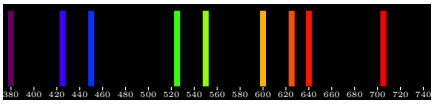
Domain name Server

@Franck Jeannot - 2016 - F158 - V1.0



1 Definition

The Internet Domain Name System (DNS) consists of the syntax to specify the names of entities in the Internet in a hierarchical manner, the rules used for delegating authority over names, and the system implementation that actually maps names to Internet addresses. DNS data is maintained in a group of distributed hierarchical databases.[1]

2 RFCs and literature

- Refer to a detailed list of RFCs at:
https://www.isc.org/community/rfcs/dns/
- Introduction to DNS:
https://tools.ietf.org/html/rfc1034
- Refer to http://lpic2.unix.nl/v3/

3 LPI : Linux Professional Institute

This document provides a DNS overview with a specific focus on the objectives of the LPI2 117-202 Certification. Refer to [2].

4 Acronyms and terminology

BIND : **B**erkeley **I**nternet **N**ame **D**omain
PTR : **P**oin**T**e**R** to another part of the domain name space
RRs : **R**esource **R**ecords
SOA (**S**tart **O**f **A**uthority) record is the first record in a zone file. The SOA record is used when using DNS to synchronize data between multiple computers.

[1] Introduction from ftp://ftp.isc.org/isc/bind9/9.10.4-P1/doc/arm/Bv9ARM.pdf
[2] https://www.lpi.org/study-resources/lpic-2-202-exam-objectives/

5 Basic DNS server configuration

5.1 LPIC-2 : Objective 207.1

Description: Candidates should be able to configure BIND to function as a caching-only DNS server. This objective includes the ability to manage a running server and configure logging.[3] [4]
Key Knowledge Areas:

- BIND 9.x configuration files, terms and utilities
- Defining the location of the BIND zone files in BIND configuration files
- Reloading modified configuration and zone files
- awareness of **dnsmasq**, **djbdns** and **PowerDNS** as alternate name servers.

The following is a partial list of the used files, terms and utilities:

- /etc/named.conf
- /var/named/
- /usr/sbin/rndc
- kill
- host
- dig

6 Authoritative and Non-authoritative DNS

An authoritative name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods, in contrast to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server only returns answers to queries about domain names that have been specifically configured by the administrator. An authoritative name server can either be a master server or a slave server. A master server is a server that stores the original (master) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy of the master records.[5] This sample configuration is for an authoritative-only server that is the master server for "example.com" and a slave for the subdomain "eng.example.com": com".

[3] http://lpic2.unix.nl/ch08.html
[4] https://www.lpi.org/study-resources/lpic-2-202-exam-objectives/
[5] https://social.technet.microsoft.com/Forums/windowsserver/en-US/89bd7e00-17b1-4fba-a2f2-1f6191d4a1c3/authoritative-dns-server-vs-nonauthoritative-dns-serve?forum=winservergen

```
options {
 // Working directory
 directory "/etc/namedb";
 // Do not allow access to cache
 allow-query-cache { none; }; //AUTHORITATIVE name server
 // This is the default
 allow-query { any; };
 // Do not provide recursive service
 recursion no;
};
// Provide a reverse mapping for the loopback
// address 127.0.0.1
 zone "0.0.127.in-addr.arpa" {
 type master;
 file "127.0.0.zone";
 notify no;
};
// We are the master server for example.com
zone "example.com" {
 type master;
 file "example.com.db";
 // IP addresses of slave servers allowed to
 // transfer example.com
 allow-transfer {
  192.168.4.14;
  192.168.5.53;
 };
};
// We are a slave server for eng.example.com
zone "eng.example.com" {
 type slave;
 file "eng.example.com.bk";
 // IP address of eng.example.com master server
 masters { 192.168.4.12; };
};
```

```
;cat 127.0.0.zone
$TTL 1W
@               IN SOA          localhost.   root.localhost.
    (
42              ; serial (d. adams)
2D              ; refresh
4H              ; retry
6W              ; expiry
1W )            ; minimum

IN NS           localhost.
```

```
1                IN PTR          localhost.
```

A non authoritative name server does not contain copies of any domains. Instead they have a cache file that is constructed from all the DNS lookups it has performed in the past for which it has gotten an authoritative response. When a non-authoritative server queries an authoritative server and receives an authoritative answer, it passes that answer along to the querier as an authoritative answer. Thus, non-authoritative servers can answer authoritatively for a given resolution request. However, non-authoritative servers are not authoritative for any domain they do not contain specific zone files for. Most often, a non-authoritative server answers with a previous lookup from its lookup cache. Any answer retrieved from the cache of any server is deemed non-authoritative because it did not come from an authoritative server. The following sample configuration is appropriate for a caching-only name server for use by clients internal to a corporation. All queries from outside clients are refused using the allow- query option. Alternatively, the same effect could be achieved using suitable firewall rules:[6]

```
// Two corporate subnets we wish to allow queries from.
acl corpnets { 192.168.4.0/24; 192.168.7.0/24; };
options {
 // Working directory
 directory "/etc/namedb";
 allow-query { corpnets; };
};
// Provide a reverse mapping for the loopback
// address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
 type master;
 file "localhost.rev";
 notify no;
};
zone    "unixmen.local"   {
type master;
file    "for.unixmen.local";
};
```

[6]Ref. p 9/280 of BIND 9.10.4-P1 Administrator Manual

6.1 DNS zones

Zones are the equivalent of domains. Zone configuration files consist of host-names and IP address information. Nameserver software responds to requests on port 53, and translates DNS host-names or domain-names to IP addresses. It can also translate IP addresses into DNS names, this is called a 'reverse DNS lookup' (rDNS). In order for rDNS to work, a so called pointer DNS record (PTR record) has to exist for the host being queried. We distinguish authoritative nameservers, recursive nameservers and so called resolvers. The authoritative nameserver for a zone is the nameserver which administrates the zone configuration. It is therefore sometimes also referred to as the zone master.
A recursive nameserver is one that resolves zones for which it is not authoritative for at other nameservers. The resolver is the part of the nameserver and DNS client software which performs the actual queries. In general, these are libraries as part of DNS software.

6.2 ISC BIND

6.2.1 BIND : DNS fundamentals

Clients look up information in the DNS by calling a **resolver** library, which sends queries to one or more **name servers** and interprets the responses. The BIND 9 software distribution contains a name server, **named**, and a resolver library, **liblwres**. The older libbind resolver library is also available from ISC as a separate download.

6.2.2 Install ISC BIND on OpenSuse from Sources - Recommended

Refer to the latest version for download in https://www.isc.org/downloads/.
This document is based on ***BIND BIND 9.10.4-P1 Administrator Reference Manual***. [7]
Refer to [8].

```
#Install openSSL dev files
#Suse : libopenssl-devel
#Red Hat, Fedora, CentOS : openssl-devel
#Debian, Ubuntu : libssl-dev
#Arch : openssl
zypper install libopenssl-devel
#Get the package and prepare it
cd /opt
mkdir bind
cd bind/
wget https://www.isc.org/downloads/file/bind-9-10-4-p1/?
    version=tar-gz
mv * bind-9-10-4-p1.tar.gz
```

[7] ftp://ftp.isc.org/isc/bind9/9.10.4-P1/doc/arm/Bv9ARM.pdf
[8] http://www.linuxfromscratch.org/blfs/view/cvs/server/bind.html

```
gunzip bind-9-10-4-p1.tar.gz
tar xvf bind-9-10-4-p1.tar
cd bind-9.10.4-P1
#configure
./configure --prefix=/usr   \
--sysconfdir=/etc       \
--localstatedir=/var    \
--mandir=/usr/share/man \
--enable-threads        \
--with-libtool          \
--disable-static        \
--with-randomdev=/dev/urandom

make
#Issue the following commands to run the complete suite of
    tests. First, as the root user, set up some test
    interfaces.
#If IPv6 is not enabled in the kernel, there will be several
     error messages: "RTNETLINK answers: Operation not
    permitted". These messages do not afffect the tests.

#Finally, install the package as the root user:
make install &&
install -v -m755 -d /usr/share/doc/bind-9.10.4-P1/{arm,misc}
     &&
install -v -m644    doc/arm/*.html \
/usr/share/doc/bind-9.10.4-P1/arm &&
install -v -m644    doc/misc/{dnssec,ipv6,migrat*,options,
    rfc-compliance,roadmap,sdb} \
/usr/share/doc/bind-9.10.4-P1/misc
#groups en user configuration
groupadd named
useradd -c "BIND Owner" -g named -s /bin/false -u 20 named
#Set up some files, directories and devices needed by BIND:
cd /srv/named
mkdir -p dev etc/namedb/{slave,pz} usr/lib/engines var/run/
    named
mknod /srv/named/dev/null c 1 3
mknod /srv/named/dev/urandom c 1 9
chmod 666 /srv/named/dev/{null,urandom}
#cp /usr/lib/engines/libgost.so usr/lib/engines &&
#[ $(uname -m) = x86_64 ] && ln -sv lib usr/lib64

#The rndc.conf file contains information for controlling
    named operations with the rndc utility. Generate a key
    for use in the named.conf and rdnc.conf with the rndc-
    confgen command:
rndc-confgen -r /dev/urandom -b 512 > /etc/rndc.conf &&
sed '/conf/d;/^#/!d;s:^# ::' /etc/rndc.conf > /srv/named/etc
    /named.conf
```

```
#Complete the named.conf file from which named will read the
    location of zone files, root name servers and secure DNS
    keys:
```

```
cat >> /srv/named/etc/named.conf << "EOF"
options {
 directory "/etc/namedb";
 pid-file "/var/run/named.pid";
 statistics-file "/var/run/named.stats";

};
zone "." {
 type hint;
 file "root.hints";
};
zone "0.0.127.in-addr.arpa" {
 type master;
 file "pz/127.0.0";
};

// Bind 9 now logs by default through syslog (except debug).
// These are the default logging rules.

logging {
 category default { default_syslog; default_debug; };
 category unmatched { null; };

 channel default_syslog {
  syslog daemon;                        // send to syslog's
      daemon
  // facility
  severity info;                        // only send priority
      info
  // and higher
 };

 channel default_debug {
  file "named.run";                     // write to named.run
      in
  // the working directory
  // Note: stderr is used instead
  // of "named.run"
  // if the server is started
  // with the '-f' option.
  severity dynamic;                     // log at the server's
  // current debug level
 };

 channel default_stderr {
```

```
  stderr;                                   // writes to stderr
  severity info;                            // only send priority
      info
  // and higher
 };

 channel null {
  null;                                     // toss anything sent
      to
  // this channel
 };
};
EOF
```

```
#Create a zone file with the following contents:
cat > /srv/named/etc/namedb/pz/127.0.0 << "EOF"
```

```
$TTL 3D
@       IN      SOA     ns.local.domain. hostmaster.local.
    domain. (
1         ; Serial
8H        ; Refresh
2H        ; Retry
4W        ; Expire
1D)       ; Minimum TTL
NS        ns.local.domain.
1               PTR     localhost.
EOF
```

```
#Create the root.hints file with the following commands:
cat > /srv/named/etc/namedb/root.hints << "EOF"
```

```
.                       6D  IN      NS      A.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      B.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      C.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      D.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      E.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      F.ROOT-SERVERS.
    NET.
```

```
.                       6D  IN      NS      G.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      H.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      I.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      J.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      K.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      L.ROOT-SERVERS.
    NET.
.                       6D  IN      NS      M.ROOT-SERVERS.
    NET.
A.ROOT-SERVERS.NET.     6D  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     6D  IN      A       192.228.79.201
C.ROOT-SERVERS.NET.     6D  IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     6D  IN      A       199.7.91.13
E.ROOT-SERVERS.NET.     6D  IN      A       192.203.230.10
F.ROOT-SERVERS.NET.     6D  IN      A       192.5.5.241
G.ROOT-SERVERS.NET.     6D  IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     6D  IN      A       128.63.2.53
I.ROOT-SERVERS.NET.     6D  IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     6D  IN      A       192.58.128.30
K.ROOT-SERVERS.NET.     6D  IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     6D  IN      A       199.7.83.42
M.ROOT-SERVERS.NET.     6D  IN      A       202.12.27.33
EOF
```

```
# The root.hints file is a list of root name servers. This
    file must be updated periodically with the dig utility. A
     current copy of root.hints can be obtained from ftp://rs
    .internic.net/domain/named.root. For details, consult the
     "BIND 9 Administrator Reference Manual", included in
    every source archive of BIND 9 distributed by ISC, in
    HTML and PDF formats, also available at BIND 9
    Administrator Reference Manual.


#Set permissions on the chroot jail with the following
    command:
chown -R named:named /srv/named
#To start the DNS server at boot, install the /etc/rc.d/init
    .d/bind init script included in the blfs-bootscripts
    -20160415 package.
cd /opt/bind/bind-9.10.4-P1/
wget http://anduin.linuxfromscratch.org/BLFS/blfs-
    bootscripts/blfs-bootscripts-20160415.tar.xz
tar xvf blfs-bootscripts-20160415.tar.xz
```

```
cd blfs-bootscripts-20160415


## Below steps to be adapted for each linux distribution...
make install-bind
#install -d -m 755 /etc/rc.d/rc{0,1,2,3,4,5,6,S}.d
#install -d -m 755 /etc/rc.d/init.d
#install -d -m 755 /etc/sysconfig
#install -m 754 blfs/init.d/bind       /etc/rc.d/init.d/
#ln -sf  ../init.d/bind /etc/rc.d/rc0.d/K49bind
#ln -sf  ../init.d/bind /etc/rc.d/rc1.d/K49bind
#ln -sf  ../init.d/bind /etc/rc.d/rc2.d/K49bind
#ln -sf  ../init.d/bind /etc/rc.d/rc3.d/S22bind
#ln -sf  ../init.d/bind /etc/rc.d/rc4.d/S22bind
#ln -sf  ../init.d/bind /etc/rc.d/rc5.d/S22bind
#ln -sf  ../init.d/bind /etc/rc.d/rc6.d/K49bind


#Check /etc/named.conf syntax
/usr/sbin/named-checkconf -z /etc/named.conf

#Create or modify resolv.conf to use the new name server
    with the following commands:
cp /etc/resolv.conf /etc/resolv.conf.bak &&
cat > /etc/resolv.conf << "EOF"
search example.com
nameserver 127.0.0.1
EOF
```

6.2.3 Install BIND on OpenSuse - from packages

Refer to:

- https://www.unixmen.com/setup-dns-server-opensuse-13-1/
- http://www.zytrax.com/books/dns/ch6/index.html#overview
- https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_dns_bind.html
- http://www.tldp.org/HOWTO/Chroot-BIND-HOWTO-2.html

Example of commands to install BIND ISC:

```
zypper in bind
#The following 2 NEW packages are going to be installed:
#bind bind-chrootenv
#warning: /etc/named.conf created as /etc/named.conf.rpmnew
#Updating /etc/sysconfig/named...
```

```
#wrote key file "/etc/rndc.key"

#Check the named group and users created
grep named /etc/group
#named:x:1000:
grep named /etc/passwd
#named:x:20:1000:BIND Owner:/var/lib/named:/bin/false
#Adapt the rights permissions
chown -R named:named /var/lib/named
#Check for your system if the main directory is a filesystem
grep named /etc/fstab
UUID=4bbbfb6e-e859-4a7c-96b0-2c5d38ea2d10 /var/lib/named
    btrfs subvol=@/var/lib/named 0 0

#########################################################
#######EDIT named.conf###################################
vi /etc/named.conf
```

```
key "rndc-key" {
algorithm hmac-md5;
secret "pf1gpo3YdpvepO3zDcicr+
    KUKO84jPUNm5h2yCJpiKSgxIcnGmDq8m7W7KkBnXjG/+
    pz8pPDpsW5PwTZcN4r1w==";
};

controls {
inet 127.0.0.1 port 53
allow { 127.0.0.1; } keys { "rndc-key"; };
};

options {
directory "/var/lib/named";
// version statement - inhibited for security
// (avoids hacking any known weaknesses)
version "hidden";
// optional - disables all transfers
// slaves allowed in zone clauses
allow-transfer {"none";};
// Closed DNS - permits only local IPs to issue recursive
    queries
// remove if an Open DNS required to support all users
// or add additional ranges
//allow-recursion {192.168.3.0/24;};
pid-file "/var/run/named.pid";
statistics-file "/var/run/named.stats";
};

zone "." {
type hint;
```

```
file "root.hint";
};

//If the root.servers files has not been defined
// BIND has its own compiled list of servers for class IN
    only.
// required zone for recursive queries
//zone "." {
//  type hint;
//  file "root.servers";
//};


zone "0.0.127.in-addr.arpa" {
type master;
file "pz/127.0.0";
};

zone  "unixmen.local"   {
type master;
file  "for.unixmen.local";
};

// required local host domain
zone "localhost" in{
type master;
file "localhost.zone";
allow-update{none;};
};

logging {
channel default_file {
file "/var/lib/named/log/logdefault.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel general_file {
file "/var/lib/named/log/loggeneral.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel database_file {
file "/var/lib/named/log/logdatabase.log" versions 3 size 5m
    ;
severity dynamic;
print-time yes;
};
channel security_file {
file "/var/lib/named/log/logsecurity.log" versions 3 size 5m
    ;
```

```
severity dynamic;
print-time yes;
};
channel config_file {
file "/var/lib/named/log/logconfig.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel resolver_file {
file "/var/lib/named/log/logresolver.log" versions 3 size 5m
    ;
severity dynamic;
print-time yes;
};
channel xfer-in_file {
file "/var/lib/named/log/logxfer-in.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel xfer-out_file {
file "/var/lib/named/log/logxfer-out.log" versions 3 size 5m
    ;
severity dynamic;
print-time yes;
};
channel notify_file {
file "/var/lib/named/log/lognotify.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel client_file {
file "/var/lib/named/log/logclient.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel unmatched_file {
file "/var/lib/named/log/logunmatched.log" versions 3 size 5
    m;
severity dynamic;
print-time yes;
};
channel queries_file {
file "/var/lib/named/log/logqueries.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel network_file {
file "/var/lib/named/log/lognetwork.log" versions 3 size 5m;
severity dynamic;
print-time yes;
```

```
};
channel update_file {
file "/var/lib/named/log/logupdate.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel dispatch_file {
file "/var/lib/named/log/logdispatch.log" versions 3 size 5m
    ;
severity dynamic;
print-time yes;
};
channel dnssec_file {
file "/var/lib/named/log/logdnssec.log" versions 3 size 5m;
severity dynamic;
print-time yes;
};
channel lame-servers_file {
file "/var/lib/named/log/loglame-servers.log" versions 3
    size 5m;
severity dynamic;
print-time yes;
};

category default { default_file; };
category general { general_file; };
category database { database_file; };
category security { security_file; };
category config { config_file; };
category resolver { resolver_file; };
category xfer-in { xfer-in_file; };
category xfer-out { xfer-out_file; };
category notify { notify_file; };
category client { client_file; };
category unmatched { unmatched_file; };
category queries { queries_file; };
category network { network_file; };
category update { update_file; };
category dispatch { dispatch_file; };
category dnssec { dnssec_file; };
category lame-servers { lame-servers_file; };
};
```

```
#Create Zone Files
#Now create the forward and reverse zone files which we
    defined in the above step.
#1. Forward Zone file
#Copy the existing zone file template /var/lib/named/
    localhost.zone to create a new forward zone file.
```

```
cp /var/lib/named/localhost.zone /var/lib/named/for.unixmen.
    local
vi /var/lib/named/for.unixmen.local
```

```
;
; BIND data file for forward.unixmen.local zone
;
$TTL    604800
@       IN      SOA     master.unixmen.local. root.unixmen.
    local. (
2         ; Serial
604800          ; Refresh
86400         ; Retry
2419200         ; Expire
604800 )        ; Negative Cache TTL
IN      A       10.0.2.15
;
@       IN      NS      master.unixmen.local.
@       IN      A       10.0.2.15
@       IN      AAAA    ::1
master  IN      A       10.0.2.15
```

```
#2. Reverse Zone file
#Copy the existing zone file template /var/lib/named
    /127.0.0.zone to create a new reverse zone file.
cp /var/lib/named/127.0.0.zone /var/lib/named/rev.unixmen.
    local
vi /var/lib/named/rev.unixmen.local
```

```
;
; BIND reverse data file for rev.unixmen.local
;
$TTL    604800
@       IN      SOA     master.unixmen.local. root.unixmen.
    local. (
3         ; Serial
604800          ; Refresh
86400         ; Retry
2419200         ; Expire
604800 )        ; Negative Cache TTL
;
@       IN      NS      master.
@       IN      A       10.0.2.15
101     IN      PTR     master.unixmen.local.
```

Review the root.int file that comes prepackaged (example for Opensuse)

```
ldsuze10:/var/lib/named # cat root.hint
;       This file holds the information on root name servers
    needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC
;       under anonymous FTP as
;           file                /domain/named.cache
;           on server           FTP.INTERNIC.NET
;       -OR-                    RS.INTERNIC.NET
;
;       last update:    Jan 3, 2013
;       related version of root zone:   2013010300
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.
    NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.      3600000      AAAA  2001:503:BA3E
    ::2:30
;
; FORMERLY NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.
    NET.
B.ROOT-SERVERS.NET.      3600000      A     192.228.79.201
;
; FORMERLY C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.
    NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; FORMERLY TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.
    NET.
D.ROOT-SERVERS.NET.      3600000      A     199.7.91.13
D.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:2D::D
;
; FORMERLY NS.NASA.GOV
;
.                        3600000      NS    E.ROOT-SERVERS.
    NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
```

```
;
; FORMERLY NS.ISC.ORG
;
.                        3600000      NS    F.ROOT-SERVERS.
    NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
F.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:2F::F
;
; FORMERLY NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.
    NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; FORMERLY AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.
    NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
H.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:1::803F
    :235
;
; FORMERLY NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.
    NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
I.ROOT-SERVERS.NET.      3600000      AAAA  2001:7FE::53
;
; OPERATED BY VERISIGN, INC.
;
.                        3600000      NS    J.ROOT-SERVERS.
    NET.
J.ROOT-SERVERS.NET.      3600000      A     192.58.128.30
J.ROOT-SERVERS.NET.      3600000      AAAA  2001:503:C27
    ::2:30
;
; OPERATED BY RIPE NCC
;
.                        3600000      NS    K.ROOT-SERVERS.
    NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
K.ROOT-SERVERS.NET.      3600000      AAAA  2001:7FD::1
;
; OPERATED BY ICANN
;
.                        3600000      NS    L.ROOT-SERVERS.
    NET.
L.ROOT-SERVERS.NET.      3600000      A     199.7.83.42
L.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:3::42
```

```
;
; OPERATED BY WIDE
;
.                       3600000      NS    M.ROOT-SERVERS.
    NET.
M.ROOT-SERVERS.NET.     3600000      A     202.12.27.33
M.ROOT-SERVERS.NET.     3600000      AAAA  2001:DC3::35
; End of File
```

```
#Check DNS configuration file using command
locate named-checkconf
#/usr/sbin/named-checkconf
named-checkconf /etc/named.conf
#Check Forward Zone:
named-checkzone unixmen.local /var/lib/named/for.unixmen.
    local
#zone unixmen.local/IN: loaded serial 2
#OK
#Check Reverse Zone:
named-checkzone unixmen.local /var/lib/named/rev.unixmen.
    local
#zone unixmen.local/IN: loaded serial 3
#OK
#Now start bind9 service.
/etc/init.d/named status
#Checking for nameserver BIND

                      unused
#named.service - LSB: Domain Name System (DNS) server, named
#   Loaded: loaded (/etc/init.d/named)
#   Active: active (running) since Tue 2016-06-28 23:09:16
    EDT; 6s ago
#  Process: 2644 ExecStart=/etc/init.d/named start (code=
    exited, status=0/SUCCESS)
#   CGroup: /system.slice/named.service
#           âŤŤâŤĂ2689 /usr/sbin/named -t /var/lib/named -u
    named

netstat -tanp |grep 53
#tcp        0      0 10.0.2.15:53            0.0.0.0:*
             LISTEN      2689/named
#tcp        0      0 127.0.0.1:53            0.0.0.0:*
             LISTEN      2689/named
#tcp        0      0 127.0.0.1:953           0.0.0.0:*
             LISTEN      2689/named

#retart after boot

##tests....
```

```
dig master.unixmen.local
dig -x master.unixmen.local
dig unixmen.local
```

6.2.4 Main files : named.conf

The file **named.conf** is the main configuration file of BIND. It is the first configuration file read by named, the DNS name daemon. The named.conf file contains statements that start with a keyword plus an opening curly brace "{" and end with a closing curly brace "}". A statement may contain other statements. The **forwarders** statement is an example of this. A statement may also contain IP addresses or the file word followed by a filename. These simple statements must be terminated by a semi-colon (;). All kinds of comments are allowed, e.g., // and # as end of line comments. See the named.conf(5) manual page for details.
The version below is taken from the Debian bind package (some comments removed) as a caching-only named.conf file. A typical named.conf file is organized similar to the following example:

```
<statement-1> ["<statement-1-name>"] [<statement-1-class>] {
<option-1>;
<option-2>;
<option-N>;
};

<statement-N> ["<statement-N-name>"] [<statement-N-class>] {
<option-1>;
<option-2>;
<option-N>;
};
```

```
options {
 directory "/var/named";
 // query-source address * port 53;
 // forwarders {
  //      0.0.0.0;
  // };
};
// reduce log verbosity on issues outside our control
logging {
 category lame-servers { null; };
 category cname { null; };
};
// prime the server with knowledge of the root servers
zone "." {
 type hint;
```

```
 file "/etc/bind/db.root";
};
// be authoritative for the localhost forward and reverse
   zones, and for
// broadcast zones as per RFC 1912
zone "localhost" {
 type master;
 file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
 type master;
 file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
 type master;
 file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
 type master;
 file "/etc/bind/db.255";
};
// add entries for other zones below here
```

6.2.5 Main files : named

The **named** name server daemon is the program that communicates with other name servers to resolve names. It accepts queries, looks in its cache and queries other name servers if it does not yet have the answer. Once it finds an answer in its own cache or database or receives an answer from another nameserver, it sends the answer back to the name server that sent the query in the first place. It is possible to send signals to the named process to control its behaviour. A full list of signals can be found in the named manpage. One example is the SIGHUP signal, that causes named to reload named.conf and the database files. Signals are sent to named with the kill command, e.g.,

```
kill -HUP 217
```

This sends a SIGHUP signal to a named process with process id 217, which triggers a reload.

6.2.6 Main files : rndc

The **rndc** (Remote Name Daemon Control) program can be used to control the named name server daemon, locally as well as remotely. It requires a **/etc/rndc.key** file which contains a key.

```
key "rndc-key" {
 algorithm hmac-md5;
 secret "QjByAdN4Tbb1FbUMnB+mqOe+
     q5ua1qYvymzuPsMWa4caMgkGpL82seHyjn+
     WAlh1c01qeJ3tnSGf7VMlDMaX3w==";
};
```

6.3 Comparison of DNS softwares

Refer to [9]

6.4 dnsmasq

6.4.1 dnsmasq : usage

dnsmasq is a **lightweight DNS, TFTP, PXE, router advertisement and DHCP server**. It is intended to provide coupled DNS and DHCP service to a LAN. Dnsmasq accepts DNS queries and either answers them from a small, local, cache or forwards them to a real, recursive, DNS server. It loads the contents of /etc/hosts so that local hostnames which do not appear in the global DNS can be resolved and also answers DNS queries for DHCP configured hosts.

[9] https://en.wikipedia.org/wiki/Comparison_of_DNS_server_software

It can also act as the authoritative DNS server for one or more domains, allowing local names to appear in the global DNS. It can be configured to do DNSSEC validation. The dnsmasq DHCP server supports static address assignments and multiple networks. It automatically sends a sensible default set of DHCP options, and can be configured to send any desired set of DHCP options, including vendor-encapsulated options. It includes a secure, read-only, TFTP server to allow net/PXE boot of DHCP hosts and also supports BOOTP. The PXE support is full featured, and includes a proxy mode which supplies PXE information to clients whilst DHCP address allocation is done by another server. The dnsmasq DHCPv6 server provides the same set of features as the DHCPv4 server, and in addition, it includes router advertisements and a neat feature which allows nameing for clients which use DHCPv4 and stateless autoconfiguration only for IPv6 configuration. There is support for doing address allo-cation (both DHCPv6 and RA) from subnets which are dynamically delegated via DHCPv6 prefix delegation. Dnsmasq is coded with small embedded systems in mind. It aims for the smallest possible memory footprint compatible with the supported functions, and allows uneeded functions to be omitted from the compiled binary.

6.4.2 dnsmasq : start/status

```
service dnsmasq status
#dnsmasq.service - DNS caching server.
#Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service;
    disabled)
#Drop-In: /run/systemd/generator/dnsmasq.service.d
#|__50-insserv.conf-$named.conf
....
```

6.5 djbdns

The djbdns software package is a DNS implementation written in 2001. It was created by Daniel J. Bernstein[10] in response to his frustrations with repeated security holes in the widely used **BIND ISC**[11] DNS software.[12]. It was released in the public domain in 2007. An exemple of recent fork is N-DJBDNS.[13]

6.5.1 djbdns : components

Servers

```
dnscache - the DNS resolver and cache.
tinydns - a database-driven DNS server.
walldns - a "reverse DNS wall", providing IP address-to-domain name lookup only.
rbldns - a server designed for DNS blacklisting service.
pickdns - a database-driven server that chooses from matching records depending
on the requestor's location. (This feature is now a standard part of tinydns.)
axfrdns - a zone transfer server.
```

Client tools

```
axfr-get - a zone-transfer client.
dnsip - simple address from name lookup.
dnsipq - address from name lookup with rewriting rules.
dnsname - simple name from address lookup.
dnstxt - simple text record from name lookup.
dnsmx - mail exchanger lookup.
dnsfilter - looks up names for addresses read from stdin, in parallel.
dnsqr - recursive general record lookup.
dnsq - non-recursive general record lookup, useful for debugging.
dnstrace (and dnstracesort) - comprehensive testing of the chains of authority
 over DNS servers and their names.
```

[10] https://en.wikipedia.org/wiki/Daniel_J._Bernstein
[11] https://en.wikipedia.org/wiki/BIND
[12] https://en.wikipedia.org/wiki/Djbdns
[13] http://pjp.dgplug.org/djbdns/index.html

6.5.2 djbdns : installation

See [14] that provides some required tricks to compile correctly some missing details provided in Bernstein Web page https://cr.yp.to/djbdns/install.html

```
#1#install ucspi-tcp
wget https://cr.yp.to/ucspi-tcp/ucspi-tcp-0.88.tar.gz
tar -xf ucspi-tcp-0.88.tar
cd ucspi-tcp-0.88
vi conf-cc
#add this to end of gcc line
-include /usr/include/errno.h
make
#2#install daemontools 0.70 or above
mkdir -p /package
chmod 1755 /package
cd /package
wget https://cr.yp.to/daemontools/daemontools-0.76.tar.gz
gunzip daemontools-0.76.tar.gz
tar -xpf daemontools-0.76.tar
rm -f daemontools-0.76.tar
cd admin/daemontools-0.76
#2.1# Extra step required from http://blog.tonycode.com/tech
    -stuff/setting-up-djbdns-on-linux/
#before running install edit this file
vi  src/conf-cc
#add this to end of gcc line
-include /usr/include/errno.h

#Instructions at the end of the compilation:
#Adding svscanboot to /etc/rc.local...

#2.2# Extra step (Franck) for opensuse...
cp /etc/rc.local /etc/init.d/svscanboot
chmod 755 svscanboot
#Reboot now to start svscan.
reboot
#
ps -ef|grep svscan |grep -v grep
#root 1739 1 0 21:58 pts/0 00:00:00 /bin/sh /command/
    svscanboot
#root 1741 1739 0 21:58 pts/0 00:00:00 svscan /service
#3#Installing djbdns,
tar -xf djbdns-1.05.tar
cd djbdns-1.05
echo gcc -O2 -include /usr/include/errno.h -include /usr/
    include/errno.h > conf-cc
make
```

[14] http://blog.tonycode.com/tech-stuff/setting-up-djbdns-on-linux/

```
make setup check
./install
./instcheck
```

6.5.3 djbdns : run

See http://cr.yp.to/djbdns/run-server.html. Let's consider that the IP adress of the server is 10.0.2.15 and heaven.af.mil the domain to manage :

```
#Creation of user, configuration and start
useradd Gtinydns
id Gtinydns
grep Gtinydns /etc/passwd
useradd Gdnslog
tinydns-conf Gtinydns Gdnslog /etc/tinydns 10.0.2.15
#
ln -s /etc/tinydns /service/tinydns
sleep 5
svstat /service/tinydns
#/service/tinydns: up (pid 1849) 18 seconds
```

6.5.4 djbdns : Publishing addresses

```
cd /service/tinydns/root
./add-host x.y.z 10.0.2.15
./add-alias www.x.y.z 10.0.2.15
```

6.5.5 djbdns : checking addresses

```
cat /service/tinydns/root/data
./add-ns heaven.af.mil 10.0.2.15
./add-ns 7.8.1.in-addr.arpa 10.0.2.15
./add-host lion.heaven.af.mil 10.0.2.15
./add-alias www.heaven.af.mil 10.0.2.15
make
#tinydns-get a www.heaven.af.mil
#1 www.heaven.af.mil:
#86 bytes, 1+1+1+1 records, response, authoritative, noerror
#query: 1 www.heaven.af.mil
#answer: www.heaven.af.mil 86400 A 10.0.2.15
#authority: heaven.af.mil 259200 NS a.ns.heaven.af.mil
#additional: a.ns.heaven.af.mil 259200 A 10.0.2.15
#dnsq a www.heaven.af.mil 10.0.2.15
```

6.6 PowerDNS

PowerDNS [15] is a Dutch supplier (2016 PowerDNS.COM BV - Dutch trade register number 27193521) of DNS software and services. The PowerDNS software is open source (GPL), and comes packaged with many distributions as pdns, powerdns-server, pdns-recursor or pdns-server. The system allows multiple backends to allow access to DNS configuration data, including a simple backend that accepts BIND style files.
PowerDNS Recursor (**pdns-recursor**) is a non authoritative/recursing DNS server.

There are two PowerDNS nameserver products: the Authoritative Server and the Recursor. While most other nameservers fully combine these functions, PowerDNS offers them separately, but can mix both authoritative and recursive usage seamlessly. The Authoritative Server will answer questions about domains it knows about, but will not go out on the net to resolve queries about other domains. However, it can use a recursing backend to provide that functionality. Depending on your needs, this backend can either be the PowerDNS recursor or an external one. When the Authoritative Server answers a question, it comes out of the database, and can be trusted as being authoritative. There is no way to pollute the cache or to confuse the daemon. The Recursor, conversely, by default has no knowledge of domains itself, but will always consult other authoritative servers to answer questions given to it. PowerDNS has been designed to serve both the needs of small installations by being easy to setup, as well as for serving very large query volumes on large numbers of domains. Additionally, through use of clever programming techniques, PowerDNS offers very high domain resolution performance. [16]

6.6.1 PowerDNS : install (OpenSuse)

```
zypper search pdns
#Loading repository data...
#Reading installed packages...

Name                      | Summary
--------------------------+----------------------------
pdns                      | Modern, advanced and high
                          | performance authoritative-only
                          | nameserver
pdns-backend-ldap         | LDAP backend for pdns
pdns-backend-lua          | Lua backend for pdns
pdns-backend-mydns        | MyDNS backend for pdns
pdns-backend-mysql        | MySQL backend for pdns
pdns-backend-postgresql   | PostgreSQL backend for pdns
pdns-backend-sqlite3      | SQLite 3 backend for pdns
pdns-recursor             | Modern, advanced and high
```

[15] https://www.powerdns.com
[16] https://doc.powerdns.com/md/

```
                          performance recursing non
                          authoritative nameserver
pdnsd                   | A caching dns proxy for small
                           networks or dialin accounts
pdnsd-doc               | Docs for pdnsd

 zypper install  php php-mcrypt php-pdo php-mysql pdns pdns-
    backend-mysql mysql-server httpd pdnsd
#...
updatedb
locate pdns.conf
#/etc/pdns/pdns.conf
#/usr/lib/tmpfiles.d/pdns.conf
```

```
#Install Latest mariadb and start the service
zypper install mariadb-server
zypper in mariadb-tools
service mysql start
service mysql status
systemctl start mysql.service
systemctl enable mysql.service
#Change root password...
mysqladmin -u root password
#New password:
#Confirm new password:
mysql_secure_installation
#Remove anonymous users? [Y/n] Y
#Disallow root login remotely? [Y/n] Y
#Remove test database and access to it? [Y/n] Y
#Reload privilege tables now? [Y/n] Y

#Connect to MariaDB as root
mysql -u root -p

#Create a DB and the DB Model
MariaDB [(none)]> CREATE DATABASE gmysql;
MariaDB [(none)]> connect gmysql
MariaDB [gmysql]>
CREATE TABLE domains (
id                    INT AUTO_INCREMENT,
name                  VARCHAR(255) NOT NULL,
master                VARCHAR(128) DEFAULT NULL,
last_check            INT DEFAULT NULL,
type                  VARCHAR(6) NOT NULL,
notified_serial       INT DEFAULT NULL,
account               VARCHAR(40) DEFAULT NULL,
PRIMARY KEY (id)
) Engine=InnoDB;
```

```
#..8<..
#..8<..
#Refer to the documentation for the full data model to
    create

service pdnsd status
/etc/init.d/pdnsd status
/usr/sbin/pdns_server --daemon
Jun 24 23:28:25 Reading random entropy from '/dev/urandom'
Jun 24 23:28:25 Loading '/usr/lib64/pdns/libgmysqlbackend.so
    '

zypper install policycoreutils  (for command sestatus )
sestatus
#SELinux status:                 disabled
```

Configure the Generic MySQL backend. This backend is called '**gmysql**', and needs to be configured in **/etc/pdns/pdns.conf**. Add the following lines, adjusted for your local setup (specifically, you may not want to use the 'root' user):

```
launch=gmysql
gmysql-host=127.0.0.1
gmysql-user=root
gmysql-dbname=pdns
gmysql-password=mysecretpassword
```

6.7 The dig and host utilities

The Internet Systems Consortium (ICS) has deprecated nslookup in favor of **host** and **dig**. However, **nslookup** is still widely used due to longevity of older Unix releases. It remains part of most Linux distributions too. Both dig and host commands can be used to query nameservers, it's a matter of preference which one to use for which occasion. **dig** has far more options and provides a more elaborate output by default. The help for both commands should give some insights in the differences.[17]

```
host
Usage: host [-aCdlriTwv] [-c class] [-N ndots] [-t type] [-
   W time]
[-R number] [-m flag] hostname [server]
-a is equivalent to -v -t ANY
-c specifies query class for non-IN data
-C compares SOA records on authoritative nameservers
-d is equivalent to -v
-l lists all hosts in a domain, using AXFR
-i IP6.INT reverse lookups
-N changes the number of dots allowed before root lookup is
    done
-r disables recursive processing
-R specifies number of retries for UDP packets
-s a SERVFAIL response should stop query
-t specifies the query type
-T enables TCP/IP mode
-v enables verbose output
-w specifies to wait forever for a reply
-W specifies how long to wait for a reply
-4 use IPv4 query transport only
-6 use IPv6 query transport only
-m set memory debugging flag (trace|record|usage)
-V print version number and exit
```

[17] Refer to 'The LPIC-2 Exam Prep - Copyright 2013 Snow B.V.'

```
dig -h
Usage:  dig [@global-server] [domain] [q-type] [q-class] {q
   -opt}
{global-d-opt} host [@local-server] {local-d-opt}
[ host [@local-server] {local-d-opt} [...]]
Where:  domain    is in the Domain Name System
q-class  is one of (in,hs,ch,...) [default: in]
q-type   is one of (a,any,mx,ns,soa,hinfo,axfr,txt,...) [
   default:a]
(Use ixfr=version for type ixfr)
q-opt    is one of:
-4                  (use IPv4 query transport only)
-6                  (use IPv6 query transport only)
-b address[#port]   (bind to source address/port)
-c class            (specify query class)
-f filename         (batch mode)
-i                  (use IP6.INT for IPv6 reverse lookups)
-k keyfile          (specify tsig key file)
-m                  (enable memory usage debugging)
-p port             (specify port number)
-q name             (specify query name)
-t type             (specify query type)
-u                  (display times in usec instead of msec)
-x dot-notation     (shortcut for reverse lookups)
-y [hmac:]name:key  (specify named base64 tsig key)
d-opt    is of the form +keyword[=value], where keyword is:
+[no]aaonly         (Set AA flag in query (+[no]aaflag))
+[no]additional     (Control display of additional section)
+[no]adflag         (Set AD flag in query (default on))
+[no]all            (Set or clear all display flags)
+[no]answer         (Control display of answer section)
+[no]authority      (Control display of authority section)
+[no]besteffort     (Try to parse even illegal messages)
+bufsize=###        (Set EDNS0 Max UDP packet size)
+[no]cdflag         (Set checking disabled flag in query)
+[no]cl             (Control display of class in records)
+[no]cmd            (Control display of command line)
+[no]comments       (Control display of comment lines)
+[no]crypto         (Control display of cryptographic
   fields in records)
+[no]defname        (Use search list (+[no]search))
+[no]dnssec         (Request DNSSEC records)
+domain=###         (Set default domainname)
+[no]edns[=###]     (Set EDNS version) [0]
+ednsflags=###      (Set EDNS flag bits)
+[no]ednsnegotiation (Set EDNS version negotiation)
+ednsopt=###[:value] (Send specified EDNS option)
+noednsopt          (Clear list of +ednsopt options)
+[no]expire         (Request time to expire)
+[no]fail           (Don't try next server on SERVFAIL)
```

```
 +[no]identify       (ID responders in short answers)
 +[no]ignore         (Don't revert to TCP for TC responses.)
 +[no]keepopen       (Keep the TCP socket open between
    queries)
 +[no]multiline      (Print records in an expanded format)
 +ndots=###          (Set search NDOTS value)
 +[no]nsid           (Request Name Server ID)
 +[no]nssearch       (Search all authoritative nameservers)
 +[no]onesoa         (AXFR prints only one soa record)
 +[no]opcode=###     (Set the opcode of the request)
 +[no]qr             (Print question before sending)
 +[no]question       (Control display of question section)
 +[no]recurse        (Recursive mode)
 +retry=###          (Set number of UDP retries) [2]
 +[no]rrcomments     (Control display of per-record comments
    )
 +[no]search         (Set whether to use searchlist)
 +[no]short          (Display nothing except short
 form of answer)
 +[no]showsearch     (Search with intermediate results)
 +[no]split=##       (Split hex/base64 fields into chunks)
 +[no]stats          (Control display of statistics)
 +subnet=addr        (Set edns-client-subnet option)
 +[no]tcp            (TCP mode (+[no]vc))
 +time=###           (Set query timeout) [5]
 +[no]trace          (Trace delegation down from root [+
    dnssec])
 +tries=###          (Set number of UDP attempts) [3]
 +[no]ttlid          (Control display of ttls in records)
 +[no]vc             (TCP mode (+[no]tcp))
 global d-opts and servers (before host name) affect all
    queries.
 local d-opts and servers (after host name) affect only that
     lookup.
 -h                          (print help and exit)
 -v                          (print version and exit)
```

Refer to https://www.madboa.com/geek/dig/
Example of a dig command result, line 15 : the default query is for an Internet address **(A)**.

```
dig www.isc.org                                                    1
                                                                   2
; <<>> DiG 9.10.4-P1 <<>> www.isc.org                             3
;; global options: +cmd                                           4
;; Got answer:                                                    5
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4051          6
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0,         7
    ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.isc.org.                   IN      A

;; ANSWER SECTION:
www.isc.org.            59      IN      A       149.20.64.69

;; Query time: 64 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Jun 29 22:58:52 EDT 2016
;; MSG SIZE  rcvd: 56
```

The short answer is given with the **+short** argument

```
dig www.isc.org +short
149.20.64.69
```

The reverse answer is given with the **-x** option

```
dig -x 8.8.8.8 +short
google-public-dns-a.google.com.
```

7 Create and maintain DNS zones

7.1 LPIC 2 objective 207.2

Candidates should be able to create a zone file for a forward or reverse zone or root level server. This objective includes setting appropriate values for records, adding hosts in zones and adding zones to the DNS. A candidate should also be able to delegate zones to another DNS server.

Key Knowledge Areas:

- BIND 9 configuration files, terms and utilities
- Utilities to request information from the DNS server
- Layout, content and file location of the BIND zone files
- Various methods to add a new host in the zone files, including reverse zones

The following is a partial list of used files, terms, and utilities:

- /var/named/*
- zone file syntax
- resource record formats
- **dig; nslookup; host**

The **$TTL** statement is the default **Time To Live** for the zone. When a name server requests information about this zone, it also gets the TTL. After the TTL is expired, it should renew the data in the cache.
The SOA resource record. The acronym SOA means Start Of Authority. It tells the outside world that this name server is the authoritative name server to query about this domain. The SOA record should contain the administrator contact address as well as a time-out setting for slave nameservers. Declaring a SOA record serves two aspects: first, the parent zone org has delegated (granted) the use of the example.org domain to us, the second is that we claim to be the authority over this zone. The SOA record is mandatory for every DNS zone file, and should be the first specified Resource Record (RR) of a zone file as well:

```
@ IN SOA lion.example.org. dnsmaster.lion.example.org. (          1
2001110700 ; Ser: yyyymmhhee (ee: ser/day start 00)              2
28800 ; Refresh                                                  3
3600 ; Retry                                                     4
604800 ; Expiration                                              5
3600 ) ; Negative caching                                        6
```

[Line 1] : @ represents the **current origin**, which expands to example.org

The **A** record is the **address** record. It connects an IP address to a hostname. An example record is:

```
lion IN A 224.123.240.1
```

8 Securing a DNS Server

8.1 LPIC 2 objective 207.3

Candidates should be able to configure a DNS server to run as a non-root user and run in a chroot jail. This objective includes secure exchange of data between DNS servers.

Key Knowledge Areas:

- BIND 9 configuration files
- Configuring BIND to run in a chroot jail
- Split configuration of BIND using the forwarders statement
- Configuring and using transaction signatures (TSIG)
- Awareness of DNSSEC and basic tools

The following is a partial list of used files, terms, and utilities:

- /etc/named.conf
- /etc/passwd
- DNSSEC
- **dnssec-keygen; dnssec-signzone**

This section is largely based on [18]

8.2 Making information harder to get

8.2.1 Hiding the version number

The below command will show the version of the BIND name server on host target:

```
dig @target test version.bind txt
```

The BIND version can be hidden by entering a **version** statement inside the options statement in named.conf. In the following example, the version will be set to the string hidden.

```
options {
// ...
// hide bind version
version "hidden";
};
```

8.2.2 Limiting access

There are several ways to limit access to name server data. First, an **access control list** must be defined. This is done with the **acl** statement.

```
acl "trusted" {
 localhost;
 192.168.1.0/24;
};
```

This acl defines an access control list with label **trusted**. ACL labels are used to simplify administration: you only need to update your ACLs in one place, after that each reference to a label will automatically point to the updated data. A nameserver supports **queries by resolvers** and **zone transfers** by other name servers. **Normal queries** occur when a resolver needs data only on a very limited subset of the data. For example it wants to know which IP address associates with a hostname. The **allow-query** statement controls from which hosts these queries are accepted.
The internet name space is build on the idea of a hierarchy of domains. To find the IP address of a host, you need its fully qualified domain name. That name consists of a series of subdomain names separated by dots, for example: www.example.com. The hostname is in the first part, i.e. "www". It is part of the "example" domain, which in turn is part of the "com" domain. Data for

[18] The LPIC-2 Exam Prep. Copyright 2013 Snow B.V.

each domain is served by nameservers, data that relates to one domain is called "a zone". Nameservers will sent out all data for a zone to allow slave (backup) servers to get the information from a master of the same zone. It is important to restrict these "zone transfers" to just the servers that need that data - no more. **Zone transfers** are controlled by the allow-transfer statement. This statement may also be specified in the zone statement, thereby overriding the global options **allow-transfer** statement. Traditionally, zone transfers can be initiated by anybody.

8.2.3 Limiting zone transfers

A sucessful failed transfer...

```
dig axfr @ns12.zoneedit.com zonetransfer.me
; <<>> DiG 9.9.9-P1 <<>> axfr @ns12.zoneedit.com
    zonetransfer.me
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

Only validated slave name servers should be allowed to issue a zone transfer request. This can be done by making adjustments to the configuration files (named.conf) of both the master and all slaves for a particular zone, see the next example.

On the master name server you should use the **allow-transfer statement to limit zone transfers** to a list of known slave servers.

```
acl "my_slave_servers" {
 224.123.240.3; // cat.example.org
};
// ...
zone "example.org" IN {
 type master;
 // ....
 allow-transfer {
  my_slave_servers;
 };
};
```

Now only the slaves (only the host cat in the example) can request a zone transfer from this master name server. **On a slave name server you should never allow any zone transfers. This can be achieved by setting the allow-transfer clause to 'none'**. DonâĂŹt forget to protect the reverse zone as well (with an allow-transfer statement)

8.2.4 Running BIND with less privileges

In some distributions, BIND runs as root. If BIND is compromised, the attacker may get root access. This can be prevented by running BIND under a non-privileged user and group.

8.2.5 Running BIND in a chroot jail

BIND comes in recent version by default with option -t that permit to chroot the corresponding directories:

```
ps -ef|grep named
#named     1204     1  0 18:13 ?        00:00:00 /usr/sbin/
    named -t /var/lib/named -u named

######
#-t directory
#Chroot to directory after processing the command line
    arguments, but #before reading the configuration file.
#Warning
#This option should be used in conjunction with the -u
    option, as #chrooting a process running as root doesn't
    enhance security
#on most systems; the way chroot(2) is defined allows a
    process with #root privileges to escape a chroot jail.
```

8.2.6 Securing name server connections

Restricting access to trusted hosts can be done using the allow-transfer and allow-query statements in named.conf and may help to limit risks. As the restrictions are based on IP addresses, there is still a residual risk that an enemy might use spoofing techniques. This might still allow him to misuse your system. Signing data may help prevent this. By having the server and client share a secret and by having them use that secret to sign the data between client and server we can be quite sure that all parties are whom they say they are. To further secure name server connections, a set of protocols called **DNSSEC** (Domain Name System Security Extensions) can be added as an extra layer of security to DNS.

8.2.7 dnssec-keygen

DNSSEC signs the messages it sends using a **MAC** (Message Authentication Code). The algorithm to compute a MAC has two inputs: the key and the message. The output is a fixed length string of hex digits. To sent a message, the MAC is computed using the key (or half of a keypair) and both the MAC and the message are sent to the other side. The other side will use the message to calculate the MAC itself using the shared key or the other half of the keypair

and if both hashes match there is a very high certainty the message has indeed not been altered and that it originates from the other side. Note that when you use a MAC the message itself will not be encrypted, it can be read by anybody. The **dnssec-keygen** command is part of the DNSSEC security extension for DNS.

```
#As an example, say we typed this command:
$  dnssec-keygen -a HMAC-MD5 -b 512 -n HOST peek.a.boo

dnssec-keygen - DNSSEC key generation tool

SYNOPSIS
dnssec-keygen [-a algorithm] [-b keysize] [-n nametype] [-3]
     [-A date/offset] [-C] [-c class] [-D date/offset] [-E
    engine] [-f flag]
[-G] [-g generator] [-h] [-I date/offset] [-i interval] [-K
    directory] [-L ttl] [-k] [-P date/offset] [-p protocol]
    [-q]
[-R date/offset] [-r randomdev] [-S key] [-s strength] [-t
    type] [-v level] [-V] [-z] {name}

DESCRIPTION
dnssec-keygen generates keys for DNSSEC (Secure DNS), as
    defined in RFC 2535 and RFC 4034. It can also generate
    keys for use with TSIG
(Transaction Signatures) as defined in RFC 2845, or TKEY (
    Transaction Key) as defined in RFC 2930.

The name of the key is specified on the command line. For
    DNSSEC keys, this must match the name of the zone for
    which the key is being
generated.

OPTIONS
-a algorithm
Selects the cryptographic algorithm. For DNSSEC keys, the
    value of algorithm must be one of RSAMD5, RSASHA1, DSA,
    NSEC3RSASHA1,
NSEC3DSA, RSASHA256, RSASHA512, ECCGOST, ECDSAP256SHA256 or
    ECDSAP384SHA384. For TSIG/TKEY, the value must be DH (
    Diffie Hellman),
HMAC-MD5, HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384,
    or HMAC-SHA512. These values are case insensitive.
If no algorithm is specified, then RSASHA1 will be used by
    default, unless the -3 option is specified, in which case
     NSEC3RSASHA1
will be used instead. (If -3 is used and an algorithm is
    specified, that algorithm will be checked for
    compatibility with NSEC3.)
```

```
Note 1: that for DNSSEC, RSASHA1 is a mandatory to implement
     algorithm, and DSA is recommended. For TSIG, HMAC-MD5 is
     mandatory.
Note 2: DH, HMAC-MD5, and HMAC-SHA1 through HMAC-SHA512
    automatically set the -T KEY option.
-b keysize
Specifies the number of bits in the key. The choice of key
    size depends on the algorithm used. RSA keys must be
    between 512 and
2048 bits. Diffie Hellman keys must be between 128 and 4096
    bits. DSA keys must be between 512 and 1024 bits and an
    exact multiple
of 64. HMAC keys must be between 1 and 512 bits. Elliptic
    curve algorithms don't need this parameter.
The key size does not need to be specified if using a
    default algorithm. The default key size is 1024 bits for
    zone signing keys
(ZSK's) and 2048 bits for key signing keys (KSK's, generated
     with -f KSK). However, if an algorithm is explicitly
    specified with
the -a, then there is no default key size, and the -b must
    be used.
-n nametype
Specifies the owner type of the key. The value of nametype
    must either be ZONE (for a DNSSEC zone key (KEY/DNSKEY)),
     HOST or ENTITY
(for a key associated with a host (KEY)), USER (for a key
    associated with a user(KEY)) or OTHER (DNSKEY). These
    values are case
insensitive. Defaults to ZONE for DNSKEY generation
```

8.2.8 dnssec-signzone

Similar to the dnssec-keygen command, BIND 9 includes the dnssec-signzone utility. As its name implies **dnssec-signzone** can be used to sign a zone.

```
Usage: dnssec-signzone [options] zonefile [keys]
```

The dnssec-signzone utility can be used to add **RRSIG** records to zones. **NSEC records (Next SECure)** are used to specify a range of non-existing domains. NSEC records were invented to solve a problem: if you queried a secured nameserver (one that signs the results of a query) and you would ask for a non-existing name, it would simply return - nothing. As you canâĂŹt sign âĂŹnothingâĂŹ you would never know if the anwer you got - nothing - really is what you should have gotten. Hence NSEC records. A NSEC record (Next SECure) states a range of names that do not exist. As NSEC records can be

signed the answer can be trusted again. An Example of an NSEC record for the dnssec-tools.org domain:

```
nis.zonetransfer.me. 10800 IN NSEC lena.zonetransfer.me.
```

The record specifies that no DNS records exist between nis.zonetransfer.me. and lena.zonetransfer.- me. , which would exclude brt.zonetransfer.me. from being an existing and therefore valid domain.

8.2.9 Internal DNS

One solution is to use what is known as a **split-level DNS setup**.

8.2.10 Configuring TSIG for BIND

TSIG (Transaction SIGnatures) provides a secured communication channel between nameservers for zone transfers. It is a resource friendly add-on, available in BIND version 8.2 and higher. TSIG uses shared keys and one-way hashing to provide authenticity and integrity.
We need a HMAC-MD55 keypair, which can be generated using the now familiar dnssec-keygen command:

```
dnssec-keygen -a HMAC-MD5 -b 512 -n HOST rndc-key
Krndc-key.+157+24876
```

The utility will generate two output files. Note that the keys in both files will always be duplicates as we required a HMAC-MD5.

```
ls -l Krndc-key.+157+24876 *
-rw------- 1 root bind 117 Jul 5 05:28 Krndc-key
   .+157+24876.key
-rw------- 1 root bind 229 Jul 5 05:28 Krndc-key
   .+157+24876.private
```

The contents of the private keyfile might be similar to this:

```
cat Krndc-key.+157+24876.private
Private-key-format: v1.3
Algorithm: 157 (HMAC_MD5)
Key: XIQDYlGaIbWfyopYHS1vtFrlfJiiIkiEbiNj4kN8Ke+
   F0hEqA7KVwcJMR/6URez32XBEmKZf2W1GUzjpbI2KJQ âĘŘ-
==
Bits: AAA=
Created: 20130705102726
Publish: 20130705102726
Activate: 20130705102726
```

Create a file to hold the secret you just generated and the IP addresses of slave nameservers. The name of the file can be freely chosen, in this example tsig.key will be used. The syntax of this file can be learned from this example:

```
key "TRANSFER" {
algorithm hmac-md5;
secret "XIQDYlGaIbWfyopYHS1vtFrlfJiiIkiEbiNj4kN8Ke+
    F0hEqA7KVwcJMR/6 âĘŘ-
URez32XBEmKZf2W1GUzjpbI2KJQ==";
};
# nameserver 2 (slave)
server 10.0.1.2 {
keys { TRANSFER; };
};
# nameserver 3 (slave)
server 10.0.1.3 {
keys { TRANSFER; };
};
```

Edit the BIND main configuration file named.conf and include the file like this:

```
include "/etc/bind/tsig.key";
```

Now, reload the BIND configuration:

```
 rndc reload
 server reload successful
```

Use **rndc tsig-list** to list the currently active TSIG keys. This completes the configuration on the master. Refer to detailed Administration guides for completing configurations.