DNS

Jason Healy, Director of Networks and Systems

Last Updated Mar 18, 2008

Contents

Chapter 1

DNS

Last updated 2008/03/18

1.1 Introduction

The **D**omain **N**ame **S**ystem, or **DNS** performs the valuable service of translating human-readable names (such as `www.example.com`) into computer-usable IP addresses (such as `192.168.1.100`). Many modern network services require a properly functioning DNS setup in order to work correctly.

A full treatment of DNS is far beyond the scope of this document. This document assumes that you are familiar with the basics of DNS, including:

- What a nameserver is, including the difference between a **master** and **slave** nameserver
- How to query a nameserver
- Forward and reverse records
- SOA, A, CNAME, and MX records

If you need to brush up on your basics, please consult the "bible" of DNS: *DNS and BIND* by Paul Albitz and Cricket Liu (published by O'Reilly).

1.1.1 Suffield's Topology

Suffield currently uses 5 nameservers in its DNS topology. We have a single master DNS machine running on an Adonis DNS Appliance. Outside of our firewall, we have 2 slave nameservers that maintain our "public" DNS information (these servers are hosted with our ISP). Inside the firewall, we have 2 more slaves that maintain our "private" (internal) DNS information. These servers also act as the primary published servers for our LAN clients, and perform recursive and cached lookups for those clients.

Graphically, it looks something like this:

```
+-----------+                                  +-----------+
| Qwest NS1 |           (Internet)             | Qwest NS2 |
+-----------+                                  +-----------+
             \                                /
              \                              /
               +----------------------------+
               | Suffield Firewall          |
               | (Allows queries to Adonis) |
               +-------------+--------------+
                             |
                             |
                 +-----------+------------+
                 |         Adonis         |
                 | (Master for all zones) |
                 +------------------------+
                /                          \
               /                            \
       +------+                              +------+
       | Luke |      (Suffield LAN)          | Leia |
       +------+                              +------+
```

1.2 Master (Adonis) Configuration

Suffield currently uses an Adonis DNS Appliance as its master DNS server.

Because the Adonis is self-contained (it has its own GUI for updating zones), this document does not cover the setup of a BIND 9 master server "by hand". Please refer to a full DNS configuration guide for information on setting up a master server.

(In reality, a master server is very similar to a slave server, except for a few settings in how the zones are defined.)

1.2.1 Adonis Notes

The Adonis comes with a detailed manual which explains its features and use. The latest version may be obtained from BlueCat's website: www.bluecatnetworks.com.

Because the manual is so complete, we will not describe the full operation of the appliance here. Instead, we shall describe how we update our appliance at Suffield.

The Adonis appliance runs BIND 9 on Linux. Thus, all the configuration data for DNS are accessible, just as they would be under a hand-configured nameserver. All critical DNS data live in the following directory:

```
/jail/named/
```

Backups of that directory can be taken directly using SSH/SCP.

1.2.2 Updating the Zones

To update zone data on the Adonis, follow these steps:

1. Launch the **Adonis Management Console** on your local computer. The console is written in Java and available for several platforms.
2. After the program launches, you will be shown a welcome screen asking you to open a project. Choose **Check Project out from server**.
3. Enter your **username** (this is used to keep a log of who has checked out the configuration), **server** address, and **password**. Click the **Check-out** button.
4. The **Server Explorer** will open, showing the server configuration.
5. Make any desired changes to the server and zones.
6. **Deploy** the configuration onto the server and verify that your changes have taken effect.
7. In the **File** menu, select **Checkin**.
8. In the window that appears, type a comment about the changes that you made, and click the **Check-in** button to stored the saved configuration file back on the server.

Note that the check-in/check-out procedure does **not** publish any changes on the server. You must still **deploy** the new configuration to the server in order for the changes to take effect. The check-in/check-out procedure simply stores the configuration file on the server so multiple people can access it.

1.3 Slave Server Configuration

Once you have a master server that is up and running, you can create slave servers that will replicate the DNS zones from the master and make them available to clients.

Note: this section describes the theory on setting up a slave server. All of this functionality is included in our standard base config, so if you just want to set up a slave as quickly as possible, please skip to the section on our standard configuration base.

1.3.1 Master Configuration

The master server must be configured to allow the slave servers to query for entire zones of data. Additionally, the master will send notifications to configured slaves whenever the zone data changes, allowing them to update more quickly.

You must add an **NS** record to the domain for every slave you wish to notify. If you do not wish to add **NS** records (for example, if you do not wish to publish the IP of one of your nameservers), you must add the IP of the slave to the **also-notify** parameter on the master.

DNSSEC Keys

For extra security, you may wish to enable DNSSEC keys so that you can digitally sign your zone transfers. This prevents malicious users from stealing all your zone data, and also prevents certain types of poisoning attacks against your server.

On the Adonis, you may set up new DNSSEC keys by clicking on the **DNS Service** configuration item and clicking on the **Security** tab in the configuration window.

Once you've set the key, you may set the **allow-transfer** parameter to use this key (instead of an IP address or ACL).

If you're configuring your master server by hand, you can generate a new key on the command line by executing the following:

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST keyname.suffieldacademy.org.
```

That will generate a keyfile which contains the base-64 encoded version of the key, which can be added directly to the config file, like this:

```
key keyname.suffieldacademy.org. {
    algorithm hmac-md5;
    secret "<base-64 key goes here>";
};
```

Alternately, you can include the keyfile directly, so you don't have to keep the secret directly in the configuration:

```
include "/etc/bind/kename-file.key"
```

1.3.2 Slave Configuration

A slave configuration looks very similar to a master configuration. Most of the options, views, and other settings are exactly the same. The main difference for slave servers is that they have a type of **slave** with regards to their zone data, and they must be told which servers to contact to get authoritative zone information.

A slave server must be configured to query the master server(s) for the domains it slaves. This means adding stanzas like this to the config file:

```
zone "suffieldacademy.org" {
    type slave;
    masters { 192.168.0.1; };
    file "slave_suffieldacademy.org";
};
```

This tells the slave to be authoritative for the `suffieldacademy.org` domain, but get all of its information from the server at IP `192.168.0.1`. All the zone data are stored in the file `slave_suffieldacademy.org` in the name server's default directory.

We don't technically have to store the zone data in a file, but it has one major advantage: it serves as a backup of the zone data if the master should permanently lose its data. The slave zone files can be quickly "promoted" to master files, so having the zone data on hand can help in a disaster recovery scenario.

DNSSEC Keys

If the master server requires DNSSEC signatures on zone transfer requests (which re recommend), you must configure your slave server by giving it a key to use and instructing it to use the key for specific servers.

Start by adding a stanza that defines the key you wish to use:

```
key keyname.suffieldacademy.org. {
    algorithm hmac-md5;
    secret "<base-64 key goes here>";
};
```

Alternately, you can include the keyfile directly, so you don't have to keep the secret directly in the configuration:

```
include "/etc/bind/kename-file.key"
```

Next, you must specify which keys go with which server:

```
server 192.168.0.1 {
    keys { keyname.suffieldacademy.org.; };
};
```

This tells the slave to sign all requests to `192.168.0.1` with the given key.

1.4 DNS with Mac OS X Server

As of this writing, Mac OS X Server 10.4 includes the ISC BIND 9 nameserver by default. The Server Admin GUI allows you to configure and run a simple nameserver in either master or slave mode.

While the GUI is adequate for basic nameserver configuration, it does not allow for advanced configuration of BIND's options. Interface options, ACLs, dynamic uptates, TSIG security, and other features are not accessible from Apple's GUI.

For this reason, we choose to configure BIND "by hand" on our servers running Mac OS X Server. The GUI may still be used to start and stop the DNS service, but should not be used to perform any configuration of the server itself.

Additionally, there is an issue with IPv6, BIND, and certain root server queries. As of this writing (Mac OS X 10.4.6, BIND 9.2.2), caching nameservers run on Mac OS X will have extremely slow queries to the root name servers (4 seconds or more). This causes slow lookups for domains on the first attempt, though later attempts are cached and therefore not affected.

BIND 9.3 introduced a flag (`-4`) that temporarily disabled IPv6 which works around this problem. While not an ideal solution (the proper solution would be to fix IPv6 support), it does prevent the bug from happening. Unfortunately, the version of BIND shipped with Mac OS X (9.2.2) lacks this flag. We therefore compile our own version of BIND 9.3 and use it instead.

A version of BIND 9.3 can be found in the DarwinPorts collection, and can be installed without interfering with the native version of BIND shipped with Mac OS X.

1.4.1 Custom Changes

Our DNS configuration for Mac OS X Server builds on the stock configuration created by Apple. Because of this, the configuration file cannot be edited using the **Server Admin** GUI (it will overwrite the changes).

For more information on the changes we make to the config files, please see the section about our standard configuration base.

We currently run our Mac OS X machines as slaves to our master server. Therefore, we use the base config and only include information that pertains to slaves.

1.4.2 Configuration Files

Our standardized config lives in an entirely different directory from the BIND configs that ship with Mac OS X. Therefore, you don't need to worry about them interfering with each other.

1.5 Suffield DNS Config Template

To make setting up nameservers easier, we have broken our standard DNS config into several include files, along with a nameserver template file. Taken together, these files can be quickly pieced together to form a functioning master or slave nameserver.

The files are stored in revision control, and can also be viewed directly from the web:

Suffield DNS Config Template

For servers, we recommend checking the files out of version control. This way, changes to the files can be easily synched up using our version control software.

1.5.1 Differences

Our configuration is broken up into several different files, which makes it easy to plug in different functionality for different servers while maintaining a consistent set of global options. For example, converting a server from master to slave would require only two lines to be edited.

In the file segments themselves, these are the major pieces of functionality we implement:

- A "main" template file, which is commented to show which sections must be edited to create a functioning configuration. Common configuration options are listed here.
- An ACL file with common Access Control Lists for address blocks.
- Two "views" (internal and external) based on the ACLs which define what answers the server will provide. We have a split-horizon DNS setup: answers are different depending on if a client is inside the firewall or outside.
- DNSSEC options, including cryptographic keys, signed updates, and slave delegation based on keys rather than source IP. Note that for security reasons, DNSSEC keys are **not** stored in source control. The file containing the keys must be crafted by hand on each server.
- Zone declarations, broken up by master/slave status and internal/external view status. Makeing thse things include files makes it very simple to update the zones on several machines (just pull a new version of the include file from source control and reload the server).

1.5.2 Using the Configuration

To use our base configuration, follow these steps:

1. Make sure DNS is not currently running on your server. Use the **Server Admin** GUI to stop the service if it is running.
2. Check out the stock config from our Subversion repository:

   ```
   cd /etc/

   sudo svn checkout \
   svn://svn.suffieldacademy.org/netadmin/trunk/software/dns/named.suffieldconf.d \
   named.suffieldconf.d
   ```

   The command above checks out the stock directory to the name `named.suffieldconf.d`. You may choose a different name for the directory, but you will need to change all of the `include` statements in the config file to reflect the new path.
3. If you're using DNSSEC (which we do), you'll need to create an include file with all the DNSSEC keys in it. First, copy our template config:

```
cp /etc/named.suffieldconf.d/key.inc.template /etc/named.suffieldconf.d/key.inc
```

Then, edit the file and add any keys that are needed for inter-server communication.

4. We need to create a few more directories where BIND will store its zone files:

```
mkdir /var/named/internal
mkdir /var/named/external
```

5. You should create a file in the `host_configs` directory of the `named.suffieldconf.d` directory. We recommend making the file on another machine, checking it into source control, and then pulling it down onto the server using `svn up`.

   Once the file is on the machine, create a symbolic from it to the standard `named.conf` location:

```
ln -s /etc/named.suffieldconf.d/host_configs/host_named.conf /etc/named.conf
```

6. Finally, copy the LaunchDaemon plist file for DNS into the server's `/Library/LaunchDaemons/` directory. You can download the file from our DNS LaunchDaemon repository.

At this point, you are ready to start the nameserver:

```
sudo launchctl load -w /Library/LaunchDaemons/org.isc.named.plist
```

Check the `/Library/Logs/named.log` file to see if the server starts correctly and loads its zone data. You can confirm that the service is running by typing:

```
sudo launchctl list
```

The `org.isc.named` identifier should appear if everything is running properly.

Important Note: the **Server Admin** GUI will show the status of the DNS server when it is running, but you **must not** use it to start and stop the service. Our LaunchDaemon file contains the pointer to our customized config files, whereas **Server Admin** will use the default one shipped by Apple. We use our own file because we observed **Server Admin** crashing when we overwrote Apple's config files with our own.

1.6 Disaster Recovery

Because we have multiple DNS servers running at all times, a short outage on one machine should not pose a serious problem for our DNS service. Even the loss of our master system would not cause a major disruption in DNS; the biggest problem would be that new records could not be added until the master was restored.

In the event that the master machine were offline for a long period of time (more than 4 days), the best way to recover would be to ”promote” one of the slaves to become a master. Currently, our slaves are configured to continue serving zone data until they have been out of contact with the master server for more than 1 week. If the outage can be corrected before that time, no promotion should be necessary.

Because our slaves hold all the zone data that the master does, promotion is relatively straightforward:

1. Change all the ”zone” entries to be of type **master** instead of type **slave**. Remove any references to master servers.
2. To avoid versioning problems, you should also rename the zone files, both on disk and in the config file. By convention, we name our master zone files with the word `zone_` in front of them (*e.g.*, `zone_suffieldacademy.org`). Slaves name their zone files with the word `slave_` in front (*e.g.*, `slave_suffieldacademy.org`). If you promote a machine to be a master, you should rename the zone files as well.
3. Make sure a promoted slave has all the keys that the defunct master had. We use TSIG to sign all transfer requests. The new master must have all the keys that the slaves will be using in order to complete the transfers.

   If you do not have access to a backup of the master's config file, you can reconstruct the keys by looking at the configurations of the slaves.

Be sure to save a copy of the old slave configuration so that you can ”demote” it back when you're done.