The Netdot Manual

Network Documentation Tool

Version 0.8

Table of Contents

This manual documents the installation, administration and operation of the Netdot application.

The Network Documentation Tool

Netdot [http://netdot.uoregon.edu] is an open source tool designed to help network administrators collect, organize and maintain network documentation.

Netdot is actively developed by the Network and Telecommunication Services [http://ns.uoregon.edu] group of the University of Oregon [http://www.uoregon.edu].

Netdot features include:

- Device discovery via SNMP
- Layer 2 topology discovery and graphing, using multiple sources of information: CDP+LLDP, Spanning Tree Protocol, switch forwarding tables, router point-to-point subnets.
- IPv4 and IPv6 address space management (also referred to as *IPAM*), including hierarchical organization, address block visualization and IP and MAC address location and tracking.
- Cable plant information including: sites, rooms, jacks, closets, inter and intra-building wiring, circuits, etc.
- Contact information for related entities: departments, providers, vendors, etc.
- Netdot can generate configuration files for various other tools, including: Nagios [http://www.nagios.org], Sysmon [http://www.sysmon.org], RANCID [http://www.shrubbery.net/rancid],

Cacti [http://www.cacti.net]. Starting with version 0.9, Netdot will also generate configurations for ISC BIND and ISC DHCPD.

Structure

Netdot consists of several components:

1. The database

   Our goal has been to make Netdot database-agnostic as much as possible. In principle, it should be able to use any database supported by Perl DBI. There are, however, some limitations to this, for example, schema migration scripts are db-specific and may not always be availab.e. Currently MySQL is fully supported. There is currently partial support for PostgreSQL.

2. The libraries

   The back-end code is a hierarchy of object-oriented Perl classes. It can function as an API as well. One advantage of this model is that presentation, collection and database can be separated among different physical machines.

3. User Interface (UI)

   The web user interface is built on a templating system called HTML::Mason.

4. Command Line scripts (CLI)

   Certain tasks, like device discovery, can be executed from the command line. Therefore, these tasks can be automated by running them periodically via CRON.

Installation

Requirements

- Perl 5.6.1 or later
- Apache2 with mod_perl2
- MySQL (5.x preferred) or PostgreSQL (limited support)
- Authentication Server. Netdot supports both Radius and LDAP.
  - For Radius, we recommend FreeRadius, available at:

    http://www.freeradius.org
  - For LDAP, you can try OpenLdap, available at:

    http://www.openldap.org
- The RRDtool package, including its Perl modules, available at:

  http://oss.oetiker.ch/rrdtool/
- The GraphViz package, available at:

http://www.graphviz.org

- Various Perl modules. To test for missing modules in your system, run:

  ```
  %make tesdeps
  ```

  ### Tip

  If any of these modules are provided by your distribution, we recommend installing those.

- You can install any missing modules automatically using the CPAN, by running:

  ```
  %make installdeps
  ```

  If you want to install modules individually, you can do this instead:

  ```
  %perl -MCPAN -e shell
  >install Module::Blah
  ```

  ### Tip

  If you have problems installing Net::IRR, it is possible that the Net::TCP module cannot be installed. There is an open bug and patch available at:

  http://rt.cpan.org/Public/Bug/Display.html?id=43071

Configuration

Netdot comes with a configuration file that you need to customize to your needs. You need to:

```
% cp etc/Default.conf etc/Site.conf
```

Then, modify Site.conf to reflect your specific options. The original file contains descriptions of each configuration item.

Netdot will first read Default.conf and then Site.conf

The reason for keeping two files is that when an upgrade is performed, the Default.conf can be re-written (to add new variables, etc.), without overwriting your site-specific configuration.

Upgrading

Netdot's database schema only changes between major versions. For example, if upgrading from 0.8.x to 0.9.x, you will need to run an upgrade script to adapt your current database to the new schema. This can be usually accomplished by running the 'make upgrade' command. However, if upgrading from 0.8.X to 0.8.Y, it should suffice to run 'make install' and restart Apache.

Installing the package for the first time

- Make sure you have created the file etc/Site.conf with your configurations (See above).

- From the top directory in the package, do:

  ```
  %make install [parameters]
  ```

Possible parameters include:

```
PREFIX=YOUR-PREFIX (default: /usr/local/netdot)
APACHEUSER=USER-YOUR-APACHE-RUNS-AS (default: apache)
APACHEGROUP=GROUP-YOUR-APACHE-RUNS-AS (default: apache)
```

- You will then be ready to initialize the database.

  ```
  %make installdb [parameters]
  ```

  Remember you need to set DB_DBA and DB_DBA_PASSWORD to your database's admin username/password in Site.conf before running this command. Or if you prefer, you can specify the DB_DBA and DB_DBA_PASSWORD values as parameters (DB_DBA and DB_DBA_PASSWORD are used by many functions in Netdot, they will need to be set to the correct value in Site.conf eventually)

  ```
  DB_DBA=DATABASE-ADMIN-ACCOUNT
  DB_DBA_PASSWORD=DATABASE-ADMIN-PASSWORD
  ```

Apache Configuration

Edit the supplied Apache config template for either Radius or LDAP, copy it to your Apache config dir and include it somewhere in your Apache config (e.g.):

```
Include conf/netdot_apache2_<radius|ldap>.conf
```

Alternatively, some Apache environments provide a directory from which files are included automatically when Apache starts. In that case, it's sufficient to copy the file into that directory, for example:

```
# cp /usr/local/netdot/etc/netdot_apache2_radius.conf /etc/apache2/conf.d/
```

Tip

Make sure you use the version of the file that gets copied into your install directory by 'make install'. This file contains relevant path substitutions based on your chosen install prefix

CRON jobs

Netdot comes with a few scripts that should be run periodically as cron jobs.

- Retrieval of forwarding tables and ARP caches for IP/MAC address tracking
- Devices should be re-discovered via SNMP frequently to maintain an accurate list of ports, ip addresses, etc.
- Rediscovery of network topology
- Netdot keeps history records for some objects every time they are updated. With time, old history should be deleted from the database to save disk space.
- Netdot can generate text documentation that is easy to find via simple grepping commands, for example, information about people, locations, device port assignments, etc. This documentation should be kept up to date by exporting it frequently.
- Configurations for external programs can be generated using Netdot data. Current supported programs include: Nagios, Sysmon and Rancid.

- The netdot.cron file included in the package is a sample crontab containing recommended periodic jobs. You should customize it to your liking and copy it to your cron directory, for example:

  ```
  # cp netdot.cron /etc/cron.d/netdot
  ```

Operation

Device Management

Netdot can discover and maintain an extensive amount of information about network devices. The easiest way to gather and store this information is by querying the devices using the Simple Network Management Protocol (SNMP). Devices can be discovered individually, by subnet, or by providing a text file with a list of devices.

Device Discovery using the web UI

Go to Management -> Devices. In the Tasks section, type the hostname or IP address of the device in question, along with the SNMP community and click [discover]. Netdot will then query the device using SNMP and present a window where you can assign an owner entity (for example, your organization), the entity that uses the device (for example, your customer), the location and a contact list.

If you are discovering a layer 3 device with ip forwarding turned on (such as a router or firewall), Netdot will ask you if you would like to automatically create subnets, based on the IP configuration of the device interfaces. This is a convenient way to add your subnets to Netdot.

Another option is to specify whether Netdot should assign any newly created subnets the same owner and user entities assigned to the device.

Once you click the [update] button, Netdot will show the discovery information and a link to the device page at the button.

You can always re-discover a device manually by using the [snmp-update] button on the top right corner of the device page. For example, if you have added a new port adapter, new interface cards, or if the device has been replaced with a different unit.

Device discovery using the command line interface (CLI)

For brevity, let's assume you are located at the Netdot installation prefix, for example, /usr/local/netdot.

You can discover a single device by executing:

```
bin/updatedevices.pl -H <device-name> -I -c <community>
```

You can also try discovering a whole subnet like this:

```
bin/updatedevices.pl -B 192.168.1.0/24 -I -c <community>
```

In addition, you can give Netdot a specific list of devices you would like to discover:

```
bin/updatedevices.pl -F <text-file> -I -c <community>
```

The file should contain a list of devices and their communities, separated by spaces, one device/community per line, for example:

```
device1 community1
```

```
device2 community1
device3 community2
...
```

Netdot can retrieve ARP caches and bridge forwarding tables. You will probably want to fetch ARP caches from your layer 3 devices (i.e. routers and firewalls), and forwarding tables from your layer 2 devices (switches). Examples:

```
bin/updatedevices.pl -H <router> -A -c <community>
```

```
bin/updatedevices.pl -H <switch> -F -c <community>
```

Netdot can also try to discover the network topology. For that, you need to run:

```
bin/updatedevices.pl -T
```

Ideally, once you have discovered all your devices, you should combine all this functionality and have it run periodically via CRON. A sample crontab entry would be:

```
0 * * * * root /usr/local/netdot/bin/updatedevices.pl -I -A -F -T
```

You will find some examples of cron jobs in the file named *netdot.cron*

Device Documentation

Once you have created a device, you can go ahead and add more information about it.

Going to Management -> Devices you can search a device based on its name, IP or MAC address.

From the device page, you can navigate to the different sub-sections depending on the information you want to edit.

Interfaces: Here you can edit interface descriptions, assign network jacks, etc. by clicking on the [edit] button. You can also edit a specific interface by clicking on its number or on its name. If you are running topology discovery, you will probably see neighbor information. If for some reason the topology discovery process cannot dectect a neighbor, you can add it manually by clicking on the [add] button in the neighbor column.

Manually adding a neighbor sets the "Neighbor Fixed" flag in the Interface object. This flag prevents the topology discovery process from removing the neighbor relationship.

Tip

Neighbor relationships tend to change frequently as hardware is replaced and connections are moved. Therefore, fixed neighbor settings can become out of date pretty soon. It is preferable to let the topology discovery process maintain neighbor relationships.

IP Address Space

Netdot is a useful tool for IP address space management (referred to as *IPAM* these days). Some key features are:

- Address space is hierarchically organized through the use of a fast binary tree algorithm, which is the same technique used by routers for doing route lookups.
- New subnets can be automatically created based on the interface information retrieved from routers and firewalls.

- Easy visualization of used vs. available address space for easier allocations

IP blocks

IP objects are called IP blocks. These objects can represent individual end-node addresses, as well as groups of addresses. The distinguishing characteristic is the prefix attribute. For example, an IPv4 block with a 32 bit prefix is an end-node address, while a block with a 24 prefix represents a group of 254 end-node addresses.

Each address or block has a corresponding status. Let's see those in detail.

IP block Status

IP objects are assigned a status to better represent their nature. Depending on whether the IP is an end address or a block, different status values can be assigned.

The status of an end-node address can be one of:

- Static: These are addresses that have been statically assigned to hosts or device interfaces.
- Dynamic: Addresses that belong to a DHCP pool
- Discovered: Addresses that have not been assigned as static or dynamic, but have been seen on the network (as part of ARP entries, for example).
- Reserved: Addresses that should not be assigned

On the other hand, the the status of an IP block can be one of:

- Container: This kind of block is meant to group or contain other blocks, such as Subnet blocks or other Container blocks. For example, let's say your whole IPv4 address space is 192.168.0.0/16. You also have partitioned this space into two /17 blocks, and from these blocks, you allocate subnets that you configure in your routers. In this case, you would have:

```
192.168.0.0/16 -> Container
    192.168.0.0/17  -> Container
        192.168.0.1/24 -> Subnet
        192.168.0.2/24 -> Subnet
    192.168.128.0/17 -> Container
        192.168.128.10/24 -> Subnet
        192.168.128.20/24 -> Subnet
```

- Subnet: This kind of block is meant to represent actual subnets that are configured on the interfaces of your layer 3 devices such as routers or firewalls. Subnets usually contain the end-node addresses that you assign to your users.
- Reserved: Similarly to reserved addresses, reserved blocks are not supposed to be allocated for whatever reason.

DNS

Support for DNS records will be available starting with release 0.9

DHCP

DHCP integration will be available starting with release 0.9

Contact Information

Cable Plant

Advanced DB operations

Reports

Exporting Configurations for External Programs

You can use the exporter tool to generate text files that can be used as configurations for third-party tools and programs.

The exporter tool is available in the web UI, under the "Export" tab. Simply select one or more programs and click on the [submit] button. Netdot will show some output from the exporter tool, including the paths to the new files.

Additionally, the exporter can be called from the command line. For example, to generate Nagios configurations:

```
bin/exporter.pl -t Nagios
```

Or you can export several in one call:

```
bin/exporter.pl -t Nagios,Sysmon,Rancid
```

There are specific export parameters for each of these which you can customize by editing your Site.conf file.