we can establish the correspondences between names and IPs. Copying this file into the different systems of our network would work, the downside is that every time we have to make a name-IP change it must be done in all the *host files* of all the systems on the network.

The DNS system has a hierarchical structure in the form of a tree, similar to the file structure of UNIX-type systems, so that the DNS servers know the information of zones of that tree, when they are queried, if they have the answer they answer the client and if not then they forward the query to another DNS server of higher rank in the hierarchy. When you reach those who know the answer, it travels the opposite way to the customer.

Domain Name System - Wikipedia

Understanding and How DNS Works

The configuration and operation of the DNS service will be taken from the The Domain Name System document, available in Aules, the description made in this article is quite clear, although in some of the examples the syntax may vary with respect to the current version of DNS/BIND.

For the test configurations, the IPs to be used will be indicated. And also, when necessary, examples of files of configurations tested in the version of the service used will be provided.

Service Installation:

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

To begin with, we will follow the analysis and, if necessary, modify the configuration files that we find in the */etc/bind directory*:

`named.conf.options` the value of the IP `forwarders` to `194.179.1.100` that corresponds to a DNS server of Movistar Telefónica, we will also comment on the references to IPv6 *listen-on-v6 { any };* and the use of *DNSSEC-Validation Auto;* .

We will not modify the `named.conf` file, it incorporates the separate configurations of different zones.

The `named.conf.local file` will contain the configuration of the zones that we are going to create in our network and that we are going to manage:

This first statement limits the use of the rndc tool to the local machine that can be used to manage the service remotely.

```
controls { inet 127.0.0.1 port 953 allow { any; } keys { "rndc-key"; }; };
```

Next, create an access control list (ACL) where you specify systems that act as secondary DNS servers, the use of these lists would be more interesting if instead of a single system we had several.

```
ACL Slaves { 192.168.1.113; };
```

Now we define a DNS zone of our network, in which we are administrators (*master*), we indicate the file that will store the database that relates names - IP, also with *allowquery* we specify who can consult this DNS zone, with *any* we establish that anyone. The *allow-transfer* statement indicates that only the specified systems are allowed to obtain a copy of the database in this zone to act as secondary DNS servers.

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

The following zone expresses the zone for the inverse resolution, the applicable specifications are the same.

```
zone "18.168.192.in-addr.arpa"
{ type master; file
"/etc/bind/db.192.168.18";
allow-query { any; }; allow-
transfer { slaves; }; };
```

Next, it remains to give the values to the files of the databases of each zone, taking as examples: db.local, db.depinfo.iesjc and db.192.168.18 (bind.zip). If our network has a mask, for example 16-bit, in reverse resolution you can take the content of bind_res_inversa_mascara_16bits.zip as an example.

In each system that must take our system as a DNS server we must make its configuration; we can do it in three ways:

- If the system has a static IP, we must put the values in the yaml format file used by *netplan*. For example:

  01-network-manager-all.yaml:

```
Network:
Version: 2
renderer: NetworkManager ethernets: enp0XX:
dhcp4: no (or the line can be deleted)
        addresses: (there are two syntax, I guess equivalent)
            - 192.168.xxx.xxx/24 or [192.168.xxx.xxx/24]
        Gateway4: 192.168.xxx.yyy
nameservers: search:
[midominio.org]
         addresses: [xxx.xxx.xxx.xxx, yyy.yyy.yyy.yyyy]
```

- If you are a DHCP customer, it should be configured using the values you receive from this service.

- We can also set the values using the commands:

```
#systemd-resolve --set-dns=xxx.xxx.xxx.xxx –set-domain=mydomain.org -
inferface=enp0sX
```

To check DNS settings, use the command: `$ resolvectl status`

(For this DNS service to query external servers, the IPs of the virtual machines must be on the same network segment as the router or computer egressing to the Internet.)

To make DNS queries we can use the host command, for example:

```
$ host debian
$ host 192.168.18.xxx
```

Description Zone Databases

SOA " (start of Authority) indicates that it is an authority zone, and the records below have authority information. Normally, in the *domain part* there is the @ indicating the name of the domain. Structure:

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

TTL » Time to Live, Validity Time of DNS Record Information expressed in Second.
After this time, the customer must consult this information again.
A " (IPv4 address) associates a name with an IP address.
AAAA » (IPv6 Address)
CNAME » (Canonical Name) associates an alias with the official name of a computer, which has previously been declared with a type A registry. Its use is not recommended, as it can cause problems.
name_alias IN CNAME defined_name.domains.
NS » (Name Server) to point to a DNS server, usually a higher level if we are in a subordinate zone.
PTR » *(pointer)* translation from address to name, reverse resolution.

MX » (Mail eXchanger) controls mail routing.

LOC » (Localització) geographical location.

TXT » (Text) comments to present information.

RP » (Responsible Person) Specifies the contact person for each computer.

SOA:

origin » absolute name of a principal server of this domain. Absolute names end with a period ".", e.g. debian.depinfo.iesjc.

Contact » e-mail from the person responsible for the DNS service. The user email.subdomain is entered. ... . domain. (also ends in dot), e.g. root.depinfo.iesjc. .

serial » is a number that the administrator increments each time he modifies the file. And its purpose is to indicate to the secondary servers that the information has been modified, so, from time to time they ask the primary server for the SOA records and see if the serial number has changed, if so, they take the complete file to have the updated information.

refresh » indicates how often the secondary servers must request SOA records from the primary.

retry » indicates the time you must wait before accessing the main server again if the previous connection failed.

Expire » If you have not been able to connect to the primary server, after the time indicated here all the information in this area will be discarded. It is recommended to put 42 days (in seconds). *minim » is* the time limit if it has not been defined in the registers with area information.

Commands check DNS performance

To validate the files that contain the zone configurations we will use:

```
named-checkconf -p file_with_zones
 #named-checkconf -p named.conf.local
```

if there are no errors, it will show the defined areas and if it finds errors, it will tell us which one it is and on which line.

To validate the files with the zone databases we will use: `named-checkzone nom_zona fich_bd_esa_zona` -->

```
# named-checkzone depinfo.iesjc db.depinfo.iesjc
```

if there are no errors indicate the area to load and OK, but if there are errors indicate them.

To consult the service we have several commands: `host name or IP` → `$ host ord1.depinfo.iesjc`

```
$ host 192.168.18.x
```

it will answer us, if we put the name of the system with its IP and if we put an IP it returns the name of the system.

```
nslookup [name]-->
$nslookup
$nslookup www.google.es
```

allows you to query different DNS servers both interactively and as a single query. If we do not indicate the name of a system, we enter the interactive mode so that we can make successive queries on the indicator. And if we accompany the command with the name of a system to be consulted, it will respond only to this one.

dig is a DNS query command that provides more complete information than the previous ones as an answer. The basic syntax of *dig*:

`dig` --> Query the root of the DNS service and display the information for the server configured in /etc/resolv.conf `dig nombre_host` --> Returns all information associated with this host available in the computed DNS service. `dig -x IP` --> Like the previous one, but for the reverse resolution.

Secondary DNS Server and Zone Transfer

In DNS services, the secondary or slave servers keep a copy of the database, or a part of it, of the DNS service so that if there is a failure on the main server, queries can be satisfied by the secondary servers. Given their mission as an alternative in the event of main server failures, it is obvious that secondary servers should share the minimum infrastructures with the main server, such as power supply, if both systems fail, we lose service, or the connection to a switch if both are connected to the same network component, if this fails we will lose connectivity with the service.

The transfer of information between the primary and secondary servers is carried out automatically, it is only necessary to configure the services in such a way that their IP parameters, zones to be transferred and the corresponding permissions or keys are complemented so that the transfers of zones are carried out securely.

For the example configurations we will follow the files contained in bind_*secondary*.zip.

For the secondary, the configuration files to modify in */etc/bind* are:

`named.conf.options` we modify `forwarders` as in the principal.

`named.conf.local` we modify according to the zones that must be transferred to our secondary server, basically change type that now serves `slave,` the name of the files on which the copy of the database is made is indicated, but without specifying the path that by default is */var/cache/bind* and with the *master directive* we indicate the IP of the main server from which we expect the copy of that zone, According to our example:

```
controls { inet 127.0.0.1 allow { 127.0.0.1; } keys {
"rndc-key"; }; };

zone "depinfo.iesjc" { type
slave; file
"sec.db.depinfo.iesjc";
allow-query { any; };
masters { 192.168.18.111; };
};

zone "18.168.192.in-addr.arpa"
{ type slave; file
"sec.db.192.168.18"; masters {
192.168.1.111; }; };
```

Remember that for our systems to know this secondary server must be configured as seen above.

Once the secondary server is configured, we restart the service and check that the files with the copies of the databases have been created in */var/cache/bind*. If so, to test the operation of the secondary server we only have to stop the service of the main system and check that the queries continue to work, if we also stop the secondary, we will check that the DNS no longer works.

DNS over different network segments

Now we are looking at a DNS service that works on two network segments, in reality, it is only an example that presents the problem when we have more than one segment.

We start from a domain (depinfo.iesjc) that is implanted on two network segments with 192.168.18.XXX and 192.168.**28.XXX**, the database for normal name-to-IP resolution can be left with a single zone as defined in the examples seen so far and in the database file (db.depinfo.iesjc) the IP-name records of the two segments will be added. For reverse resolution, we'll use different zones for each segment, so we'll define two zones: *18.168.192.in-addr.arpa* for computers in the first segment, and *28.168.192.in-addr.arpa* for computers in the second segment. Each zone will have its database file where it will store the necessary IP-name records in the reverse resolution of that zone. The configuration files can be found in bind_2_segment_network.zip.

Update DNS using DHCP

The steps to follow can be found on the website:

http://lani78.wordpress.com/2008/08/12/dhcp-server-update-dns-records/ PDF version of the article.

A simple example configuration for bind9 and dhcp3 in dhcp_update_dns.zip

Policies that allow this process in the DHCP service configuration in /etc/dhcp/dhcpd.conf we added:

ddns-update-style **interim**; # Enables DNS update **ignore client-updates;** # Prevents the client from putting its FDDN **ddns-domainname "depinfo.iesjc.";** # Indicates the domain name **ddns-rev-domainname "in-addr.arpa.";** # Reverse domain name

We add the zones that are going to be updated in DNS:

```
zone depinfo.iescj. { primary 127.0.0.1; } zone

150.168.192.in-addr.arpa. { primary 127.0.0.1; }
```

It should also be noted that DHCP clients must send their name to the server so that it can add it to the rental record and pass it to DNS. This name must be indicated in the DHCP client configuration file, for example in **/etc/dhcp/dhclient.conf** in the **send host-name option; Note**: The AppArmor security application may limit access to /etc/bind files. If this is the case, the database files must be created in /var/lib/bind. Or you can disable the AppArmor service to facilitate DHCP + DNS testing with the commands: # service apparmor stop

#systemctl disable apparmor