Center for Internet Security®

CIS ISC BIND DNS Server 9.9 Benchmark

v3.0.0 - 12-21-2016

Table of Contents

Overview

This document is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate ISC (Internet Systems Consortium) BIND (Berkeley Internet Name Domain) DNS Server 9.9 running on Linux.

There are several environment variables defined to identify the BIND configuration files and directory paths which may differ for each installation. The variables are referenced by audit and remediation steps in order to make the benchmark as independent of installation specifics as reasonable.

- $CONFIG_FILES – List of the primary configuration file and all included configuration files. Typically /etc/named.conf and other included files. A recursive search for the "include" directive should locate all configuration files.
- $ZONE_FILES – All zone files referenced in the configuration files regardless of type.
- $BIND_HOME - Directory under which BIND runs, typically /var/named or a chrooted equivalent.
- $RUNDIR – Directory for temporary run time files, typically /var/run, /run or or chrooted equivalent.
- $DYNDIR – Directory for managed keys which are dynamically updated. Typically /var/named/dynamic/ or a chrooted equivalent.
- $SLAVEDIR – Directory for dynamically updated slave zone files. Typically /var/named/slaves/.
- $DATADIR – Directory for run time statistics.
- $LOGDIR – Directory for log files. Typically /var/named/slaves/
- $TMPDIR – Directory for temporary files.

Intended Audience

This document, CIS ISC BIND DNS Server Benchmark, provides prescriptive guidance for establishing a secure configuration posture for the ISC BIND DNS Server versions 9.9 running on Linux. This guide was tested using BIND version 9.9.4 installed from rpm packages on CentOS Linux 7.2. To obtain the latest version of this guide, please visit http://benchmarks.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@cisecurity.org.

Consensus Guidance

This benchmark was created using a consensus review process comprised of subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS benchmark undergoes two phases of consensus review. The first phase occurs during initial benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the benchmark. This discussion occurs until consensus has been reached on benchmark recommendations. The second phase begins after the benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the benchmark. If you are interested in participating in the consensus process, please visit https://community.cisecurity.org.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
Monospace font	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Scoring Information

A scoring status indicates whether compliance with the given recommendation impacts the assessed target's benchmark score. The following scoring statuses are used in this benchmark:

Scored

Failure to comply with "Scored" recommendations will decrease the final benchmark score. Compliance with "Scored" recommendations will increase the final benchmark score.

Not Scored

Failure to comply with "Not Scored" recommendations will not decrease the final benchmark score. Compliance with "Not Scored" recommendations will not increase the final benchmark score.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Authoritative Name Server**

  ISC BIND configured to be a master or slave authoritative name server, the server is authoritative for one or more domains. The name server is configured to not answer queries for any other domains for which it is not authoritative.

- **Caching Only Name Server**

  A caching only name server is not authoritative for any domain, but provides DNS service for a limited and well defined set clients and systems. The Caching Only Name Server will perform recursive DNS queries on behalf of its clients, and will cache answers to improve performance.

Acknowledgements

This benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Author
Ralph Durkee CISSP, GSEC, GCIH, GSNA, GPEN, C|EH, *Durkee Consulting, Inc.*
Dan Berry, *Leviathan Security Group*

Contributor
Tim Harrison CISSP, ICP, *Center for Internet Security*
Glenn Brunette
Brian Campbell, *Leviathan Security Group*
Blake Frantz
Dave Shackleford
Chad Thunberg, *Leviathan Security Group*
John Traenkenschuh
Rex Warren, *Leviathan Security Group*

Recommendations

1 Planning and Architecture

DNS name servers are a foundational part of your network architecture. How many name servers you need and what roles they should play depends on your organization's network architecture. For this reason, it is critical that the DNS strategy be considered early on, while decisions about the network topology are being formed. Questions that should be answered include, "how is the e-mail going to be delivered?", "are there going to be DNS sub-domains for the organization?", "is DHCP going to be used?", and "is Microsoft Windows Active Directory going to be used?" Providing the detailed information needed to make a recommendation for every possible DNS architecture is beyond the scope of this CIS Benchmark. However, some important DNS architectural recommendations and principles are discussed in this section.

1.1 Use a Split-Horizon Architecture (Not Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

Running a Split-Horizon DNS architecture refers to running authoritative DNS servers and services for external DNS queries separate from the internal authoritative DNS servers, which answer all queries originating from within the organization. The external servers are configured to provide only a limited amount of information for the services needed for communication with external clients and services. Typically, the information published in the externally available DNS is the minimal needed for the Internet services such as email, web and gateway systems such as VPNs. The separate internal DNS service typically provides a richer information set typically needed by internal clients.

Rationale:

The two goals of Split-Horizon are to:

1. Minimize the amount and type of externally available information.
2. Physical and logical separation of external and internal DNS services.

Separating the external and internal DNS servers in this manner adheres to a defense-in-depth approach that limits the potential damage and impact should the external name server be compromised, since it does not service internal clients, nor does it have information on the internal systems and services.

BIND 9 Views can be used to provide different responses based on the source IP address, and have been suggested by some as a means to implement split-horizon without having to separate the internal and external servers. However, the usage of views without separating the servers does not accomplish the second goal. In addition, the usage of views often erroneously assumes that source IP addresses are a reliable security control and cannot be spoofed. Therefore, it is necessary that the internal DNS server be located internally in a way that firewalls and other network controls will ensure external malicious queries will not reach the internal server.

Audit:

Review the network and DNS architecture, and identify the external authoritative DNS servers along with the internal authoritative DNS servers to ensure they are separate and serve only external and only internal clients respectively. Review the external DNZ zones to ensure only minimal name information is included in the external zones. Perform a query of an internal only name to the external DNS servers to ensure they do not provide a positive response.

Remediation:

Implement Split-Horizon Architecture to separate external and internal DNS services. The external DNS servers should respond only to names of approved external services, such as web, email and VPN services.

Default Value:

Not Applicable

References:

1. http://www.deer-run.com/~hal/EUGLUGBINDTalk.pdf

1.2 Do Not Install a Multi-Use System (Not Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

Default server configurations often expose a wide variety of services unnecessarily increasing the risk to the system. Just because a server can perform many services doesn't mean it is wise to do so. The number of services and daemons executing on the ISC BIND DNS server should be limited to those necessary, with the DNS service being the only primary function of the server.

Rationale:

Maintaining a server for a single purpose increases the security of your system. The more services which are exposed to an attacker, the more potential vectors an attacker has to exploit the system and therefore the higher the risk for the server. A DNS server should function as only a name server and should not be mixed with other primary functions such as email, web, or database.

Audit:

Leverage the package or services manager for your OS to list enabled services and review to ensure each service is necessary and has a documented business need. On Red Hat systems, the following commands will produce the list of current services enabled. The `systemctl` command lists any `systemd` based services, which are common on RHEL7, while `chkconfig` will list any traditional SysV based services.

```
# systemctl -t service --state enabled list-unit-files

# chkconfig --list | grep ':on'
```

Remediation:

Disable all unnecessary services or move necessary primary services other than DNS to another server. Leverage the package or services manager for your OS to uninstall or disable unneeded services. On Red Hat systems, the following commands may be used to uninstall a package or disable a service:

```
# yum erase

# systemctl disable .service
```

Default Value:

Depends on the platform

1.3 Dedicated Name Server Role (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

A name server may be an authoritative name server for one or more domains for which it is configured to provide information. An authoritative-only name server only answers queries on the domains for which it is configured, and will reject queries for other domains. A caching name server will answer queries any domain. The caching name server gets answers by sending recursive DNS queries to other name servers and then storing the answer in its cache to provide a quicker response to the next query for that name. A caching-only name server is not authoritative for any domain. The BIND DNS names server should be configured to be either a caching-only or an authoritative-only name server, but not both.

Rationale:

DNS name servers are a foundational part of your network architecture and the security of other network services depend on their integrity. It is important to separate the roles of caching and authoritative name servers to minimize functionality and reduce risk for each server. Each name server role faces different threats in addition to direct attacks on the server. For example, the caching name server faces unique threats of malicious replies with bogus answers or over-sized answers intended to deny service. The authoritative name server is a critical part of the infrastructure should not be exposed to these additional attacks.

Audit:

Authoritative-Only Name Server:
To audit an authoritative name server, ensure it doesn't answer queries for other non-authoritative domain names. The following command may be run on a Linux system other than the name server to verify the query status is refused.

```
# dig @<ip_address_of_nameserver> <non_authoritative_name> | grep status
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 52410
```

Alternatively the BIND configuration file `name.conf` and all included configuration files may be searched for the allow-recursion statement. No configured occurrences other than localhost and local loopback should be found.

```
# grep allow-recursion $CONFIG_FILES
named.conf: allow-recursion { 127.0.0.1; };
```

Caching-Only Name Server:
To audit a caching-only name server, send a query with the no recursion specified for a valid authoritative name in the organization.

```
# dig +norecurse @<ip_address_of_nameserver> <authoritative_name> | grep status
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 12379
```

Alternatively the BIND configuration file `name.conf` and all included configuration files may be searched for the zone statements. Zone that contain a local host name or a local loopback ip addresses should be allowed.

```
# grep -w zone $CONFIG_FILES

named.rfc1912.zones:zone "localhost.localdomain" IN {
named.rfc1912.zones:zone "localhost" IN {
named.rfc1912.zones:zone
"1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" IN {
named.rfc1912.zones:zone "1.0.0.127.in-addr.arpa" IN {
named.rfc1912.zones:zone "0.in-addr.arpa" IN {
```

Remediation:

Authoritative-Only Name Server:
For the authoritative-only name server add or modify the allow-recursion statement to only include the localhost to as shown below.

```
options {
. . .
allow-rescursion { local; };
```

Caching-Only Name Server:
For the caching-only name server remove the non-local zone statements from the configuration file and restart the server.

1.4 Use Secure Upstream Caching DNS Servers (Not Scored)

Profile Applicability:

- Caching Only Name Server

Description:

Caching name servers often forward queries to another caching name server to allow the name service work to be aggregated and improve performance by taking advantage of the cache of an upstream name server. The default caching name server provided by the Internet service provider is often used in this manner. This may also be a security weakness by relying on insecure servers outside the organization's control and security policies.

Rationale:

The security of all of the external connections that your systems on your network depend in part on getting accurate IP addresses for external names. If the upstream caching name server is compromised, or has its cache poisoned with malicious records, then your entire network may be subject to an attack which may redirect web, email, or VPN traffic to malicious servers, or may cause denial of services attacks. Therefore, it is important to evaluate the security of the upstream caching name servers to reduce the risk of DNS attacks propagated to your network via the upstream provider.

There are a number of security companies that offer secure caching DNS services that are worth considering. Features to look for and test include:

- Blocking of traffic to websites known to contain malware.
- Configurable categories for blocking inappropriate content, such as adult content.
- Detecting and blocking of malware communications to an external command and control server.
- Prevent DNS spoofing by ensuring the integrity and authenticity of all DNS responses.

Audit:

Perform the following for an audit:

- Check the network architecture and the identify all internal authorized caching DNS servers configured via DHCP or statically.

- Verify that there are network firewall and access controls rules that prevent internal systems from sending DNS queries directly to unauthorized external DNS servers. Only the authorized internal DNS servers should be allowed to send external DNS queries and they should be configured to only use authorized external DNS servers. An example direct DNS query on a Microsoft Windows system can be done via `nslookup`, and should timeout as shown below.

```
C:\> nslookup cisecuritry.org 8.8.8.8
DNS request timed out.
 timeout was 2 seconds.
Server: UnKnown
Address: 8.8.8.8
. . .
```

- Review the service provider's agreements, policies and statements, or speak with the vendor and consider if the security of the approved external DNS servers, meet your organizations security standards and requirement. The following features and risk mitigations are recommended for consideration:
    - Prevent spoofing of external DNS replies to redirect traffic to malicious server
    - Prevent spoofing of DNS queries to solicit large DNS replies to perform a denial of service.
    - Blocking of known malicious or infected websites
    - Blocking known botnet C&C communications.
    - Reporting, alerting and configuration on blocked DNS traffic.

Remediation:

Perform the following for remediation:

- Review network architectural, approved internal DNS servers, and block outbound DNS traffic, except for the approved DNS servers.
- Select an external DNS provider that sufficiently mitigates malicious DNS traffic to meet your organizational requirements
- Review, test and document the approved external DNS servers, and configure the internal caching-only DNS servers to use the approved external caching DNS server.

1.5 Installing ISC BIND 9 (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The ISC BIND Benchmark recommends using the binary packages provided by your platform vendor for most situations in order to reduce the effort and increase the effectiveness of maintenance and security patches. The Red Hat Enterprise Linux 7 has been used for testing the benchmark.

Rationale:

The benefits of using the vendor supplied binaries include:

- Ease of installation.
- It is customized for your OS environment.
- It will be tested and have gone through QA procedures.
- Additional software you may need is likely to be included, such as `chroot` setup and startup scripts.
- Your vendor will tell you about security issues so you have to look in less places.
- Updates to fix security issues will be easier to apply.

However, building from source is suitable for those that want full control of the build process, prefer to build from source, or do not have a suitable package available for their platform. Source download and build information is available on the ISC website knowledge base at the URL reference below.

Audit:

Perform the follow commands to check for an installed BIND rpm and to search the current path for the named executable.

```
# rpm -q bind
bind-9.9.4-29.el7_2.3.x86_64

# which named
/sbin/named
```

Remediation:

Installation depends on the operating system platform. The following commands were tested on RHEL7.

```
# yum install bind
. . .
# yum install bind-chroot
. . .
```

References:

1. https://kb.isc.org/article/AA-00768/0/Getting-started-with-BIND-how-to-build-and-run-named-with-a-basic-recursive-configuration.html

2 Restricting Permissions and Ownership

Security at the operating system (OS) level is the vital foundation required for a secure server. This section will focus on restricting platform permissions and privileges for the BIND DNS server.

2.1 Run BIND as a non-root User (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

To start BIND you must execute it as the root user. After the initial startup, BIND has the ability to change to a non-root user, allowing it to drop the root privileges.

Rationale:

The reason for configuring BIND to run as a non-root user is to limit the impact in case a future vulnerability is discovered and exploited. This is a common practice, which implements the principal of least privilege. This principle states that an entity, such as a service or user, should be granted only those specific privileges necessary to perform authorized actions. The server will still need to be started as root, but it should be configured to give up the root privilege after listening on port 53. The user ID under which named runs, needs to be created if it does not already exist and needs appropriate access to the DNS configuration and data files. Many systems including Red Hat Linux will come with a named user already created. Usage of the user and group id of 53 in the examples is arbitrary but is intended to be easier to recognize as it matches the listening port number.

Audit:

Perform the following to ensure the named account exists and has an appropriate non-root and non-user UID, and the `-u named` parameter is passed to `named` on startup.

1. Run the following commands to ensure the named account exists, and has been created with a UID greater than zero and less than MIN_UID.

```
# id named
uid=25(named) gid=25(named) groups=25(named)

# grep '^UID_MIN' /etc/login.defs
UID_MIN 1000
```

1. Verify that named service has been started and that the `-u named` option was passed when the daemon was executed, and that the user (first column) is equal to `named`.

```
# ps axu | grep named | grep -v 'grep'
named 423 0.0 1.0 236472 20512 ? Ssl Aug22 2:46 /usr/sbin/named -u named -t
/var/named/chroot
```

Remediation:

Create the named user and group if it does not already exist. Using a shell of /dev/null is best practice.

```
if ! id named; then
 groupadd -g 53 named
 useradd -m -u 53 -g 53 -c "BIND named" -d /var/named -s /dev/null named
fi 2>/dev/null
```

Add the `-u named` to the `OPTIONS` parameters in the `/etc/sysconfig/named` if not already present.

Default Value:

The default named startup parameters include the `-u named` value.

2.2 Give the BIND User Account an Invalid Shell (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The BIND user account, named by default, must not be used as a regular login account, and should be assigned an invalid or `nologin` shell to ensure that the account cannot be used to login.

Rationale:

Service accounts such as the `named` account represent a risk if they can be used to get a login shell to the system.

Audit:

Check the `named` login shell in the `/etc/passwd` file:

```
# grep named /etc/passwd
named:x:25:25:Named:/var/named:/sbin/nologin
```

The named account shell must be `/sbin/nologin` or `/dev/null` similar to the entry shown

Remediation:

Change the named account to use the `nologin` shell as shown:

```
# chsh -s /sbin/nologin named
```

Default Value:

/sbin/nologin

2.3 Lock the BIND User Account (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The user account under which BIND runs should not have a valid password, but should be locked.

Rationale:

As a defense-in-depth measure the named user account should be locked to prevent logins, and to prevent a user from `su`'ing to `named` using a password. In general, there shouldn't be a need for anyone to have to `su` as `named`, and when there is a need, then `sudo` should be used instead, which would not require the account password.

Audit:

Ensure the named account is locked using the following:

```
# passwd -S named
named LK 2016-07-10 -1 -1 -1 -1 (Password locked.)
```

The results will be similar to the output shown above.

Default Value:

Account is locked by default.

2.4 Set root Ownership of BIND Directories (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

All of the directories under which ISC BIND runs should be owned by root. Of course, any files created at run time by BIND will still be owned by named.

Rationale:

Restricting ownership of the directories provides defense in depth and will reduce the probability of unauthorized modifications to those resources. If there was a BIND vulnerability that allowed code execution as the named user, then the code would not be able to modify permissions on the BIND directories owned by root.

Audit:

Ensure that the variable $BIND_HOME is set to the directory under which BIND runs, typically the directory `/var/named/`. In the case of a `chroot`'ed configuration, the daemon will likely run under `/var/named/chroot/`, however the upper level directory of `/var/named/` should still be used as it is specific to the BIND service, and will include the `chroot` directory. Also the variable `$RUNDIR` should be set to the directory which is used to create run-time files such as the `pid` file and session-key. Perform the following to ensure the directory ownership:

```
# find $BIND_HOME $RUNDIR type d \! -user root -ls
```

There should be NO directories listed in the output from the find command.

Remediation:

To correct the directory ownership, perform the following:

```
chown -R root $BIND_HOME $RUNDIR
```

Default Value:

The following directories are owned by `named` in the default RHEL7 package install

- /var/named/dynamic
- /var/named/slaves
- /var/named/data
- /run/named

2.5 Set root Ownership of BIND Configuration Files (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The configuration files in the ISC BIND directories should be owned by root. Of course, any files created at run time by BIND, such as `pid` files, log files and slave zone files will necessarily be owned by named.

Rationale:

Restricting ownership of the configuration files provides defense in depth and will reduce the probability of unauthorized modifications to those important files. If there was a BIND vulnerability that allowed code execution as the named user, then the code would not be able modify the configuration files.

Audit:

Run the command below to ensure that all BIND configuration files are owned by root, except for those found in the run-time directories. Ensure that the BIND benchmark variables used below are set as described in the benchmark overview, as these variables identify the run-time directories. (`$DYNDIR`, `$SLAVEDIR`, `$DATADIR`, `$RUNDIR`, `$LOGDIR`, `$TMPDIR`) If a `chroot`'ed directory is not used, then `$LOGDIR` and `$TMPDIR` are not generally a subdirectory of `$BIND_HOME`, and the two directories may be omitted, however including them will not cause any errors or false positives.

```
# find $BIND_HOME -type f \! -user root | egrep -v \
 \^$DYNDIR\|\^$SLAVEDIR\^$DATADIR\|\^$RUNDIR\|\^$LOGDIR\|\^$TMPDIR
```

There should be no files listed in the output from the find command.

Remediation:

Perform the following:

- Capture the output of the previous audit command to a file named nonroot-files.txt and review any files not owned by root to ensure the files are necessary and are not expected run-time files. Delete any unnecessary files, and ensure any run-time files are being created in the appropriate run-time directory.

```
# find $BIND_HOME -type f \! -user root | egrep -v \
 \^$DYNDIR\|\^$SLAVEDIR\^$DATADIR\|\^$RUNDIR\|\^$LOGDIR\|\^$TMPDIR > \
$TMPDIR/nonroot-files.txt
```

- The remaining non-run-time files should be changed to be owned by root, with a command like the following:

```
# cat $TMPDIR/nonroot-files.txt | xargs chown root
# rm $TMPDIR/nonroot-files.txt
```

Default Value:

The default rpm has all configuration files owned by root.

2.6 Set Group named or root for BIND Directories and Files (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

All the BIND directories and files should have a group of either named or root.

Rationale:

In general the BIND directories and files default to a group of named, however some system files may have a group of root. Examples of system files include `chroot`'ed system device files. Either group root or named is accepted, as the intent is to prevent unexpected group ids, from getting inappropriate access to BIND files. Run time directories to which BIND will need write access should have a group of named, so that write access may be granted via the group permissions.

Audit:

Ensure that the BIND benchmark variables used below are set as described in the benchmark overview. Run the command below to ensure that all BIND directories and files have a group of either named or root.

```
# find $BIND_HOME $RUNDIR \! \( -group root -o -group named \) -ls
```

There should be no files listed in the output from the find command.

Remediation:

Run the command below to change all BIND directories and files to the group named.

```
chgrp -R named $BIND_HOME $RUNDIR
```

Default Value:

The default rpm install has all directories and files in the BIND home and the run time directory with a group of named.

2.7 Set Group and Other Permissions Read-Only for BIND Non-Runtime Directories (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

All the BIND directories except the run-time directories into which BIND will create files should have group and other permissions set to not be writable. No directories in the BIND_HOME or the RUNDIR should have other write permissions, even a `chroot`'ed `tmp` directory only needs to be writable by the named group.

Rationale:

Restricting permissions on the directories provides defense in depth and will reduce the probability of unauthorized modifications to important files. If there was a BIND vulnerability that allowed code execution as the named user, then the code would not be able create or modify configuration files.

Audit:

Ensure the BIND_HOME and runtime directory variables are set as specified in the overview without a trailing slash on the directory name. Run the commands below to ensure that all BIND directories are read-only for other, and read-only for group except for the expected run time directories where the named service will create files.

```
# find $BIND_HOME -type d -perm /020 | egrep -vx
$DYNDIR\|$SLAVEDIR\|$DATADIR\|$RUNDIR\|$LOGDIR\|$TMPDIR

# find $BIND_HOME $RUNDIR -type d -perm /002
```

There should be no files listed in the output from the find commands.

Remediation:

Perform the following:

- Capture the output from the audit commands above into a file named write-dirs.txt
- Review the purpose for the identified directories and either delete them if the directory is not needed, or change the permissions of the directory to not be writable by group or other.
- The following command can be used to change the permissions of the directories that are appropriate.

```
xargs -a write-dirs.txt chmod go-w
```

Default Value:

The default rpm install has all non-runtime directories without group or other write access.

2.8 Set Group and Other Permissions Read-Only for All BIND Files (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

All the files in BIND home and run time directories should have group and other permissions set to not be writable. Configuration files should, of course, not be writable by named, and any run time files created by BIND will be owned by named and writable by the user. Therefore, there are no exceptions required for the run time files.

Rationale:

Restricting permissions on the files provides defense in depth and will reduce the probability of unauthorized modifications to important files. If there was a BIND vulnerability that allowed code execution as the named user, then the code would not be able to modify configuration files.

Audit:

Run the command below to ensure that all BIND files are read-only for group and other. Note that a `chroot'ed` directory will have some special files which may need to be writable. Special files includes device files, like dev/null and a socket file for logging, but the -type f restricts the find to just regular files.

```
# find $BIND_HOME $RUNDIR -type f -perm /022
```

There should be no files listed in the output from the find command.

Remediation:

Perform the following:

- Capture the output from the audit commands above into a file with the name $TMPDIR/write-files.txt
- Review the purpose for the identified files and either delete them if the file is not needed, or change the permissions of the file to not be writable by group or other.
- The following commands can be used to change the permissions of the appropriate files.

```
# find $BIND_HOME $RUNDIR -type f -perm /022 > $TMPDIR/write-files.txt
# xargs -a $TMPDIR/write-files.txt chmod go-w
# rm $TMPDIR/write-files.txt
```

Default Value:

The default rpm install has all BIND files without group or other write access.

2.9 Isolate BIND with chroot'ed Subdirectory (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The `chroot()` system call causes an application to run with limited file system access so that a subdirectory becomes the root directory for the application environment. When this is done, the application is "jailed" and no longer has access to the entire file structure but is limited to the given subdirectory.

Rationale:

Although there are ways that a `chroot` jail can be broken, most methods require that a process be running as root in order to escape. Since BIND should be run as a different user than root, a `chroot` is an effective defense, to limit access to sensitive system configuration files. In the event that BIND has a vulnerability that allows code execution, the attack will not have access to the real system files such as `/etc/password`, but will be limited to the files placed in the `chroot` subdirectory.

Audit:

Run the following two commands to find the root directory of the currently running named process. If the named process is `chroot`'ed, then the listing will show a symbolic link to the `chroot` subdirectory. If process is not `chroot`'ed, then the symbolic link will point to the real root directory "`/`".

```
# NAMEDPID=$(pidof named)
# ls -ld /proc/$NAMEDPID/root
lrwxrwxrwx 1 named named 0 Sep 10 13:21 /proc/423/root -> /var/named/chroot
```

Remediation:

Perform the following:

- Stop the named service and install the `bind-chroot` package to provide the `chroot` directories.

```
# systemctl stop named.service
# yum install bind-chroot
```

- Edit the `/etc/sysconfig/named` configuration file to have a line similar to the one shown below that sets the `ROOTDIR` environment variable.

```
ROOTDIR="/var/named/chroot"
```

- Move all the configuration files and any master zone files into their respective directions under the subdirectory `/var/named/chroot/`
- It may be helpful to create symbolic links from a couple of system `/etc` files such as `/etc/named.conf` and `/etc/rndc.key` to the real files in the chroot-ed subdirectory, so that utilities like `rndc` will work as expected. **Do not create symbolic links or hard links from inside the chroot to external resources!** Instead use symbolic links to point from the outside resources into the `chroot`.
- Restart the named service and test the configuration.

```
# systemctl start named.service
```

Default Value:

The BIND service is not `chroot`'ed by default.

3 Restricting Queries

Recommendations in this section pertain to configurable access control mechanisms that are available in ISC BIND to restrict queries.

3.1 Ignore Erroneous or Unwanted Queries (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

BIND can be configured to ignore requests originating from specified network segments. This is accomplished by implementing the `blackhole` option in `named.conf`. It is recommended that this feature be implemented to ignore requests that originate outside of expected network segments.

Rationale:

By ignoring traffic that originates from unexpected networks, the server's exposure to malicious entities is reduced.

Audit:

Attempt to query the server from an address that has been placed in the `blackhole` list. If properly configured, the query will fail.

```
nslookup www.google.com ns1.example.com
```

Remediation:

Add a `blackhole` option for multicast and link local addresses, and all private RFC 1918 addresses that are not being used.

```
blackhole {
    // Private RFC 1918 addresses
    10/8; 192.168/16; 172.16/12;
    // Multicast
    224/8;
    // Link Local
    169.254/16;
};
```

Default Value:

No networks are `blackhole`'d by default.

3.2 Restrict Recursive Queries (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

A recursive DNS query is your typical DNS query from a client to a caching DNS server. It places the burden of finding the answer on the caching DNS server which will recursively query other DNS servers authoritative for the domains, until it gets the answer which is then returned to the client. The DNS server will then cache the answer to that query until its time-to-live expires in order to provide a quick answer to future queries for the same name. BIND can be configured to restrict fulfillment of recursive lookups to only authorized network segments and hosts. This is made possible by the `allow-recursion` option. Caching non-authoritative name servers should only allow recursive queries from clients on their own authorized networks. Authoritative name servers should not allow recursive queries, except to the local host.

Rationale:

Recursive DNS queries are commonly used in malicious attacks, including DNS amplification attacks and DNS cache poisoning attacks. A DNS amplification attack is a form of a reflected distributed denial-of-service attack, where multiple publicly accessible servers are sent recursive queries with the source IP address spoofed to be that of the victim. A high volume of relatively large DNS responses then flood the victim. For a DNS cache poisoning attack, the attacker may perform a query, and then provide a bogus response for the server to store in the cache. The bogus response may redirect clients to a different IP address which is provided by the attacker. Once the cache is poisoned, then clients visiting web sites, connecting to mail servers or VPNs may be connected with a malicious server configured to attack the client or steal credentials.

Limiting recursive queries to trusted networks does not prevent all of the DNS attacks possible, but it does make the attacks much more difficult and dramatically limits the scope of possible attacks so that detection and response are manageable.

Audit:

From the command prompt on Windows or Linux, send a recursive DNS request for a domain name for which the server not authoritative from a client that is not permitted to perform recursive queries:

```
$ nslookup www.cisecurity.com 10.10.3.53
. . .

** server can't find www.cisecurity.com: REFUSED
```

The expected result is for the query to be refused.

Remediation:

Authoritative Name Server:

For an authoritative name, insert the following either into the global options or into every zone section.

```
allow-recursion { localhost; };
```

Caching Name Server:

- Define an ACL named `trusted_clients` which will identify the networks which are expected to use the DNS caching server, and will be allowed to send recursive DNS queries.

```
acl trusted_clients { 10.19.4.0/28; . . . }
```

- Insert the following into the global options.

```
allow-recursion { localhost; trusted_clients };
```

Default Value:

The allow-recursion option is not defined by default.

References:

1. https://www.us-cert.gov/ncas/alerts/TA13-088A
2. https://www.godaddy.com/help/what-risks-are-associated-with-recursive-dns-queries-1184

3.3 Restrict Query Origins (Not Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

BIND can be configured to restrict access to its query services based on source IP address. It is recommended that the `allow-query` option be used to restrict access to only the networks authorized to use the name server. For an external authoritative only name server, the authorized networks may include all networks, however for internal authoritative or caching name servers the authorized networks should be explicitly configured.

Rationale:

Using allow-query in conjunction with an ACL of trusted networks will reduce the risk of unauthorized access to name services content. Additionally, the exposure of vulnerabilities present in BIND's query handlers is reduced by this configuration as requests with an untrusted source will be rejected before the request is fully parsed by named. Keep in mind however, that the source IP addresses can be easily spoofed, and the firewall and network architecture also needs to protect internal name servers from external spoofed requests.

Audit:

Verify that the BIND configuration files contain a global allow-query option with only the predefined ACL `localhost` and an ACL of the explicitly authorized networks. For an external authoritative only name server, the authorized networks may be the ACL "`any`" which represents any IPv4 or IPV6 host, but for caching and internal name servers, the `authorized_networks` should be an ACL with an explicit list of networks. The name of the ACL does not have to be "`authorized_networks`".

```
$ grep allow-query $CONFIG_FILES
    allow-query   { localhost; authorized_networks };
```

For an external authoritative only name server:

```
$ grep allow-query $CONFIG_FILES
    allow-query   { any };
```

Remediation:

For remediation:

- Create an ACL for the authorized trusted networks in the `named.conf` file.

```
acl authorized_networks { 10.10.32.0/24; 10.10.34.0/24; . . .  };
```

- Add the allow-query statement to the global options of the `named.conf` file with the localhost ACL and the authorized trusted networks ACL.

```
allow-query   { localhost; authorized_networks };
```

Default Value:

The default package install allows queries only from localhost.

3.4 Restrict Queries of the Cache (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The BIND option `allow-query-cache` may be used to restrict or allow BIND to provide answers to queries from the current cache of previously resolved queries. An authoritative only name server should not allow cache queries, except from the localhost, A caching only name server should allow cache queries only from the list of authorized networks.

Rationale:

Caching only name servers are critical to the security of all of the clients and servers using them, only the local authorized networks should be allowed to perform queries of the server's cache. In addition to malicious malformed queries, an attacker could use information about what is or is not in the name servers cache to help setup a DNS attack against the systems using the caching name server.

Audit:

From the command prompt on a Microsoft Windows or Linux client not on the authorized network, send a non-recursive DNS request for a domain name for which the server is not authoritative, as shown below.

```
$ nslookup -norecursion www.cisecurity.org 10.3.5.53
Server: UnKnown
Address: 10.3.5.53
*** UnKnown can't find www.cisecurity.org: Query refused

-or-

** server can't find www.cisecurity.org: REFUSED
```

The correct response should say REFUSED, or Query Refused as shown above. If an IP address is returned, then the server accepted the quest and returned the value from the cache. If the reply includes any of the phrases “can't find” or “No Answer”, or “Server failed” similar to the samples below, then the request was allowed, and the value was not in the cache.

```
Example of Query responses that were allowed, but not in the cache

*** Can't find www.cisecurity.org: No answer

*** UnKnown can't find www.cisecurity.org: Server failed
```

Remediation:

Authoritative Only Name Server:

For an authoritative name, insert the following either into the global options or into every zone section.

```
allow-query-cache { localhost; };
```

Caching Only Name Server:

Use the previously defined an ACL named `trusted_clients` which will identify the networks which are expected to use the DNS caching server, and will be allowed to send DNS cache queries.

```
allow-query-cache { localhost; trusted_clients };
```

Default Value:

If the `allow-query-cache` option is not present in the configuration, the default value is the `allow-recursion` setting. If the `allow-recursion` setting is not present, then the `allow-query` setting is used, unless recursion is set to no. If recursion is set to no, then the default value is none. Otherwise, if `allow-query` is also not present then the default value is `localnets` and localhost.

References:

1. https://kb.isc.org/article/AA-00845/0/BIND-9.9-Administrator-Reference-Manual-ARM.html

4 Transaction Signatures -- TSIG

Transaction Signature (TSIG) is used by BIND to ensure the authenticity and integrity of DNS requests and responses. TSIG is implemented by generating a secure hash of the DNS data combined with a shared secret. Since TSIG depends on a shared secret between the two servers it is really only suitable for server to server communications such as authenticating your organization's or possible partnering organization's DNS servers. There are a few critical DNS functions for which using TSIG works well to provide authentication: zone transfers, notifications and dynamic updates. This section will discuss the secure generation and protection of the keys used for TSIG communication, and discussion of specific usage of TSIG for securing transfers and updates will follow.

4.1 Use TSIG Keys 256 Bits in Length (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The TSIG secret keys used by the name server should be generated from a good source of entropy and should be at least 256 bits in length.

Rationale:

Weak cryptographic keys may allow cryptographic attacks to discover the key value, through repeated guesses. A strong HMAC key requires a good source of entropy and at least 256 bits in length.

Audit:

To check if the secret key is at least 256 bits long before encoding then the base64 encoded value should be 44 characters long or longer with the padding and a trailing "=". To easily count the number of characters, copy the key value into the quoted value in the command below.

```
$ echo -n "ezoZopbE4Q73HShuFYlf3FRvLWjtNXI5fd0TeQAYOug=" | wc -c
44
```

Remediation:

For remediation, replace any keys which are too short with a securely generated key with a length of 256 or 512. The `dnssec-keygen` command below can be used to generate a key.

```
$ dnssec-keygen -a HMAC-SHA256 –b 256 -n HOST ns1-ns2.cisecurity.org.

$ cat Kns1-ns2.cisecurity.org.+163+21730.key
ns1-ns2.cisecurity.org. IN KEY 512 3 163 ezoZopbE4Q73HShuFYlf3FRvLWjtNXI5fd0TeQAYOug=
```

Default Value:

The `rndc` key is generated as 128 bits during `bind-utils` package install, while the `nsupdate` session key is dynamically generated with a length of 256 bits.

4.2 Include Cryptographic Key Files (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

Do not place keys directly in the BIND `named.conf`, but use separate configuration files for the keys and include them into the `named.conf` file, in order to protect the keys from unintentional disclosure.

Rationale:

Although the keys may be placed directly in the named.conf file, putting it in a separate file will limit the number of times it needs to be viewed, and make it independent of viewing and changes to the main configuration file.

Audit:

Use the grep command below to search the `named.conf` file to ensure it doesn't have any secret keys placed in the file.

```
# grep -C 3 secret /etc/named.conf

key host1-host2.cisecurity.org {
    algorithm hmac-sha256;
    secret "1R3DP9D81/yWXjqf3hlg2beRpti1883JnZ3s7RVb1HU=";
};
```

Remediation:

Move each key definition statement from the `named.conf` file into its own key file. It is recommended to name both the key and the key file after the two hosts that will be sharing the secret key, in order to avoid confusion. Then include the key files with include statements in the `named.conf`. An example is shown below with the key definition statement moved to a separate key file, however it is also accepted for only the secret statement to be moved to another file.

```
# grep -C 1 include /etc/named.conf

// Include the key file used for the host1 and host2 TSIG comms
include "/etc/private/host1-host2.cisecurity.org.key";
```

```
# cat /var/named/chroot/etc/private/host1-host2.cisecurity.org.key
key host1-host2.cisecurity.org {
 algorithm hmac-sha256;
 secret "1R3DP9D81/yWXjqf3hlg2beRpti1883JnZ3s7RVb1HU=";
};
```

Default Value:

During a default install an `rndc` key is generated in a separate file `/etc/rndc.key` and included in the `named.conf`.

4.3 Use Unique Keys for Each Pair of Hosts (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

A unique TSIG key should be used for each pair of communicating hosts. For example if there is one master authoritative name server and three slave authoritative name servers that were updated by the master, then there would need to be a unique TSIG key for at least the following:

- Master <-> Slave1
- Master <-> Slave2
- Master <-> Slave3

Rationale:

Each communication channel should have a unique key, to reduce the risk of key disclosure. If one of the TSIG keys or one of the slave servers is compromised, then the remaining TSIG keys are not disclosed.

Audit:

To verify each key is unique, and has unique usage, perform the following:

- The sample command below will extract the secret keys from the configuration files and count the number of occurrences of each key value.

```
# cat $CONFIG_FILES | egrep -o "secret.*;" | sort | uniq -c
      1 secret "R/eBXL/5xso142dGZSGJixKAAW+bO1UHlIpxZAj92Cc=";
      2 secret "P3/AuCgxdt3buLyeb/QxRmPe9IfMwsXRrKyNvQSbN1k=";
      1 secret "SGNiICKGf86GbhzpDBZOkQ==";
      1 secret "gyxEId4g2gB+pVJSKXA=";
```

The count occurrences preceding each key should be one in the output.

- Search the configuration files for duplicate uses of the same key name. The command below will extract references to key names.

```
# egrep "keys +{.+}"  $CONFIG_FILES
named.conf:        allow { 127.0.0.1; } keys { "rndc-key"; };
ns1-ns2.cisecurity.org.key:        keys { "ns1-ns2.cisecurity.org"; };
ns1-ns3.cisecurity.org.key:        keys { "ns1-ns3.cisecurity.org"; };
```

Each key name should be referenced only once.

Remediation:

Generate unique keys for host to host communication. The command below can be used to generate 2 files, and `<anem>`.key file and a `<name>.private` file with secret keys of suitable length with base64 encoding. The files themselves are not needed, and should be securely deleted once the values are copied into a key file for including in the named configuration.

```
$ dnssec-keygen -a HMAC-SHA256 –b 256 -n HOST ns1-ns2.cisecurity.org
Kns1-ns2.cisecurity.org.+163+13013

$ cat Kns1-ns2.cisecurity.org.+163+13013.key
ns1-ns2.cisecurity.org. IN KEY 512 3 163 9FQ2dYCQ17HJwDi/uHgANh2dlb8M7eb+F4AjML8tTdA=
```

Default Value:

The `rndc` key is automatically generated during package installation.

References:

1. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-81-2.pdf

4.4 Restrict Access to All Key Files (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The TSIG keys should be readable only by the named and root accounts. No other user accounts or groups should have read access. Note that BIND often creates a session key on startup for usage by `nsupdate -l`. Both the `$BIND_HOME` and `$RUNDIR` are included since the session key should also have the recommended permissions.

Rationale:

The secret key protects the authenticity and integrity of TSIG communications and disclosure of a key would allow an attacker to perform the authenticated operations such as `rndc` administrative operations, zone transfers or dynamic updates.

Audit:

Perform the following to audit the recommendation:

- Find all of the TSIG key files in the `$BIND_HOME` and `$RUNDIR` directory, and capture the list to a file named `key_file.txt` in a `tmp` directory with the command below. If the `RUNDIR` is a subdirectory of `BIND_HOME`, which is typical for a `chroot'ed` directory, then some key files may be found twice. Duplicates are removed by the final sort command.

```
# find $BIND_HOME $RUNDIR -type f | xargs fgrep -l secret | sort -u >
$TMPDIR/key_files.txt
```

- Check for appropriate ownership, group and permissions on the files with the following commands.

```
# find $(cat $TMPDIR/key_files.txt ) \! \( -user root -o -user named \) -ls
# find $(cat $TMPDIR/key_files.txt ) \! \( -group root -o -group named \) -ls
# find $(cat $TMPDIR/key_files.txt ) -perm /022
```

- There should be no output from the three find commands. Remove the temporary file.

```
rm $TMPDIR/key_files.txt
```

Remediation:

Perform the following for remediation:

- Use the command below to find secret key files. Review the list of key files, and delete any unused or unnecessary key files. Recreate the file list, after deleting any unused files.

```
# find $BIND_HOME $RUNDIR -type f | xargs fgrep -l secret | sort -u >
$TMPDIR/key_files.txt
```

- Change the ownership, group and permissions on the key files.

```
# xargs -a $TMPDIR/key_files.txt chown -R root
# xargs -a $TMPDIR/key_files.txt chgrp -R named
# xargs -a $TMPDIR/key_files.txt chmod o-r
```

- Remove the temporary file,

```
rm $TMPDIR/key_files.txt
```

Default Value:

Ownership, Group and Permissions are correct for any default key files.

4.5 Protect TSIG Key Files During Deployment (Not Scored)

Profile Applicability:

- Authoritative Name Server

Description:

Do not expose the TSIG key files through insecure network transmission of the files when deployed, or via insecure permissions or shares on any intermediate systems used for the key deployment.

Rationale:

The secret key protects the authenticity and integrity of TSIG communications and disclosure of a key would allow an attacker to perform the authenticated operations such as `rndc` administrative operations, zone transfers or dynamic updates.

Audit:

Review the technical procedure for generating and deploying the TSIG keys to ensure the files are not inappropriately disclosed on the original systems where the key is generated, on any intermediate systems, or file shares. Also, ensure that the process does not allow the keys to be copied over the network via clear text of weak file transfer protocols, such as telnet, ftp or rcp.

Remediation:

Perform the following:

- Correct the deployment procedure to ensure secure transmission and intermediate storage protection of keys during deployment.
- Regenerate new keys via the corrected procedure and replace all previous TSIG keys.

5 Authenticate Zone Transfers and Updates

Recommendations in this section pertain to the configuration of secure DNS Zone transfers and dynamic updates to ensure the authenticity and integrity of the requests.

5.1 Securely Authenticate Zone Transfers (Scored)

Profile Applicability:

- Authoritative Name Server

Description:

A zone transfer is a mechanism commonly used by DNS deployments to replicate zone information from master/primary servers to slave/secondary servers. Each pair of name servers participating in zone transfers should authenticate the requests and ensure the integrity of the responses by using a unique shared secret TSIG key. BIND can be configured to respond only to authenticated transfer requests by using the allow-transfer statement with a key statement, that restricts the transfers to servers that provide a MAC using the named key.

Rationale:

A zone transfer is a popular information disclosure attack as it provides the entire list of resource records for a zone. There should be very few systems such as the slave name servers that should be authorized to perform a zone transfer for your domains. Authentication of transfer requests should not be made using only an IP address, since IP addresses can be spoofed, but rather by using TSIG keys.

Audit:

Perform the following:

- Search all of the included configuration files and zone files for the allow-transfer option.

```
grep -C 1 allow-transfer $CONFIG_FILES $ZONE_FILES
```

- If there are no allow-transfer statements found, then the configuration allows zone transfers, and is not compliant.
- If the only value in the address match list of all the allow-transfer statements is the value "`none`", either with or without quotes, then the configuration is compliant. Examples output is shown below.

```
allow-transfer { none; };
allow-transfer {"none";};
```

- If **all** of the address list values of the allow-transfer statements have the keyword "key" followed by a name, then the configuration is compliant.

```
allow-transfer { key ns1-ns2.cisecurity.org.; key ns2-ns3.cisecurity.org.; };
```

- If the predefined address value of "any" appears in the allow-transfer statement, then the configuration is not compliant. If any of the address list values contains ACL names, IP addresses or network ranges, then the configuration is also not compliant.

```
allow-transfer { any; }
allow-transfer { key ns1-ns2.cisecurity.org.; 10.10.42.56; }
```

- Additionally, it is possible to confirm if a transfer is allowed to an IP address without a key, by performing the following command on the system with the suspected allowed IP address. An error of "Transfer failed" is the expected result. If a list of resource records is returned, then the transfer was allowed without a key, and the configuration is non-compliant.

```
$ dig @ns1.cisecurity.org cisecurity.org axfr
; <<>> DiG 9.9 . . .
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

Remediation:

Generate TSIG keys 256 bits in length, unique for each host-to-host communication. Securely Transfer the keys and configure the keys to be required in all allow-transfer statements.

Default Value:

If the allow-transfer statement is missing, then transfers are allowed to any host.

5.2 Securely Authenticate Dynamic Updates (Scored)

Profile Applicability:

- Authoritative Name Server

Description:

Dynamic updates are used to automate the updating of zones. Dynamic updates are typically used with DHCP; however, updates may include other records. The allow-update option allows deleting or adding any resource records of a zone except the SOA and NS records, and should not be used. Instead the update-policy option allows a more granular policy to be specified so that only specific resource record types and a specific sub-domain may be updated. The update-policy must be securely authenticated with a key identifier, rather than by an IP address. The key identifier may specify a TSIG key, a GSS-TSIG key, or a SIG(0) key.

Rationale:

Allowing other systems to make permanent updates to your zones is of course not allowed by default, and needs to be carefully secured. Consider the power of an attack that could update the zone to direct clients and servers to the malicious server of the attacker's choice. The attack would not be restricted to just HTTP, but every connection and protocol that uses a name and allows weak authentication may be subject to redirection and a variety of man-in-the-middle and protocol downgrade attacks. Therefore, it is important that all dynamic updates be securely authenticated using a cryptographic key, and not rely on an IP address.

Audit:

Perform the following steps:

- Search for the allow-update option in all of the included configuration files, and in the zone files. If any allow-update options are present, other than “none” or “localhost”, as shown below, then the configuration is not compliant.

```
# grep allow-update $CONFIG_FILES $ZONE_FILES
/etc/named.conf: allow-update { none; };
/. . . /data/cisecurity.org: allow-update { “localhost”; };
```

- Search for any update-policy options in all of the zone files. Any update policies found, should not contain any IP addresses, network CIDR notations, or any ACL names that represents an IP addresses. The only entries in the update-policy should be key identifiers or “local” as shown below. All of the following are compliant.

```
# grep update-policy $ZONE_FILES
/. . ./data/internal.org: update-policy { grant ns1-dhcp-update-key name
dyn.internal.org A; };
/. . ./data/cisecurity.local: update-policy { grant dyn_update_key self
office.cisecurity.local A; };
/. . ./data/test.local: update-policy { local; };
```

Remediation:

Perform the following steps for remediation:

- Remove any `allow-update` options from the global options configuration.
- Replace or add `allow-update` options to the zone files to specify a securely generated TSIG or SIG(0) key identifier, along with the appropriate domain or sub-domain, and the appropriate resource record type.

Default Value:

Dynamic updates are not allowed by default.

5.3 Securely Authenticate Update Forwarding (Scored)

Profile Applicability:

- Authoritative Name Server

Description:

A secondary authoritative name server is allowed to accept zone updates on behalf of the primary name server, and forward them to the master name server, where the zone file can be updated. In this case, the authentication of the dynamic updates is configured with the allow-update-forwarding option. The update requests must be securely authenticated with a key identifier, rather than by an IP address. The key identifier may specify a TSIG key, a GSS-TSIG, or a SIG(0) key.

Rationale:

Of course, allowing unauthenticated updates to a zone should not be allowed. It is necessary for the secondary authoritative name server to carefully authenticate the update request before sending it on to the primary name server, to prevent malicious DNS updates be propagated via the secondary server.

Audit:

Search for the allow-update-forwarding option in all of the included configuration files, and in the zone files. If any allow-update-forwarding options are present, then verify that there are no IP addresses or networks used for authentication. Instead a key identifier should be used, or the value “none” may be used to disable dynamic updates. Note that the key identifiers, may reference a TSIG key, GSS-TSIG key or SIG(0) key. Use the grep command below to search for allow-update-forwarding options, and verify that either key identifiers or the value "none" are used in each.

```
# grep allow-update-forwarding $CONFIG_FILES $ZONE_FILES
/etc/named.conf: allow-update-forwarding { none; };
/. . ./dyn.internal.org: allow-update-forwarding { key dhcp-server.internal.org.; };
```

Remediation:

Modify any allow-update-forwarding options to specify a securely generated TSIG or SIG(0) key identifier used by the DHCP server.

Default Value:

Dynamic updates are disabled by default.

6 Information Leakage

Recommendations in this section are intended to limit the disclosure of potentially sensitive information.

6.1 Hide BIND Version String (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

BIND includes a built-in zone, `version.bind` which may be queried to get the version of the name server. The version may be set to a value of `none`, to disable reporting of the version information.

Rationale:

Making detailed BIND version information easy to obtain remotely helps attackers automate and target their attacks. The information is not necessary for the health of the server, and should not be disclosed.

Audit:

Use the dig command shown below to query the chaos class TXT record on `version.bind`. If there is no output from the command, or if a value of "No Info" or "None" is returned then the configuration is compliant.

```
$ dig @ns1.cisecurity.org version.bind chaos txt | grep '^version.bind.' | grep TXT
version.bind. 0 CH TXT "No Info"

$ dig @ns2.cisecurity.org version.bind chaos txt | grep '^version.bind.' | grep TXT
$
```

Remediation:

Add or modify the version option to have a value of `none` in the BIND global options, as shown below.

```
options {
version none;
. . .
}
```

Default Value:

Default value returns the current BIND detailed version.

6.2 Hide Nameserver ID (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The `server-id` option provides a server identifier that will be returned in response to an NSID query. An NSID query is described in RFC-5001, and is a method to identify servers in an environment where there are multiple DNS servers sharing the same IP address. With the use of load balancing and other IP sharing mechanisms, it can become difficult to discern exactly which name server is responding to a particular query. NSID allows a name server to respond with identifying information. The `server-id` option should be disabled with a value of none.

Rationale:

Enabling the NSID option may allow external parties to obtain information about the configuration and architecture of the DNS server. If it is found to be necessary to enable this service, then the identifying information should be generic. You should not use the server's geographic location, internal IP address or any other privileged information.

Audit:

Use the dig command below to send an NSID query, on the built-in zone `id.server`, for a chaos class TXT record. There should not be any output for a compliant configuration.

```
$ dig @ns2.cisecurity.local id.server chaos txt | grep '^id.server.' | grep TXT
```

An example of a non-compliant response to an NSID query is shown below.

```
$ dig @ns1.cisecurity.local id.server chaos txt | grep '^id.server.' | grep TXT
id.server.          0       CH      TXT     "cpe-172.lima.ny.us.local"
```

Remediation:

To explicitly disable NSID support, add or modify the `server-id` option in the global BIND options with a value of `none` as shown below.

```
server-id none;
```

Default Value:

NSID is disabled by default.

References:

1. https://tools.ietf.org/html/rfc5001

7 Secure Network Communications

Recommendations in this section pertain to the configuration of secure communications to ensure the authenticity and integrity of DNS related network traffic.

7.1 Do Not Define a Static Source Port (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

BIND can be configured to always use the same source port when communicating with other DNS servers. This capability is made possible through the query-source port option, and the query-source-v6 port option. It is recommended that the source port be omitted if the query-source option is used, or that the port be specified as a "*", so that the port will not be a static port number.

Rationale:

DNS attacks which involve spoofing a bogus DNS reply may require the attacker to guess the source port number of the request, if the attacker is unable to see the initial DNS query. Making the source port static makes the attack easier, as it eliminates the effort of getting the correct destination port number for the spoofed reply. Instead of a static source port, the port number should be selected randomly amount the client ephemeral ports.

Audit:

Verify that a static port is not specified in a query-source option or a query-source-v6 option, using the command below.

```
$ egrep '^\w*query-source' $CONFIG_FILES | grep port
/etc/named.conf:    query-source address 10.1.45.53 port *;
```

If there is no output from the grep or if the port number is specified as *, then the configuration is compliant. Examples of a non-compliant configurations are shown below.

```
query-source port 1053;

query-source port-v6 53;
```

Remediation:

Either remove the port specification from the query-source or the query-source-v6 option or use an * for the port number.

Default Value:

The default is to not use a static source port for queries.

7.2 Enable DNSSEC Validation (Scored)

Profile Applicability:

- Caching Only Name Server

Description:

DNS Security Extensions or DNSSEC for short provides authentication of the name servers through public key cryptography. With DNSSEC, the name server signs its responses with its private key. This allows other name servers that have the public key of the name server to verify the integrity and authenticity of the response. DNSSEC also provides for signing of public keys so that delegated sub-domains may have their keys signed by a higher-level authority. This creates a chain of trust so that any name server that trusts the public key of the higher level signing authority can trust the signed key. It is recommended that DNSSEC be enabled and be configured to validate domains that are signed. DNSSEC and validation are enabled via the options `dnssec-enable` and `dnssec-validation`, respectively.

Rationale:

DNSSEC reliably authenticates DNS responses to prevent the DNS spoofing and cache poisoning attacks.

Audit:

Perform the following to verify compliance.

- To verify that the name server will validate the trust for DNSEC signed domains, perform the following dig command on the name server. The command queries the name www.isc.org, which has a valid trusted DNSSEC signature.

```
$ dig @127.0.0.1 www.isc.org. A +dnssec +multiline | grep 'flags:'
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 13
; EDNS: version: 0, flags: do; udp: 4096
```

The compliant result should have the Authenticated Data (`ad`) flag in the header, and the DNSSEC OK (`do`) flag indicating the recursive server is DNSSEC aware.

- To verify that the name server will properly reject DNSSEC signed domains with an invalid signature, perform the dig command below on the www.dnssec-failed.org name.

```
$  dig @127.0.0.1 www.dnssec-failed.org. A | egrep 'status:|flags:'
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 37402
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
; EDNS: version: 0, flags:; udp: 4096
```

The status value should be `SERVFAIL` with `ANSWER: 0`.

Remediation:

Perform the following for remediation:

- Check the BIND configuration files, and in the global options set the two options `dnssec-enable` and `dnssec-validation` to yes as shown below:

```
dnssec-enable yes
dnssec-validation yes
```

- Restart the named server.

Default Value:

DNSSEC and DNSSEC validation are enabled by default.

References:

1. https://users.isc.org/~jreed/dnssec-guide/dnssec-guide.html#easy-start-guide-for-recursive-servers

7.3 Disable the dnssec-accept-expired Option (Scored)

Profile Applicability:

- Caching Only Name Server

Description:

The `dnssec-accept-expired` option allows BIND to accept expired signatures during validation. The option should be disabled so that expired signatures will not be accepted.

Rationale:

Allowing expired signatures would leave the server vulnerable to replay attacks.

Audit:

Verify the `dnssec-accept-expired` option is not present in the configuration files, or is set to a value of "`no`".

```
# grep dnssec-accept-expired $INCLUDE_FILES
/var/named/chroot/etc/named.conf:    dnssec-accept-expired no;
```

Remediation:

Change the `dnssec-accept-expired` option to have a value of "`no`", or remove the option from the configuration files.

Default Value:

The `dnssec-accept-expired` option is disabled by default.

8 Operations - Logging, Monitoring and Maintenance

This section provides recommendations for the BIND server configurations related to operations, updates, logging and monitoring.

8.1 Apply Applicable Updates (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

Over time, patches will be released to resolve defects in BIND. It is recommended that such patches be applied soon after they are available based on risk. High risk vulnerabilities should be patched within 30 days of availability.

Rationale:

By ensuring that BIND remains current and patched, the probability of an attacker successfully compromising BIND is reduced.

Audit:

Verify that the latest patch for the platform version of BIND is installed within 30 days of being available. Use the `-v` option to `named` to get the details version information.

```
$ /sbin/named -v
BIND 9.9.4-RedHat-9.9.4-29.el7_2.4 (Extended Support Version)
```

Remediation:

Update BIND to the most current revision available. Institute a patch process that aims to apply security updates within 30 days of their release. Subscribe to bind-announce@lists.isc.org on the https://www.isc.org web site to receive notifications of available BIND updates.

Default Value:

Not Applicable

References:

1. https://www.isc.org/

8.2 Configure a Logging File Channel (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

To capture logs to a local file, setup a channel for the file, in the logging configuration section. It's often helpful to have one log file for security related logs, and a second one with a dynamic severity level to be used as needed for debugging.

Rationale:

Logging security related events is critical for monitoring the security of the server in order to see any issues affecting the server, and to be able to respond to attacks.

Audit:

Perform the following:

- Search the logging options of the configuration file for configured log files

```
# grep -C 4 channel $CONFIG_FILES | egrep '^\s+file\s+\"'
 file "/var/log/named.log" versions 10 size 20m;
 file "/var/log/secure.log" versions 10 size 20m;
```

- Perform a security related event such as a denied zone transfer to generate a log entry.

```
dig @ns2.cisecurity.org cisecurity.org axfr
```

- Check the log file to verify the attempt was logged.

```
tail /var/log/secure.log
30-Sep-2016 08:54:58.664 client 10.11.214.113#38401 (cisecurity.org): zone transfer
'cisecurity.org/AXFR/IN' denied
```

Remediation:

In `named.conf`, configure a channel for a local security log file with the categories `config`, `dnssec`, `network`, `security`, `updates`, `xfer-in` and `xfer-out`. The local log file will be within the `chroot` directory.

```
logging {
. . .
    channel local_security_log {
        file "/var/run/named/secure.log" versions 10 size 20m;
        severity debug;
        print-time yes;
    };
    // Config file processing
    category config { local_security_log; };
    // Processing signed responses
    category dnssec { local_security_log; };
    // Network Operations
    category network { local_security_log; };
    // Approved or unapproved requests
    category security { local_security_log; };
    // dynamic updates
    category update { local_security_log; };
    // transfers to the name server
    category xfer-in { local_security_log; };
    // transfers from the name server
    category xfer-out { local_security_log; };
    // Optional debug log file, may be enabled dynamically.
    channel local_debug_log {
        file "/var/run/named/debug.log";
        severity dynamic;
        print-time yes;
    };
    category default { local_debug_log; };
    category general { local_debug_log; };
};
```

Default Value:

There is no security log by default.

8.3 Configure a Logging syslog Channel (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

The `syslog` option of the logging configuration allows specification of the syslog facility to send log events. A syslog channel should be configured with the value of `daemon` or other appropriate syslog facility. The `default` and `general` categories should be included and the severity level should be info or lower.

Rationale:

Configuring a syslog channel allows BIND to log important information via the standard system syslog facility. It is important that the BIND logs be included with the system monitoring and response that is performed on other system logs, and the syslog facility is helpful to ensure that the important log information isn't lost, or ignored.

Audit:

Search the configuration file for a syslog logging channel, as shown below.

```
# grep -C 3 channel /etc/named.conf | egrep '^\s+syslog\s+'
                syslog daemon;          # send to syslog's daemon facility
```

Usage of the syslog facility daemon is common practice, but other facilities may be configured.

Remediation:

Configure a syslog channel to capture at least the default and general categories of log events. For external authoritative name servers, the category `lame-servers` may be redirect to null, so that it is not logged. Using lame name servers is common for the domains used for SPAM and may overload the log with information that is not very useful.

```
logging {
. . .
        // Syslog
        channel default_syslog {
                syslog daemon;       # send to syslog's daemon facility
                severity info;       # only send priority info and higher
        };

        category default { default_syslog; };
        category general { default_syslog; };
        // Too many lame servers, especially from SPAM
        category lame-servers { null; };
```

Default Value:

There is no syslog channel by default.

8.4 Disable the HTTP Statistics Server (Scored)

Profile Applicability:

- Authoritative Name Server
- Caching Only Name Server

Description:

Starting in BIND 9.5.0 there was a new statistics web server included, that is a useful debugging tool in a non-production environment. The HTTP server provides data in XML format about the condition of a BIND 9 server. The statistics server provides the same statistics that are available to the statistics-file dump. This server should be left disabled.

Rationale:

A production name server should not have additional, unnecessary services running, as the additional services increases the risk of vulnerabilities.

Audit:

Verify that there is NOT a statistics channel statement:

```
# grep statistics-channel $CONFIG_FILES
```

No output is expected and confirms that the HTTP service is not enabled.

Remediation:

Remove the `statistics-channel` option from the configuration file.

Default Value:

The HTTP server is disabled by default.

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Planning and Architecture**		
1.1	Use a Split-Horizon Architecture (Not Scored)	☐	☐
1.2	Do Not Install a Multi-Use System (Not Scored)	☐	☐
1.3	Dedicated Name Server Role (Scored)	☐	☐
1.4	Use Secure Upstream Caching DNS Servers (Not Scored)	☐	☐
1.5	Installing ISC BIND 9 (Scored)	☐	☐
2	**Restricting Permissions and Ownership**		
2.1	Run BIND as a non-root User (Scored)	☐	☐
2.2	Give the BIND User Account an Invalid Shell (Scored)	☐	☐
2.3	Lock the BIND User Account (Scored)	☐	☐
2.4	Set root Ownership of BIND Directories (Scored)	☐	☐
2.5	Set root Ownership of BIND Configuration Files (Scored)	☐	☐
2.6	Set Group named or root for BIND Directories and Files (Scored)	☐	☐
2.7	Set Group and Other Permissions Read-Only for BIND Non-Runtime Directories (Scored)	☐	☐
2.8	Set Group and Other Permissions Read-Only for All BIND Files (Scored)	☐	☐
2.9	Isolate BIND with chroot'ed Subdirectory (Scored)	☐	☐
3	**Restricting Queries**		
3.1	Ignore Erroneous or Unwanted Queries (Scored)	☐	☐
3.2	Restrict Recursive Queries (Scored)	☐	☐
3.3	Restrict Query Origins (Not Scored)	☐	☐
3.4	Restrict Queries of the Cache (Scored)	☐	☐
4	**Transaction Signatures -- TSIG**		
4.1	Use TSIG Keys 256 Bits in Length (Scored)	☐	☐
4.2	Include Cryptographic Key Files (Scored)	☐	☐
4.3	Use Unique Keys for Each Pair of Hosts (Scored)	☐	☐
4.4	Restrict Access to All Key Files (Scored)	☐	☐
4.5	Protect TSIG Key Files During Deployment (Not Scored)	☐	☐
5	**Authenticate Zone Transfers and Updates**		
5.1	Securely Authenticate Zone Transfers (Scored)	☐	☐
5.2	Securely Authenticate Dynamic Updates (Scored)	☐	☐
5.3	Securely Authenticate Update Forwarding (Scored)	☐	☐
6	**Information Leakage**		
6.1	Hide BIND Version String (Scored)	☐	☐
6.2	Hide Nameserver ID (Scored)	☐	☐

7	**Secure Network Communications**		
7.1	Do Not Define a Static Source Port (Scored)	☐	☐
7.2	Enable DNSSEC Validation (Scored)	☐	☐
7.3	Disable the dnssec-accept-expired Option (Scored)	☐	☐
8	**Operations - Logging, Monitoring and Maintenance**		
8.1	Apply Applicable Updates (Scored)	☐	☐
8.2	Configure a Logging File Channel (Scored)	☐	☐
8.3	Configure a Logging syslog Channel (Scored)	☐	☐
8.4	Disable the HTTP Statistics Server (Scored)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
January, 2006	1.0	Public Release
May 4th, 2009	2.0.0	Updated all previous information with updated security practices. Included several new technologies included in updates up to BIND 9.5.0-P2, including DHCID, NSID, and the HTTP statistics server.
November 30th, 2016	3.0.0	Ticket #1: Updated to recommend using a variable or `/etc/named.conf` in top level configuration files and recommended symbolic links for chroot'ed installations.
November 30th, 2016	3.0.0	Ticket #2: Updated Audit and Remediation discussions to follow current CIS style guidelines.
November 30th, 2016	3.0.0	Ticket #3: Updated benchmark to a version of BIND which supports the HMAC-SHA256 algorithm.
November 30th, 2016	3.0.0	Ticket #4: Corrected grammar error in the Rationale Statement for recommendation 2.1.
November 30th, 2016	3.0.0	Ticket #5: Corrected grammar error in the Remediation Procedure for recommendation 1.5
December 21st, 2016	3.0.0	Major rewrite and reorganization.

December 21st, 2016	3.0.0	Audit steps were added for most of the rules, previously many were labeled as "not applicable".
December 21st, 2016	3.0.0	Profiles renamed for the name sever roles, and simplified the roles to just Authoritative and Caching Only.
December 21st, 2016	3.0.0	Removed the Solaris 10 specific recommendations, remediations and audits.
December 21st, 2016	3.0.0	Simplified the chroot restrictions and removed SELinux as an alternative.
December 21st, 2016	3.0.0	The chroot restrictions were simplified and SELinux was not included as an alternative, as it's not commonly used for BIND.
December 21st, 2016	3.0.0	Divided directory and file permissions requirements into five separate requirements and reworked to require specific auditable controls.
December 21st, 2016	3.0.0	Removed or combined multiple recommendations
December 21st, 2016	3.0.0	Planned Update