

## Configure a BIND DNS Server

Created September 7, 2006  
by Bruce A. Westbrook

Revisions: 12-21-06 BAW – Sanitized for public distribution

### Introduction

This document describes the step-by-step process of installing and configuring a BIND nameserver on Fedora Core 5 (or Red Hat Enterprise) for the purposes of having a DMZ DNS server that holds only the records needed for the DMZ. Additionally, this server will serve as the forwarder for our internal DNS, enabling us to restrict or redirect access to known “evil” sites, such as adware, spyware, etc.

For the examples in this document, the following will be used. Change these as necessary to your own network:

DMZ	10.0.0.0 /24
Internal LAN	192.168.1.0 /24
Internal DNS Domain	mydomain.local
BIND Server	ns.mydomain.local @ 10.0.0.1

### Install BIND

√	Description
<b>Download &amp; Install BIND</b>	Use YUM to download and install BIND: <code>yum -y install bind</code>  OR  Download and install the RPM directly from:  <code>wget http://download.fedora.redhat.com/pub/fedora/linux/core/5/source/SRPMS/bind-9.3.2-4.1.src.rpm</code>  <code>rpm -ivh bind-9.3.2-4.1.src.rpm</code>
<b>Start BIND</b>	Turn on the service at startup: <code>chkconfig named on</code>  Start the service now: <code>service named start</code>

## Configure BIND as a Regular Nameserver

√	Description
	<p><b>Configure resolv.conf</b></p> <p>Linux DNS clients use the <code>/etc/resolv.conf</code> file to determine both the location of their DNS server and the domains to which they belong. The file generally has two columns; the first contains a keyword, and the second contains the desired values separated by commas.</p> <p>You'll have to make your DNS server refer to itself for all DNS queries by configuring the <code>/etc/resolv.conf</code> file to reference localhost only.</p> <pre>vim /etc/resolv.conf</pre> <p>Remove or remark out any information in the file and add only the following two lines, substituting <code>localdomain.tld</code> with your internal DNS domain:</p> <pre>search mydomain.local nameserver 127.0.0.1</pre>
	<p><b>Configure named.conf</b></p> <p>The main BIND configuration file is located in <code>/etc/named.conf</code> which contains the the main DNS configuration and tells BIND where to find the configuration files for each domain you own (or configure).</p> <p>Looking through the file you will see three different statements. First, there is the <code>options</code> statement, which contains directory and files paths. Next there is a <code>controls</code> statement, which configures various settings of the BIND server. Finally, there are multiple <code>zone</code> statements. These zone statements are what tells the BIND server what domains it has information for.</p> <p>First, let's put in some <b>security settings</b>.</p> <p>Since this will be our only DNS server let's disable zone transfers. In the <code>options</code> statement, add the following line to what's already there:</p> <pre>allow-transfer { none; };</pre> <p>Change the display version of our BIND server to potential attackers by adding this line to the <code>options</code> statement as well:</p> <pre>version "Not Available";</pre> <p>Since we are using this DNS server for only our internal and DMZ networks, we will restrict access to only our known networks. Add this line to the <code>options</code> statement – add, delete or change the ranges as necessary:</p> <pre>allow-query { 127.0.0.1; 10.0.0.0/8; 172.16.0.0/20; 192.168.0.0/16; };</pre> <p>Let's also configure our forwarding servers. The goal here is to use our ISP DNS servers to get DNS information as much as possible since they're heavily used and will have many DNS records cached. Doing this will typically be faster then going to the DNS root servers and performing the lookups ourselves. Add this line to the <code>options</code> statement, where the IP addresses shown would be replaced with your own ISP DNS servers:</p> <pre>forwarders { 4.2.2.2; 216.136.95.2; };</pre>

Now that security and DNS forwarding is setup, let's configure our own domains or **zone statements**:

Place the following zone statements right before the current zone statements – again replacing the domain names and IP addressing as needed for your own network:

```
zone "mydomain.local" IN {  
    type master;  
    file "mydomain.local.db";  
};  
  
zone "1.168.192.IN-ADDR.ARPA" IN {  
    type master;  
    file "192.168.1.db";  
};  
  
zone "0.0.10.IN-ADDR.ARPA" IN {  
    type master;  
    file "10.0.0.db";  
};
```

Save and close the file.

<p><b>Configure Forward Lookup Zone File</b></p>	<p>The zone files are located in <code>/var/named/</code>. Here we will first create the forward lookup zone file for <code>mydomain.local</code>. Let's assume that the name server's FQDN is <code>ns01.mydomain.local.org</code> and the IP address is <code>10.0.0.1</code> for these examples. Replace this name as needed.</p> <pre>cd /var/named vim mydomain.local.db</pre> <p>Insert the following information into this new file:</p> <pre>;Time To Live \$TTL 24h ;SOA @ IN SOA ns01.mydomain.local. sysadmin.mydomain.local. (     2007090701 ;Serial Number - YYYYMMDD + #     3h ;Refresh after 3 hours     1h ;Retry after 1 hour     1w ;Expire after 1 week     1h ) ;Negative caching TTL of 1 day ;Name Servers @ IN NS ns01 ;Address Records ns01 IN A 10.0.0.1 file01 IN A 192.168.1.11 file02 IN A 192.168.1.12 mail IN A 10.0.0.5 www IN A 10.0.0.6 etc...</pre> <p>Save and close the file once all your A records are added.</p>
<p><b>Create Reverse Lookup Zone File for 192.168.1.x</b></p>	<p>Now let's configure the reverse lookup file:</p> <pre>vim 192.168.1.db</pre> <p>Insert the following information into this new file:</p> <pre>;Time To Live \$TTL 24h ;SOA @ IN SOA ns01.mydomain.local. sysadmin.mydomain.local. (     2007090701 ;Serial Number - YYYYMMDD + #     3h ;Refresh after 3 hours     1h ;Retry after 1 hour     1w ;Expire after 1 week     1h ) ;Negative caching TTL of 1 day ;Name Servers @ IN NS ns01.mydomain.local. ;Pointer Records 11 IN PTR file01.mydomain.local. 12 IN PTR file02.mydomain.local. etc...</pre>

<p><b>Create Reverse Lookup Zone File</b></p> <p>for</p> <p><b>10.0.0.x</b></p>	<pre>vim 10.0.0.db</pre> <p>Insert the following information into this new file:</p> <pre>;Time To Live \$TTL 24h ;SOA @ IN SOA ns01.mydomain.local. sysadmin.mydomain.local. (     200709071 ;Serial Number - YYYYMMDD + #     3h ;Refresh after 3 hours     1h ;Retry after 1 hour     1w ;Expire after 1 week     1h ) ;Negative caching TTL of 1 day ;Name Servers @ IN NS ns01.mydomain.local. ;Pointer Records 1 IN PTR ns01.mydomain.local. 5 IN PTR mail.mydomain.local. 6 IN PTR www.mydomain.local.</pre>
<p><b>Chroot Jail</b></p> <p>- OPTIONAL -</p>	<p>If you're using these instructions to run BIND on Red Hat Enterprise Linux, and you install BIND from the Red Hat CDs, BIND will be installed in a chroot jail. As such, you will simply need to move your files to the chroot jail:</p> <pre>cd /var/named mv mydomain.local.db /var/named/chroot/var/named/ mv 192.168.1.db /var/named/chroot/var/named/ mv 10.0.0.db /var/named/chroot/var/named/</pre>
<p><b>Change Ownership</b></p>	<p>Now change the ownership of these files from root to named:</p> <pre>cd /var/named/chroot/var/named chown named:named mydomain.local.db chown named:named 192.168.1.db chown named:named 10.0.0.db</pre>
<p><b>Configure iptables</b></p>	<p>If you want anyone to make requests to your DNS server, you'll need to allow port 53 through iptables. Make sure you open the port, similar to this:</p> <pre>iptables -D INPUT -j DROP iptables -A INPUT -p tcp --dport 53 -j ACCEPT iptables -A INPUT -p udp --dport 53 -j ACCEPT iptables -A INPUT -j DROP</pre> <pre>service iptables save service iptables restart iptables -nL</pre>
<p><b>Restart BIND</b></p>	<p>You must restart BIND anytime you make changes to any of the configuration files, as they are read at startup and then cached in memory.</p> <pre>service named restart</pre>

## Malware Prevention through Blackhole DNS

√	Description
	<p data-bbox="289 300 440 321"><b>Explanation</b></p> <p data-bbox="488 300 1412 506">One of the more popular techniques for fighting malware among home users is through the use of a host file for DNS redirection. A host file can be used to map hostnames associated with malware to a different IP address (such as a loopback address, 127.0.0.1). This will prevent connections to those malicious sites from ever taking place. (There is an irony here, as some of the more "evil" malware hijacks your host file to prevent their removal or to redirect search queries).</p> <p data-bbox="488 541 1398 600">Unfortunately, there are several problems with using host files, especially in a corporate environment:</p> <ul data-bbox="537 636 1412 1276" style="list-style-type: none"> <li>• There needs to be an exact entry in the hosts file for each malware host/domain combination. Even within the same domain, there must be separate host entry. For example, there must be individual entries for www1.malwaresite.com, www2.malwaresite.com, www3.malwaresite.com, www4.malwaresite.com, etc. A host not listed (such as www5.malwaresite.com) will continue to be resolved to the actual malware-associated site. (This is a major problem with pattern-matching spam filters, and will become even more of a problem as "wildcard dns" continues to increase in popularity.). Major malware players may have dozens of hosts per domain. This causes the host file to become quite large.</li> <li>• While a small hosts file will speed up browsing (since anything listed in the file are not downloaded to the desktop) some issues have been reported with very large host files.</li> <li>• Updating multiple host files on a corporate LAN can quickly become unwieldy. Because there is no centralized administration point, a single change must be pushed to every client. (If the host file is copied via a login script, the sysadmin would have to ensure that user log into the domain on a daily basis to receive the latest file). Also, copying host files via a login script may be problematic if the user does not have permissions to overwrite their local host file.</li> </ul> <p data-bbox="488 1312 1377 1398">As an alternative to host files, we will configure our Internet forwarding DNS server (the DMZ DNS server) to be the primary master DNS server for domains associated with malware and spyware.</p> <p data-bbox="488 1434 1412 1549">The DNS server, believing it is an "authority" for the malware zone, will answer the query instead of querying an Internet DNS server for the answer. The desktop receiving the answer doesn't know that the IP address received is not the malware IP.</p> <p data-bbox="488 1585 1409 1734">We will configure the resolved IP address for these malware sites to be a local internal web server that is configured to "answer" all image requests mapped to a single pixel image file, and all text pages mapped to a warning message page. This configuration has the added benefit of generating log files for inspection, providing us with a list of potential problem PCs on the network.</p> <p data-bbox="488 1770 1412 1829">An internal DNS server configured to answer for malware domains has several advantages over a host file, including:</p> <ul data-bbox="537 1864 1328 1915" style="list-style-type: none"> <li>• No slowdown due to large number of domains and host entries.</li> <li>• Centralized control and administration point for configuration</li> </ul>

		<p>modifications. Any additions or modifications are immediately available to all users.</p> <ul style="list-style-type: none"> <li>• No need to continue to add individual host entries when a malware site modifies the hostname portion of the domain. If the domain is configured for "Wildcard DNS", then it will answer for any and all host requests with a single domain. Host file entries for <code>www1.malwaresite.com</code>, <code>www2.malwaresite.com</code>, <code>www3.malwaresite.com</code>, <code>www4.malwaresite.com</code> are now replaced by a single domain entry <code>malwaresite.com</code>. This eliminates having to list multiple hosts within the same domain in a local hosts file.</li> <li>• OS independent: the local DNS server will work for all clients (XP, W2K, Linux, MacOS, etc...).</li> </ul> <p>We will initially configure our server to use the list "black-hole" DNS zone file from Bleeding Snort. There are over 10,000 malware domains included in this file!</p> <p>The file(s) can be downloaded from: <a href="http://www.bleedingsnort.com/blackhole-dns/files/">http://www.bleedingsnort.com/blackhole-dns/files/</a></p>
	<p><b>Download and Configure</b></p>	<p>To configure our DNS server to block these malware sites, we need to do two things. First, add all of the domains as zones into our <code>named.conf</code> configuration file. Second, create a zone file to point to wherever we want the sites to resolve to.</p> <p>Rather than adding all the spyware domains into our <code>named.conf</code>, we'll take advantage of the <code>named.conf</code> file's ability to include other files into the config. This will allow us to easily maintain the spyware domain file seperately. We'll also create a directory that will specifically house the spyware configuration files for ease of administration.</p> <p><b>IMPORTANT CHROOT NOTE:</b> If you are running BIND in a CHROOT jail, you will need to change <u>SOME</u> of the directory paths in these instructions to your chroot jail. I will indicate where. For the examples I will use the RHEL default chroot in <code>/var/named/chroot/var/named/</code></p> <ol style="list-style-type: none"> <li>1. Change to your BIND files directory:        Normally: <code>cd /var/named/malware</code>        Chroot Jail: <code>cd /var/named/chroot/var/named</code> </li> <li>2. Create the directory <code>./malware</code>:  <code>mkdir ./malware</code> </li> <li>3. Change to this new directory:  <code>cd ./malware</code> </li> <li>4. Download the following files into your new malware directory:  <code>wget http://www.bleedingsnort.com/blackhole-dns/files/spywaredomains.zones</code> </li> <li>5. Run the following single command (it's line-wrapped here) to properly configure the path in the <code>spywaredomains.zones</code> file for our server (this is the <u>same</u> whether it's in a chroot jail or not):  <code>sed -e 's#/etc/namedb#./malware#g' ./spywaredomains.zones &gt; ./spyware.zones</code> </li> <li>6. Delete the original <code>spywaredomains.zones</code> file:  <code>rm spywaredomains.zones</code> </li> <li>7. Create a forward lookup zone file in our malware directory for our blocked domains, as follows:  <code>vim blockeddomain.hosts</code> </li> </ol>

File contents:

```
$TTL 24h
@ IN SOA ns01.mydomain.local. null.mydomain.local. (
    1
    8h
    2h
    1w
    1h )
@ IN NS ns01.mydomain.local.
* IN A 127.0.0.1
```

8. In the file contents above, you can change the 127.0.0.1 to point to a webserver instead that will respond with a "Spyware domain blocked" type index page.
9. Edit the `/etc/named.conf` file and add the following `include` statement at the very END of the file (this is the same whether it's in a chroot jail or not):  
`include "/var/named/malware/spyware.zones";`
10. Restart the BIND service – note that this could take a couple of minutes as the service loads 10,000+ domains into memory:  
`service named restart`
11. Test the DNS response by pinging one of the domains from the `spyware.zones` file.

## Maintaining the DNS Server

√	Description
	<p data-bbox="282 296 444 352"><b>Update Host Records</b></p> <p data-bbox="488 296 1406 386">After the BIND server has been configured and is operational, it's quite simple to keep it updated with DNS information. Just keep in mind that the directory the files are located in depend on whether it's in a chroot jail or not.</p> <p data-bbox="488 417 760 447"><b><u>To add a new record:</u></b></p> <ol data-bbox="537 451 1406 995" style="list-style-type: none"> <li>1. Edit the <code>mydomain.local.db</code> file  <pre data-bbox="683 480 1208 506">vim ./var/named/mydomain.local.db</pre> </li> <li>2. Increment the serial number at the top of the file – update to current date + change number of the day. 2007100201 = 2007 Oct 2, First change of the day</li> <li>3. Go to the bottom of the file and add the record in the following format:  <pre data-bbox="683 636 1127 661">host_name IN A IP_address</pre> </li> <li>4. Save and close the file</li> <li>5. Edit the <code>ip_range.db</code>  <pre data-bbox="683 730 1159 756">vim ./var/named/xxx.yyy.zzz.db</pre> </li> <li>6. Increment the serial number</li> <li>7. Go to the bottom of the file and add the record in the following format:  <pre data-bbox="683 825 1062 850">last_octet IN A FQDN.</pre> </li> <li>8. Save and close the file</li> <li>9. Reload the zone files:  <pre data-bbox="683 913 1224 995">rndc reload mydomain.local. rndc reload 1.168.192.IN-ADDR.ARPA rndc reload 0.0.10.IN-ADDR.ARPA</pre> </li> </ol> <p data-bbox="488 1031 732 1060"><b><u>To delete a record:</u></b></p> <ol data-bbox="537 1064 1208 1482" style="list-style-type: none"> <li>1. Edit the <code>mydomain.local.db</code> file  <pre data-bbox="683 1094 1208 1119">vim ./var/named/mydomain.local.db</pre> </li> <li>2. Increment the serial number</li> <li>3. Find the record (line) you want to delete and delete it</li> <li>4. Save and close the file</li> <li>5. Edit the <code>ip_range.db</code>  <pre data-bbox="683 1251 1159 1276">vim ./var/named/xxx.yyy.zzz.db</pre> </li> <li>6. Increment the serial number</li> <li>7. Find the record (line) you want to delete and delete it</li> <li>8. Save and close the file</li> <li>9. Reload the zone files:  <pre data-bbox="683 1402 1224 1484">rndc reload mydomain.local. rndc reload 1.168.192.IN-ADDR.ARPA rndc reload 0.0.10.IN-ADDR.ARPA</pre> </li> </ol> <p data-bbox="488 1520 699 1549"><b><u>To edit a record:</u></b></p> <ol data-bbox="537 1554 1406 1915" style="list-style-type: none"> <li>1. Edit the <code>mydomain.local.db</code> file  <pre data-bbox="683 1583 1208 1608">vim ./var/named/mydomain.local.db</pre> </li> <li>2. Increment the serial number</li> <li>3. Go to the record you want to edit and make your changes using the following format:  <pre data-bbox="683 1707 1127 1732">host_name IN A IP_address</pre> </li> <li>4. Save and close the file</li> <li>5. Edit the <code>ip_range.db</code>  <pre data-bbox="683 1801 1159 1827">vim ./var/named/xxx.yyy.zzz.db</pre> </li> <li>6. Increment the serial number</li> <li>7. Find the record (line) you want to delete and edit it using the following format:</li> </ol>

		<pre>last_octet IN A FQDN.</pre> <ol style="list-style-type: none"> <li>Save and close the file</li> <li>Reload the zone files: <pre>rndc reload mydomain.local. rndc reload 1.168.192.IN-ADDR.ARPA rndc reload 0.0.10.IN-ADDR.ARPA</pre> </li> </ol>
	<p><b>Update DNS Forwarding</b></p>	<ol style="list-style-type: none"> <li>Edit the <code>/etc/named.conf</code> file <pre>vim /etc/named.conf</pre> </li> <li>Locate the line near the beginning of the file (in the <code>options</code> statement) that starts with <code>forwarders</code>. It will look something like this: <pre>forwarders { 4.2.2.2; 216.136.95.2; }</pre> </li> <li>Edit the IP addresses as necessary</li> <li>Save and close the file</li> <li>Restart the BIND service: <pre>service named restart</pre> </li> </ol>
	<p><b>Update Spyware Sites</b></p>	<p>To keep the spyware sites updated, we can choose to do this manually or automatically. We should set up the automatic method to ensure we keep the blocklist as updated as possible, but I will review both methods.</p> <p>Again, keep in mind that the directory some of these files are located in depend on whether it's in a chroot jail or not.</p> <p><b>Manual:</b></p> <ol style="list-style-type: none"> <li>Change to our spyware zone directory: <pre>cd ./var/named/malware</pre> </li> <li>Make a backup copy of our current spyware.zones file: <pre>cp spyware.zones spyware.zone.bak</pre> </li> <li>Download the latest following file into your malware directory: <pre>wget http://www.bleedingsnort.com/blackhole-dns/files/spywaredomains.zones</pre> </li> <li>Run the following single command (it's line-wrapped here) to properly configure the path in the spywaredomains.zones file for our server. This will overwrite our current spyware.zones file (thus the earlier backup): <pre>sed -e 's#/etc/namedb#./malware#g' ./spywaredomains.zones &gt; ./spyware.zones</pre> </li> <li>Delete the original spywaredomains.zones file: <pre>rm spywaredomains.zones</pre> </li> <li>Restart the BIND service: <pre>service named restart</pre> </li> <li>Test name resolution using ping, host or whatever</li> </ol>

**Automatic:**

1. Change to our spyware zone directory:  
`cd ./var/named/malware`
2. Create the `update.sh` script found at the end of this document.
3. Edit the file and change the email address line:  
`admin=sysadmin@mydomain.local`
4. Also be sure to change the malware directory if this is chroot'd:  
`malware=/var/named/chroot/var/named/malware`
5. Save and close the file
6. DOS to Unix the file (to remove Windows characters from copying the text):  
`dos2unix update.sh`
7. Add executable properties to the file and remove world writeable:  
`chmod 755 update.sh`
8. Cron the script sometime overnight:  
`vim /etc/cron.daily/dns-blacklist.cron`

File contents:

`/var/named/malware/update.sh`

OR

`/var/named/chroot/var/named/malware/update.sh`

9. Set permissions on the cron file:  
`chmod 700 /etc/cron.daily/dns-blacklist.cron`
10. Edit the sendmail client configuration file to point to our mail servers (for sending messages from the update script):  
`vim /etc/mail/submit.cf`

**Search for the following line:**

`# "Smart" relay host (may be null)`  
`DS`

Change DS to `DSmail.mydomain.local`

## Detailed Explanations

√	Description	
	<p><b>Explanation of Named.conf</b></p>	<p>This is an explanation of the syntax used in our named.conf file.</p> <p><b>zone</b> = the domain name we're going to be authoritative for, with the class type. In almost all cases the class will always be <b>IN</b> for Internet.</p> <p><b>type</b> = the type of DNS server we are for the domain. In our case we're the primary master</p> <p><b>file</b> = the zone file that holds this domains records</p> <p>There are additional parameters that can be used in the named.conf file but we're keeping things pretty simple here.</p>
	<p><b>Explanation of Forward Lookup Zone File</b></p>	<p>This is an explanation of the syntax used in our forward lookup zone file.</p> <p><b>\$TTL</b> = This is the zone's "Time to Live" (TTL) value. The purpose of a TTL is to reduce the number of DNS queries the authoritative DNS server has to answer. If the TTL is set to three days, then caching servers use the original stored response for three days before making the query again.</p> <p><b>SOA</b> = "Start of Authority" (SOA) Resource Record. This is the first record in the zone file and contains general administrative and control information about the domain (or zone). It has the format:</p> <p style="text-align: center;"><b>Name,Class,Type,Name-Server,Email,Serial#,Refresh,Retry,Expiry,Min-TTL</b></p> <p>For the sake of formatting, you can insert new line characters between the fields as long as you insert parenthesis at the beginning and end of the insertion to alert BIND that part of the record will straddle multiple lines. You can also add comments to the end of each new line separated by a semicolon when you do this. That is exactly how we formatted our SOA record.</p> <p>Here is an explanation of each of the individual fields in the SOA:</p> <p><b>Name</b> = The root name of the zone. The "@" sign is a shorthand reference to the current origin (zone) in the <code>/etc/named.conf</code> file for that particular database file.</p> <p><b>Class</b> = There are a number of different DNS classes. Almost always we will be using the <b>IN</b> or Internet class used when defining IP address mapping information for BIND. Other classes exist for non Internet protocols and functions but are very rarely used.</p> <p><b>Type</b> = The type of DNS resource record. In the example, this is an SOA resource record. Other types of records exist, such as NS, MX, A, CNAME and PTR.</p> <p><b>Name-Server</b> = Fully qualified name of your primary name server. Must be followed by a period. Note that any names in the zone data file that don't end in a dot will have the origin (domain name) appended to the end.</p>

		<p><b>Email</b> = The e-mail address of the name server administrator. The regular @ in the e-mail address must be replaced with a period instead. The e-mail address must also be followed by a period.</p> <p><b>Serial #</b> = A serial number for the current configuration. You can use the date format YYYYMMDD with an incremented single digit number tagged to the end. This will allow you to do multiple edits each day with a serial number that both increments and reflects the date on which the change was made.</p> <p><b>NOTE:</b> The following four parameters can have their time-related values set in several different ways by using a suffix. These suffixes are D for days, W for weeks and H for hours. The suffix is not case sensitive.</p> <p>By default, if there is no suffix, BIND assumes the number is in seconds (where 3600 would equal 60 minutes or 86400 would equal 24 hours, for instance).</p> <p><b>Refresh</b> = Tells any slave (or secondary) DNS servers how often they should check the master DNS server for updates. We'll configure the setting although we're not using any secondary DNS servers at this time.</p> <p><b>Retry</b> = The secondary DNS servers retry interval to connect to the master DNS server in the event of a connection failure.</p> <p><b>Expiry</b> = Total amount of time a secondary DNS server should retry to contact the master before expiring the data it contains. Future references will be directed towards the root servers.</p> <p><b>Min-TTL</b> = There are times when remote clients will make queries for subdomains that don't exist. The DNS server will respond with a <i>no domain</i> or <i>NXDOMAIN</i> response that the remote client caches. This value defines the caching duration your DNS includes in this response.</p> <p><b>NS</b> = Name Server Record. This record provides the IP address or CNAME of the name server. The format is: <b>domain, class, type, host_name</b> The domain and class are both implied as @ and IN if you leave them blank, but you MUST have a blank space at the beginning of the line to do this. To make it easier to see in our configuration we've included all the fields. You can have more than one NS by having additional records (or lines).</p> <p><b>A</b> = "Address" records are the actual forward lookup records for host names. The format is: <b>host_name, class, type, IP_Address</b></p> <p>Two other record types exist that we are not currently using in our forward lookup zone file. They are CNAME and MX. An explanation of each follows:</p> <p><b>CNAME</b> = "Canonical Name" record. This is simply an alias name record that points to an A record name. The question typically posed here is why even use a cname record instead of an A record? I actually could find little information to support the ongoing use of CNAME records. There is some history about sendmail having behavioral problems without CNAME records, and sysadmins using CNAME records as a way to admin systems that are multihomed, but other than that, nothing. My opinion as of this writing is to simply use</p>
--	--	--

		<p>A records for everything.</p> <p><b>MX</b> = "Mail Exchanger" record. Points to the mail server for the domain. The format is:  <b>domain_name, class, type, preference_value, mail_server</b></p> <p>The preference_type tells a mailer which mail server to try first. The number can be any 16-bit number (between 0 and 65535). The number itself is unimportant, it's only the relationship to the values of other mail exchangers that matters – is it higher or lower than the other mail exchanger records?</p> <p>An example MX record for mydomain.local would be:  <pre>mydomain.local. IN MX 10 mail.mydomain.local. mydomain.local. IN MX 50 mail2.mydomain.local.</pre></p> <p>A mailer would try the lowest preference number first (<b>mail.mydomain.local</b>). If it does not respond, it will try the next higher number, and so on.</p> <p>Note that in this case you would also then need to have a corresponding <b>A</b> record for the <b>mail</b> and <b>mail2</b> host names.</p>
	<p><b>Explanation of Reverse Lookup Zone File</b></p>	<p>The reverse lookup zone file is basically the same as the forward lookup zone file, except that we are mapping IP addresses to host names rather than host names to IP addresses.</p> <p>Having reverse lookups is very important when it comes to mail servers as many mail servers relay only from hosts whose IP addresses resolve correctly in DNS. Also, NFS requires valid reverse lookup capabilities.</p> <p>The SOA statement is almost the same as the forward lookup zone.</p> <p>The NS statement is also the same, using the FQDN (Fully Qualified Domain Name).</p> <p>In the reverse lookup zones we use PTR record types. A PTR record is one which points the last octet of the IP address to a FQDN. The format is:</p> <p style="text-align: center;"><b>last_octet, class, type, FQDN</b></p> <p>For example, for <b>file01</b> at <b>192.168.11</b>, the line would be:</p> <pre>11 IN PTR file01.mydomain.local.</pre>

## Troubleshooting

√	Description
	<p><b>Log Files</b></p> <p>By default, log messages are logged to the <code>/var/log/messages</code> file. This will show any errors as well as tell you which zones loaded using which serial numbers.</p> <p>You can use the <code>host</code> command to see if your forward lookup entries work, like this:</p> <pre>host file01</pre> <p>The result should provide the FQDN with the IP address, like this:</p> <pre>file01.mydomain.local has address 192.168.1.11</pre> <p>If the lookup fails, you will get something like this:</p> <pre>Host file01 not found: 3 (NXDOMAIN)</pre> <p>You can also make your query specifically to your name server to ensure you are talking to the right DNS – for instance, from the DNS server itself:</p> <pre>host file01 localhost</pre> <p>or from a client to your DNS server with the name or IP address:</p> <pre>host file01 ns01</pre> <p>Your results will look something like this:</p> <pre>Using domain server: Name: ns01 Address: 10.0.0.122#53 Aliases:  file01.mydomain.local has address 192.168.1.11</pre> <p>You can also use the <code>dig</code> command to query the name server for various information.</p> <p>To <code>dig</code> for the SOA:</p> <pre>dig mydomain.local soa</pre> <p>Should return all of the SOA information, including something like this:</p> <pre>... ... ;; ANSWER SECTION: mydomain.local. 86400 IN SOA ns01.mydomain.local. sysadmin.mydomain.local. 2007090901 10800 3600 604800 3600  ;; AUTHORITY SECTION: mydomain.local. 86400 IN NS ns01.mydomain.local.  ;; ADDITIONAL SECTION: ns01.mydomain.local. 86400 IN A 10.0.0.1 ... ...</pre>

<b>Zone Transfer Protection</b>	<p>As we saw the host command does one DNS query at a time, but the dig command is much more powerful. When given the right parameters it can download the entire contents of your domain's zone file.</p> <p>In this example, the AXFR zone transfer parameter is used to get the entire contents of the mydomain.local zone file, like this:</p> <pre>dig mydomain.local AXFR</pre> <p>The results will display the entire zone contents – all the records and information about the domain. While this may not seem like an important security threat at first glance, it actually is. Anyone can use this command to determine all your server's IP addresses and from the names determine what type of server it is and then launch an appropriate cyber attack. It's a lot of information for an attacker to get.</p> <p>To protect against this type of information leak, with a single DNS server and no secondary servers, you can simply turn off zone transfers all together. This is done by applying the <code>allow-transfer</code> directive to the global options section of the <code>named.conf</code> file. We did this in our initial configuration, but specifically you simply add the following line to the <code>options</code> section:</p> <pre>allow-transfer { none; };</pre> <p>Restart the named service and try to dig the entire domain again – it should now fail.</p>
<b>Versioning</b>	<p>If a hacker can easily find out which version of BIND we're running, they can tailor their attacks. And by default, BIND will happily give out its version to anyone that asks. See for yourself – just type the following, using the FQDN or IP address of your server:</p> <pre>dig @ns01 txt chaos version.bind</pre> <p>In our case, given that you followed the configuration setup in these procedures, you should see something like this, where we already changed the version to <code>Not Available</code>:</p> <pre>... ... ;; QUESTION SECTION: version.bind.      CH      TXT  ;; ANSWER SECTION: version.bind.     0      CH      TXT      "Not Available"  ;; AUTHORITY SECTION: version.bind.     0      CH      NS      version.bind. ... ...</pre> <p>To change the version you simply add the <code>versions</code> directive to the options statement in your <code>named.conf</code> file, like this:</p> <pre>version "Not Available";</pre>

<b>RNDC Error: Connection to Remote Host Closed</b>	<pre>rndc: connection to remote host closed</pre> <p>This may indicate that the remote server is using an older version of the command protocol, this host is not authorized to connect, or the key is invalid.</p> <p>If you are getting the above error when trying to check the status of the server, or when you start up the named service, I found that with RHEL it's a key mismatch. Depending on which version of RHEL is installed, and whether it's been updated or not, you may have one of the Red Hat messed up updates.</p> <p>To fix it, try the following:</p> <p>Make a copy of the original key config file: <code>cp /etc/rndc.conf /etc/rndc.conf.orig</code></p> <p>Copy the secret key from the rndc.key file into the rndc.conf file: <code>vim /etc/rndc.key</code></p> <p>...copy out the <b>secret</b> line...</p> <pre>vim /etc/rndc.conf</pre> <p>...then paste it over the <b>secret</b> line in the <b>rndc.conf</b> file...</p> <p>Also, within the <b>rndc.conf</b> file, change all instances of <b>key</b> to <b>rndckey</b> to look similar to this:</p> <pre>options {     default-server localhost;     default-key "rndckey"; };  server localhost {     key "rndckey"; };  key "rndckey" {     algorithm hmac-md5;     secret "koS5xGJSU0EtjlHqagrHng=="; };</pre>
---	--

**update.sh script**

```
#!/bin/bash

ToAddr="sysadmin@mydomain.local"
FromAddr="DNS_Spyware_Server@mydomain.local"
stopBind="/etc/init.d/named stop"
startBind="/etc/init.d/named start"
restartBind="/etc/init.d/named restart"
malware="/var/named/malware"
syslog="/var/log/messages"
downloadedZoneFile="spywaredomains.zones"
sedZoneFile="spywaredomains.sed"
activeZoneFile="spyware.zones"
backupZoneFile="spyware.zones.bak"

errorsFound ()
{
  mv $activeZoneFile $downloadedZoneFile
  mv $backupZoneFile $activeZoneFile
  $stopBind
  sleep 5
  $startBind
  errors="Errors found while installing spyware zone file."
  checkStarted=$(host google.com 127.0.0.1 | wc | cut -d' ' -f7)
  if [ $checkStarted = "1" ]
  then
    started="DNS Spyware Zone File . CRITICAL ERROR - SERVICE IS NOT STARTED"
  else
    started="DNS Spyware Zone File . ERROR - Old File Restored"
  fi
  echo $error1\n\n$error2 | mail -s "$started" $ToAddr -- -r $FromAddr
  return 0
}

### Change to proper directory
cd $malware

### Remove old downloaded file if it exists
rm -f $downloadedZoneFile

### Pull down the latest zones file
wget http://www.bleedingsnort.com/blackhole-dns/files/spywaredomains.zones

### Edit downloaded file with correct path information
sed -e 's#/etc/namedb#./malware#g' ./downloadedZoneFile > ./sedZoneFile

### Check if the downloaded file is the same as the current file
if [ -f "$activeZoneFile" ] #if spyware.zone exists
then
  same=$(diff --brief $sedZoneFile $activeZoneFile) #if spywaredomains.sed and
  spyware.zone files are the same, the variable same will be empty. If they are
  different the variable will not be empty
else
  cp $sedZoneFile $activeZoneFile
  same=""
fi
```

```
### If they are the same OR spyware.zone did not exist, then do nothing and
exit
if [ "$same" = "" ]
### Debug
#then
# echo "same and exit"
# echo $same
# exit 1
#else
# echo "different and continue"
# echo $same
# exit 1
#fi

then
  echo "Duplicate zone file - nothing changed" | mail -s "DNS Spyware Zone
File. No Changes" $ToAddr -- -r $FromAddr
  rm -rf $sedZoneFile
  rm -rf $downloadedZoneFile
  exit 1

  ### If they are different, backup current file and replace with new
else
  rm -f $backupZoneFile
  mv $activeZoneFile $backupZoneFile
  mv $sedZoneFile $activeZoneFile
  rm -rf $downloadedZoneFile

  ### Restart BIND
  $restartBind
fi

### Wait 10 minutes then check for errors in the syslog on starting BIND
sleep 10m
error1=$(grep named $syslog | tail -n 500 | grep "file not found")
error2=$(grep named $syslog | tail -n 5 | grep "exiting (due to fatal
error)")

if [ -n "$error1" ]
then
  errorsFound
  exit 1
elif [ -n "$error2" ]
then
  errorsFound
  exit 1
fi

echo "DNS Spyware zone file has been updated successfully" | mail -s "DNS
Spyware Zone File - Updated" $ToAddr -- -r $FromAddr
```