

[UNIX] Hardening the BIND DNS Server

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2001-09/0085.html>

From: support@securiteam.com

Date: 09/27/01

From: support@securiteam.com

To: list@securiteam.com

Subject: [UNIX] Hardening the BIND DNS Server

Message-Id: <20010927102548.2A088138BF@mail.der-keiler.de>

Date: Thu, 27 Sep 2001 12:25:48 +0200 (CEST)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> – Know that you're safe.

Hardening the BIND DNS Server

SUMMARY

This paper presents the risks posed by an insecure DNS server and walks through compiling, installing, configuring, and optionally chroot'ing BIND 8. The test environment is Solaris 2.5, 2.6, 7 and 8. Many configuration and troubleshooting tips are provided, along with up-to-date references on BIND and alternatives for NT, Linux, and Solaris.

Your Domain Name Service is the road sign to your systems on the Internet. No matter how secure and robust your Web, mail and other servers are, compromised and corrupted DNS systems can prevent customers and legitimate users from ever reaching you. DNS, like many of the older protocols, was developed at a time when the Internet was a kinder, gentler place and was meant to provide a simple and unlimited way to provide information about what computers you have to anyone else. Obviously, the model of the Internet has changed, and changes to BIND (Berkeley Internet Name Domain software, the most common implementation of DNS), along with widely accepted configuration guidelines, have improved our ability to lock down DNS.

BIND is the most frequently used DNS server and maintained by the ISC. It is also known as "named", since this is the name of the actual daemon

itself. BIND has a long history, is a core tool for most Internet sites. As with many applications exposed to the increasingly hostile Internet environment, security weaknesses have been discovered in BIND.

Why Bother? What Risks Does an Insecure BIND Pose?

So what, you say? Yet another program with security problems? There are so many problems in so many applications these days, it is just not possible to keep up with all these advisories and patches. Do we really have to worry about DNS too? Well, a compromised DNS server can pose some interesting risks:

1) An attacker can gain much interesting information if zone transfers are allowed: the entire list of hosts and routers with IP addresses, names and possible even comments indicating location, names etc.

2) Denial of service: If all your Internet DNS servers go down,
* Your Website is no longer visible (other Websites cannot look up your IP address).
* Emails cannot be delivered (some other Internet sites that you frequently exchange data with may have cached DNS entries, but they will not last more than a few days).
* An attacker could start up a fake DNS server that pretends to be yours and delivers false DNS information to the Internet about your domain. That is, integrity is lost – see next section.

3) Loss of integrity: If an attacker can change the DNS data or spoof other sites into believing false data (this is known as DNS poisoning), it gets very dangerous:
* Fake Websites can be set up to look like yours and capture user input destined for your site, which may be anything from user/passwords to PINs to account information.
* All email can be diverted to a relay which can copy, change or delete email before passing it to your site.
* If your firewall or any Internet-accessible host uses DNS hostnames for authentication or trust relationships, these can be completely compromised, especially if a weak packet filter protects the Internet servers and Intranet. Imagine a Web proxy configured to only allow proxy requests from *.mydomain.com. The attacker adds his host to the domain, then the Web proxy may allow requests from him, allowing the attacker HTTP access to the Intranet. Imagine a system administrator who uses SSH (great crypto stuff), but the firewall hosts have a ".shosts" trust to "admin.mydomain.com", where "admin" is the administrator's workstation. If the attacker can replace the entry for "admin.mydomain.com" in the DNS, he has password-free access to the firewall hosts.

DETAILS

So What Needs to Be Done?

BIND weakness may be addressed with several prevention measures, but detection and reaction should not be forgotten either:

Securiteam: [UNIX] Hardening the BIND DNS Server

- 1) Resource isolation: Use a dedicated, hardened server for Internet DNS, do not share with other services, and especially do not allow user logins. Minimal services/users means reducing the amount of software running and hence the amount exposed to network attacks. Separation prevents other services or users possibly using local weakness in the system to attack BIND.
- 2) Redundancy: Install a secondary on a different Internet connection (foreign branch of your company, another ISP, etc.). If your site dies, at least other sites will not think you "cease to exist"; they just think you're "not available," so that emails, for example, won't get lost but will be queued (typically up to four days).
- 3) Use the latest version (i.e. 8.2.3 or 9.1 or later).
- 4) Access control: Restrict zone transfers to minimize the amount of information on your networks available to attackers. Consider using transaction signatures. Consider restricting/not allowing recursive queries.
- 5) Run BIND with minimum privileges: as a non-root user, with a tight umask.
- 6) More resource isolation: Run BIND in a "chroot" jail, so it is much more difficult for a compromised bind daemon to damage the operating system or compromise other services.
- 7) Configure BIND not to report its version number (see below). Some people do not believe in hiding the version number, as it is "security by obscurity", but we maintain that it at least helps against the script kiddies who are roaming the net looking for obvious targets. Defending against the pros is a different matter.
- 8) Detection: Monitor logs for unusual activity and the system for unauthorized changes with an integrity checker.
- 9) Keep an eye on relevant advisories, make sure you are notified of future BIND problems in a timely manner.

The procedure in this paper concentrates only on measures 4), 5) and 6), which should help to protect a server against possible future weakness in BIND. This procedure has been tested on several production systems: a secondary on Solaris 2.5 + 2.8, a primary on Solaris 2.6 + 2.7.

Install and Configure BIND

It is assumed that Solaris is up and running and appropriately hardened and that the BIND delivered with Solaris is disabled. If you have not yet hardened Solaris, check out the YASSP tool first. This section runs through:

Securiteam: [UNIX] Hardening the BIND DNS Server

1) Compiling BIND on a master host, since you probably do not have (or should not have) a compiler on the hardened DNS server.

2) Copying and installing BIND on the DNS server.

3) Setting up DNS data files.

4) Running BIND.

1. Compiling BIND (on a host with a compiler).

Download the distribution, and extract it to a subdirectory and compile.

This can be done as any user.

Note: It is recommended that "GNU make" be installed first, so that the installation to a temporary directory below works correctly.

```
cd src;
make clean;
make depend && make;
```

Now change to the root account, install to a temporary directory, and create a tarball:

```
su - root
# allow group, but not world access
umask 027
make install DESTDIR=/tmp
cd /tmp/usr/local
tar cf - * | compress > bind_dist.tar.Z
```

2. Copy and install BIND on the DNS server

Create a user and group account for BIND:

```
echo "named:x:20000:20000:BIND DNS daemon:/tmp:/bin/false" >> /etc/passwd
echo "named:NP:6445::::::" >> /etc/shadow
echo "named::20000:" >> /etc/group
```

Don't allow the BIND account to use ftp:

```
echo "named" >> /etc/ftpusers
```

Copy bind_dist.tar.Z to /usr/local on the DNS server and extract into /usr/local:

```
zcat bind_dist.tar.Z | tar xvf -
```

Set up file permissions:

```
cd /var; mkdir named
/usr/ucb/chown -R root.named named
chmod 770 named
```

Securiteam: [UNIX] Hardening the BIND DNS Server

```
touch /var/run/named.pid;
/usr/ucb/chown named.named /var/run/named.pid;

chmod 755 /usr/local /usr/local/bin /usr/local/sbin /usr/local/etc

/usr/ucb/chown root.named
/usr/local/bin/{dig,dnsquery,nslookup,nsupdate,host};
chmod 755 /usr/local/bin/
/usr/local/bin/{dig,dnsquery,nslookup,nsupdate,host};

/usr/ucb/chown root.named
/usr/local/sbin/{dnskeygen,irpd,named,named-bootconf,named-xfer,ndc};

chmod 755 /usr/local/sbin/{dnskeygen,
irpd,named,named-bootconf,named-xfer,ndc};
```

Optionally, you can rename the default DNS binaries installed on the system, so that they are not used by accident, and leave a reminder.

```
cd /usr/sbin;
mv nslookup nslookup.old
mv nsupdate nsupdate.old
mv in.named in.named.old
mv named-bootconf named-bootconf.old
mv named-xfer named-xfer.old
touch NAMED_NOW_IN_USR_SBIN
Add /usr/local/sbin to the root PATH in /.cshrc or /.profile.
```

3. Setting up DNS data files

```
cd /usr/local/etc
vi named.conf
chown root.named named.conf
chmod 640 named.conf
```

So what do you put in named.conf? Have a look at the examples provided in. The file consists of options, logging, ACL, server, and Zone sections. Some of the directives are:

- * The directory tells BIND where to look for data files.
- * Internal DNS servers without Internet access will need to use forwarders to forward all unknown queries to DNS servers which have Internet access.
- * The process number of BIND is stored according to the pid-file directive. The "named user" needs read and write access to this file.
- * BIND's logging is very flexible. The example show logs to the syslog local1.info facility.

* Access control lists (ACLs) can and should be used to restrict what servers are allowed zone transfers. It is recommended to use this feature to make it more difficult for attackers to map your network layout. Servers typically allowed in the list are primaries/secondaries for the domain, your ISP and NIC for your country.

* The version number that BIND reports to clients such as dig can be changed with the version directive. It is a good idea to set this, so stop attacker scanning for particular version (with known weaknesses/exploits) will not find useful information.

After setting up named.conf, the files containing the DNS records have to be set up on primaries (in /var/named in our example); these are automatically downloaded by secondaries.

4. Running BIND

Check the console and syslog (daemonlog) for errors, e.g.

```
tail -f local0log | grep "named"
```

Start BIND:

```
/usr/local/sbin/named -u named
```

Configure automatic starting on boot

In /etc/init.d/inetsvc, change the DNS startup lines to:

```
# Start the BIND DNS server:
if [ -f /usr/local/sbin/named -a -f /usr/local/etc/named.conf ]; then
  echo "Starting BIND domain name server."
  /usr/local/sbin/named -u named;
```

Before moving on to the next stage, BIND should be working well, with no errors in the logs. See also the troubleshooting section.

Chroot'ing BIND

This process has three steps: create a general chroot jail, install BIND into the jail, start, and test the chroot'ed BIND.

Set up a general chroot environment

BIND is now up and running, but we want to tighten security further by forcing it to run in a chroot environment (also called a jail or padded cell: Basically, restrict the files visible to BIND to a subdirectory within the file system). See also footnote for a discussion of chroot environments.

We will now walk through the steps for setting up the chroot environment, copying over the BIND files, starting BIND and troubleshooting. These steps chroot the entire BIND program, not just using BIND's "-t" feature (see Note 1).

Securiteam: [UNIX] Hardening the BIND DNS Server

The following steps assume use of the C-Shell. We start by setting a variable for the chroot environment (jail) location, and setting umask so that all files copied can be read by both groups and world. These commands are designed to be copied and pasted.

1. Set source and destination directories

```
csh
set jail='/home/dns';
umask 022;
```

2. Set up empty directories and links:

```
mkdir $jail;
cd $jail;
mkdir -p {dev,opt,usr,var,etc};
mkdir -p var/{run,log,named} usr/{local,lib};
mkdir -p usr/share/lib/zoneinfo;
```

3. Setup /etc

```
cp /etc/{syslog.conf,netconfig,nsswitch.conf,resolv.conf,TIMEZONE}
$jail/etc
```

4. Create a user and group account within chroot and for the whole system. BIND will run under this account.

```
echo "named:x:20000:20000:BIND DNS daemon:/tmp:/bin/false" >> /etc/passwd
echo "named:x:20000:20000:BIND DNS daemon:/tmp:/bin/false" >
$jail/etc/passwd
echo "named:NP:6445::::::" >> /etc/shadow
echo "named:NP:6445::::::" > $jail/etc/shadow

echo "named::20000:" >> /etc/group
echo "named::20000:" > $jail/etc/group
```

5. Set up libraries:

Use ldd to see what shared object libraries named and named-xfer rely on:

```
ldd /usr/local/sbin/named /usr/local/sbin/named-xfer
```

Copy the files listed by ldd, for example for Solaris 2.6/7:

```
cp -p /usr/lib/libnsl.so.1 \
/usr/lib/libsocket.so.1 /usr/lib/libc.so.1 \
/usr/lib/libdl.so.1 /usr/lib/libmp.so.2 $jail/usr/lib
```

On Solaris 2.5:

Securiteam: [UNIX] Hardening the BIND DNS Server

```
cp -p /usr/lib/libnsl.so.1\  
/usr/lib/libsocket.so.1 /usr/lib/libc.so.1\  
/usr/lib/libdl.so.1 /usr/lib/libmp.so.1 /usr/lib/libw.so.1\  
/usr/lib/libintl.so.1 $jail/usr/lib
```

Experience has shown the following are also needed for Solaris 2.5/6/7:

```
cp /usr/lib/ld.so.1 /usr/lib/nss_files.so.1 $jail/usr/lib
```

And with v8.2.3 on Solaris 8:

```
cp /usr/lib/libl.so.1 $jail/usr/lib
```

("Experience" means that first attempts did not work, but by running BIND with truss, one could see what libraries were being sought after.)

6. Copy over Timezone files (We use MET, here in Europe):

```
mkdir -p $jail/usr/share/lib/zoneinfo;  
cp -p /usr/share/lib/zoneinfo/MET $jail/usr/share/lib/zoneinfo/MET
```

7. Set up devices for communication, console, syslog, etc.

```
cd $jail/dev  
mknod tcp c 11 42  
mknod udp c 11 41  
mknod log c 21 5  
mknod null c 13 2  
mknod zero c 13 12  
chgrp sys null zero  
chmod 666 null  
mknod conslog c 21 0  
mknod syscon c 0 0  
chmod 620 syscon  
chgrp tty syscon  
chgrp sys conslog
```

Copying BIND to the Jail

We assume bind was already in /usr/local, so copy the BIND files over from there:

```
cd $jail;  
mkdir -p usr/local/{bin,lib,sbin,bind,etc}  
cd $jail/usr/local/sbin;  
(cd /usr/local/sbin; tar cf - dnskeygen named* irpd ndc ) |tar xvf -  
cd $jail/usr/local/bin;  
(cd /usr/local/bin; tar cf - dnsquery dig host nslookup nsupdate) |tar  
xvf -  
cd $jail/usr/local;  
cp /usr/local/etc/named.conf etc;  
(cd /usr/local; tar cf - bind) |tar xvf -
```

Securiteam: [UNIX] Hardening the BIND DNS Server

Your DNS data can be located in several directories; here we present two examples. The location is specified in named.conf.

1. Data in /etc/named

```
mkdir -p $jail/etc/named; cd $jail/etc/named;  
chgrp named $jail/etc/named;  
(cd /etc/named; tar cf - * ) | tar xvf -
```

For secondaries, allow named to create data files:

```
chmod 770 $jail/etc/named;
```

2. or DNS data in /var/named (my preference)

```
mkdir -p $jail/var/named; cd $jail/var/named;  
chgrp named $jail/var/named;  
(cd /var/named; tar cf - * ) | tar xvf -
```

Set Jail permissions

Next, set permissions on files, so that root owns files and named can read all files and write some files. And, disable any SUID/SGID files.

The PID file is put in /var/run and not /usr/local, because we don't want the named user to be able to write to /usr/local/etc (and hence named.conf). The location of the PID file is specified in named.conf.

```
jail=/home/dns  
cd $jail
```

```
# remove group write from var  
chmod -R g-w var;  
# remove all write access to opt and usr  
chmod -R a-w opt usr
```

```
# For secondaries, allow named to create data files:  
chown -R root:named $jail/var/named;  
chmod 770 $jail/var/named;
```

```
# For primaries, allow named to read, but not write data files:  
chown -R root:named $jail/var/named;  
chmod 750 $jail/var/named;  
chmod -R go-w $jail/var/named;
```

```
# Create empty log and pid files:  
touch var/log/all.log var/run/named.pid;  
chown named:named var/log/all.log var/run/named.pid;
```

```
# Allow named user/group to write logs and pid files:  
chgrp -R named $jail/var/log $jail/var/run;  
chmod 770 $jail/var/run $jail/var/log  
chmod -R o-rwx $jail/var/run $jail/var/log
```

Securiteam: [UNIX] Hardening the BIND DNS Server

```
# Allow named to access BIND config file:
chgrp named $jail/usr/local/etc;
chown root:named $jail/usr/local/etc/named.conf;
chmod 640 $jail/usr/local/etc/named.conf;
chmod 750 $jail/usr/local/etc;
```

```
# Remove SUID or SGID bits, if any exist:
find . -type f -exec chmod ug-s {} \;
```

See footnote for an example of an "ls -alR" on a production DNS primary.

Edit DNS config file: if the PID or data location has changed from your original installation, then \$jail/usr/local/etc/named.conf needs to be adapted (see also the section BIND Configuration Notes).

Starting BIND

The chroot environment is set up and BIND is installed, so the current (non-chroot'ed) BIND can be stopped and the new one started.

1) Set up a tail on the (syslog) logs, to watch BIND activity:
tail -f local0log | grep server1 The logs may be in /var/adm/messages or on a remote server, depending on your /etc/syslog.conf configuration. In this example, a centralized server collects logs and we look for messages from server1 (my test server).

2) Stop the existing BIND...
On Solaris 2.7: kill `pgrep named`
On Solaris 2.5/6: ps -ef |grep named then kill the appropriate PID.

3) Start BIND chroot'ed:
/usr/sbin/chroot /home/dns /usr/local/sbin/named -u named

4) Check for errors:
* in the syslog log
* do nslookups
* make sure the secondary can do zone transfers
* send a HUP signal to named, to ensure that it reloads configuration correctly.
* Check the domains using the IP-Plus tool.

5) If everything still looks good, change the /etc/rc2.d/S72inetsvc (or equivalent startup file) entries for starting BIND to something like this:

```
if [ -f /home/dns/usr/local/sbin/named -a -f
/home/dns/usr/local/etc/named.conf ]; then
/usr/sbin/chroot /home/dns /usr/local/sbin/named -u named;
echo "Started chroot'ed BIND domain name server."
fi
```

Troubleshooting

If you have a problem, a few tips:

- * Use dig or nslookup to check server results. Dig is better, as nslookup needs to find a PTR record for the DNS server; otherwise it won't start.

- * Client:

- * Check /etc/nsswitch.conf and /etc/resolv.conf.

- * Start nslookup with the "-d2" option to get buckets of debugging info, or start it without any arguments and type "help" at the prompt. There is also a "debug" command from the interactive prompt.

- * Try killing the nsd daemon.

- * Server

- * Send a HUP signal to named, to reread the config file after changes.

```
kill -HUP `cat /var/run/named.pid`
```

- * Look at the syslog entries. Typically logs are found in the syslog "daemon" section.

- * Named has a "-d X" option, which switches on debugging (X is a number indicating the debug level), for example:

```
/usr/sbin/chroot /home/dns /usr/local/sbin/named -u named -d 3
```

- * To get statistics from the name server into /usr/tmp/named.stats:

```
kill -ABRT `cat /var/run/named.pid`
```

- * If the logs indicate permission problems, check your file permissions against the example of an "ls -alR" on a production DNS primary.

- * If domain transfers are not working, try manual transfers, for example:

```
cd /var/named;
```

```
/usr/local/sbin/named-xfer -z DOMAIN.NAME -f test.results -d 3 -l test.log  
PRIMARY_NAME
```

Then look at "test.*" in /var/named. To test a chroot'ed BIND, try -

```
/usr/sbin/chroot /home/dns /usr/local/sbin/named-xfer -z DOMAIN.NAME -f  
test.results -d 3 -l test.log PRIMARY_NAME
```

and check /home/dns/test.results and test.log.*

- * ndc also has a -dt switch for debugging.

- * Make sure the option forward only has not been enabled by mistake in named.conf.

- * truss is very useful for following program execution, for example:

```
truss /usr/local/sbin/named -u named
```

```
truss /usr/sbin/chroot /home/dns /usr/local/sbin/named -u named
```

- * Check the domains using the IP-Plus tool.

- * If an error like No root nameservers for class IN appears, the root server list needs to be regenerated:

```
cd var/named
```

```
dig @A.ROOT-SERVERS.NET > named.root
```

```
chgrp named named.root
```

```
chmod 640 named.root
```

Or, you have defined forward only, but no forwarders in the named configuration.

- * Read the sections Known Problems and Configuration Notes below.

Known Problems

* BIND will still log to syslog "daemon" for certain events, even if the logging directive tells BIND to log to "local1" (for example).

* ndc does not work correctly in a chroot'ed environment. It would be better to start BIND via ndc: /usr/sbin/chroot /home/dns /usr/local/sbin/ndc -c /var/run/ndc start -u named rather than:

```
/usr/sbin/chroot /home/dns /usr/local/sbin/named -u named
```

One reader (J. S. Townsley) had similar problems, so he replaced ndc with a script:

```
#!/bin/sh
case "$1" in
start)
/etc/rc.d/init.d/named start;
;;
stop)
/etc/rc.d/init.d/named stop;
;;
restart)
/etc/rc.d/init.d/named restart;
;;
*)
/usr/sbin/chroot /chroot/named /usr/sbin/ndc $1
esac
```

Configuration Notes

Note 1: BIND's Built-in chroot

BIND 8 had its own chroot function, which works by giving named an option "-t" which points to the chroot jail, for example "named -t /home/dns."

When BIND starts up, it chroot's to the jail, after processing command line options and before it starts to answer queries. By compiling BIND and installing directly into the jail, all the bind programs will be correctly in place. Since BIND chroot's after reading user/group information, it does not need the other /etc and /usr/lib files noted above.

If BIND is set up as described above, it can still be started using this method, for example,

```
/home/dns/usr/local/sbin/named -t /home/dns -u named
```

This method is simpler than the general chroot method below, but it does have the disadvantage that we rely on the BIND code to be executed before the chroot is bug-free. In addition, a general chroot can be used to "jail" other programs too.

For the moment, we will continue using the general chroot method described below, and update this article as problems are found/solved. See also the Known Problems section.

Note 2: The BIND Configuration File

The following is an example named.conf with many comments to explain individual features. The file consists of an options section, ACL and server definitions, and Zone data.

```
acl "trusted-nameservers" {
    localhost;
    193.A.B.C; // my secondary
    193.A.B.D; // another secondary
    X.A.A.A; // ISP
    X.Y.Z.X; // NIC for your country
};

options {
    directory "/var/named"; /* /etc/named is also common */
    //forward only;
    // for Internal DNS servers, forward all unknown queries to external
    servers:
    //forwarders { 193.a.b.c; 193.a.b.d; };
    /* If there is a firewall between you and nameservers you want
    * to talk to, you might need to uncomment the query-source
    * directive below. Previous versions of BIND always asked
    * questions using port 53, but BIND 8.1 uses an unprivileged
    * port by default.
    */
    query-source address * port 53;
    pid-file "/var/run/named.pid";
    check-names master warn; /* default. */
    datasize 20M;
    stacksize 30M;
    statistics-interval 1440; // stats once per day is enough
    transfer-format many-answers; // faster transfers
    version "DNS server"; // hide BIND version

    // The following will restrict transfers for all zones, if enabled:
    // allow-transfer { trusted-nameservers; };
    recursion yes; // default

    //allow-query { 193.a.b.c/24; };
};
controls { unix "/var/run/ndc" perm 0600 owner 20000 group 20000; };

logging {
    channel syslog_errors {
        // Syslog logging: typically daemon, if we choose local1,
        // some message still go to daemon (bug), so we leave daemon for now .
        syslog daemon;
        //syslog local1; severity info;
    };
    channel file_log {
        file "/var/named/debug.log" versions 3 size 10m; // limit size + count
    };
};
```

Securiteam: [UNIX] Hardening the BIND DNS Server

```
severity dynamic; // catch debug messages
print-category no; // Category unneeded in debug file?
print-severity yes; print-time yes;
}
category default {
syslog_errors; // you can log to as many channels
//default_syslog; // by default goes to daemon in syslog
};
// ignore all "lame server" errors (only do this if none of the lame
// servers belong to you; otherwise, fix them)
category lame-servers { null; };
//category statistics { null; }; // We don't need stats for this server

// enable for testing/debugging:
//category default { file_log; syslog_errors; };
//category panic { file_log; };
//category packet { file_log; };
//category eventlib { file_log; };
//other categories; queries, cname, config, load, notify, parser,
//response-checks, security, statistics, update, xfer-in, xfer-out
};

// Example definition of a Primary
zone "mytestdomain.com" {
type master;
file "mytestdomain.com";
allow-query { any; }; // no restriction on queries
allow-update { none; }; // don't allow dynamic updates
allow-transfer { trusted-nameservers; }; // restrict zone transfers
};

// Example definition of a secondary
zone "mytestdomain2.com" {
type slave;
file "mytestdomain2.com";
masters { A.B.C-D }; // IP address of the primary
allow-query { any; };
allow-update { none; };
allow-transfer { trusted-nameservers; };
};

// if you get servers giving bad data, ignore them with:
//server 10.0.0.2 { bogus yes; };

// this is the main file for the domain name server. Each line gives
// the file where is stored the name table for a particular domain.
// named.root can be downloaded from ftp.rs.internic.net/domain

zone "." {
type hint;
file "named.root";
```

```
};  
  
zone "0.0.127.in-addr.arpa" {  
  type master;  
  file "db.127.0.0";  
};
```

Note 3: TSIG, Transaction Signatures

BIND provides some new security features in its latest release. Here we examine one: the use of TSIG (transaction signatures) to authenticate zone transfers.

Zone transfers are usually limited to a list of IP addresses (via the ACL mechanism) which correspond to specific DNS servers for a zone. Since only IP addresses are used, this mechanism is open to IP spoofing. BIND 8.2 and later allow authentication and verification of zone data. A key is configured on primary and secondary name servers and used to sign messages exchanged between the servers. It's important that the server times are synchronized. If the transfer is not authenticated with the correct key, no zone transfer may take place.

Let's look at an example where we use TSIG to restrict the zone transfers between a DNS primary "prim" (IP address 10.1.1.2) and DNS slave secondary "sec1" (IP address 10.1.2.2).

a) Generate an MD5 key, which will be used as a shared secret between the DNS servers. The "dnskeygen" tool is used. The key is written to a file.

```
# /usr/local/sbin/dnskeygen -H 128 -h -n prim-sec1.  
Generating 128 bit HMAC-MD5 Key for prim-sec1.  
Generated 128 bit Key for prim-sec1. id=0 alg=157 flags=513  
# cat Kprim-sec1.+157+00000.private  
Private-key-format: v1.2  
Algorithm: 157 (HMAC)  
Key: bFs2bXnLTYTI7r0WJv7HMA==
```

b) Create an identical key entry on both servers in named.conf:

```
key prim-sec1 {  
  algorithm hmac-md5;  
  secret "bFs2bXnLTYTI7r0WJv7HMA==";  
};
```

c) Add an ACL on both servers to limit transfers to specific hosts, for example:

```
acl "my-nameservers" {  
  localhost;  
  10.1.1.2;  
  10.1.2.2;  
};
```

d) For each host in the ACL, tell the BIND which key to use (do this for each server), for example on the primary:

```
server 10.1.2.2 {
  transfer-format many-answers;
  keys { prim-sec1 ; };
};
zone "mytestdomain.com" {
  type master;
  file "mytestdomain.hosts";
  allow-query { any; };
  allow-update { none; };
  allow-transfer { my-nameservers; };
}
```

d) Restart both named servers (send a HUP signal), then check the (syslog) logs for errors.

Testing if it works:

On the secondary, stop named, delete one of the zone files and restart. Watch the logs. A message indicating a successful zone transfer should appear and the corresponding zone file be created.

To make sure the TSIG key is really being used, change the key on the primary (e.g. add the letters "TEST" as the first few letters), stop both name servers, delete the zone file on the secondary, and restart both. The zone transfer will not take place, an error message like `named-xfer[16280]: [ID 745729 daemon.notice] SOA TSIG verification from server [10.1.1.2], zone mytestdomain.com: BADKEY (-17)` appears in the log, and the zone file is not recreated. Changing back the key on the primary and restarting BIND on primary and secondary fixes our deliberate fault.

Note the order of key, acl and server sections above. They must be in this sequence, as keys and ACLs must be defined before the server definition is parsed.

So it's really not difficult to significantly increase the security of your zone transfers. It is important that the file permissions on `named.conf` be restrictive so that it cannot be read by everyone on a system. The secret string used in the key must remain secret.

Note 4: More tips

Firewall filters: Resolvers and servers query other servers by initialising a connection from a high port (>1024) to port 53 on the target server.

* DNS uses port 53, both udp (resolving) and tcp (server to server communications).

* For server to server communications, the source server can be configured to use source port 53 (rather than a dynamic high port), with the option:

```
query-source * port 53;
```

* Typical firewall rules would be:

```
allow udp 53 in from outside to dns server [queries to your server]
```

```
allow udp 53 in from dns server to outside [queries from your server]
```

```
allow TCP 53 in from secondaries or ISP server to DNS server [zone transfers from your server]
```

```
allow TCP 53 out from DNS server to outside [zone transfers from primaries, for which you are a secondary]
```

Note: queries normally use UDP, but apparently also use TCP under load, so restrict queries to UDP may cause headaches in some situations.

* If you want to enable dynamic updates, despite the additional risk, use TSIG for better authentication of hosts allowed to make updates. Always restrict updates via an ACL.

* BIND can be configured to only allow resolver queries for explicit hosts or networks via the option:

```
allow-query { 193.a.b.c/24; };
```

An interesting way of using this is to restrict access globally (in 'options') to your own networks and then allow 'all' access in specific zones or subzones that you wish to be published to the Internet.

* Intranet DNS servers do not need to have Internet access. They can forward unresolved queries to the external Internet DNS server(s), with the forwarders command:

```
forwarders { 193.a.b.c; 193.a.b.d; };
```

Likewise, queries on intranet DNS servers can be restricted to valid intranet addresses via the "allow-query" command.

```
allow-query { 193.a.b.c/24; };
```

* Internet servers or 'delegate' servers should be configured to only allow "recursive queries" from valid DNS servers/clients in your company, but not the Internet. Recursive queries allow the DNS client to ask the DNS server for information on IP addresses for which it is not authoritative. The server will do it is best to get the needed information and will cache it, but we do not want that, we only want to serve information to others that we know is 100% correct, i.e. that the server is authoritative for. Stopping or restricting recursion can improve performance and help prevent a form of attack known as DNS cache poisoning. The appropriate option is:

```
allow-recursion { 193.a.b.c/24; };
```

* To stop recursion completely (e.g. on a delegate server), set the option:

```
recursion no;
```

In the same vein, bind can be prevented from automatically resolving name server names in NS or RDATA records by setting the option:

Securiteam: [UNIX] Hardening the BIND DNS Server

fetch-glue no;

* Read the book DNS and BIND.

ADDITIONAL INFORMATION

The information has been provided by <mailto:sean@boran.com> Sean Boran
for <<http://www.securityportal.com>> SecurityPortal.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

-
- **Previous message:** support@securiteam.com: "[\[TOOL\] SnortSam, Making Snort and Firewall-1 Work Together](#)"
 - **Messages sorted by:** [\[date \] \[thread \] \[subject \] \[author \] \[attachment \]](#)