IPv6 Sauron - Quick How To

Padrta, A., Kostěnec, M.

September 22, 2014

Contents

1 Introduction

This document describes deploying Sauron 0.7.4 with IPv6 support. Three possible scenarios are considered:

1. upgrade from lower version (version 0.7.2. or 0.7.3 is supposed),
2. new installation & transfer IPv4 database from old Sauron (version 0.7.2. or 0.7.3 is supposed) & import IPv6 data,
3. new installation & import data from DNS & DHCP configurations.

2 Deployment Scheme

The deployment scheme is depicted in Figure 1. Starting point is either IPv4 Sauron or new user, who wants to use Sauron IPv6. The final result is fully operational Sauron with IPv6 support. All three above mentioned scenarios are depicted.

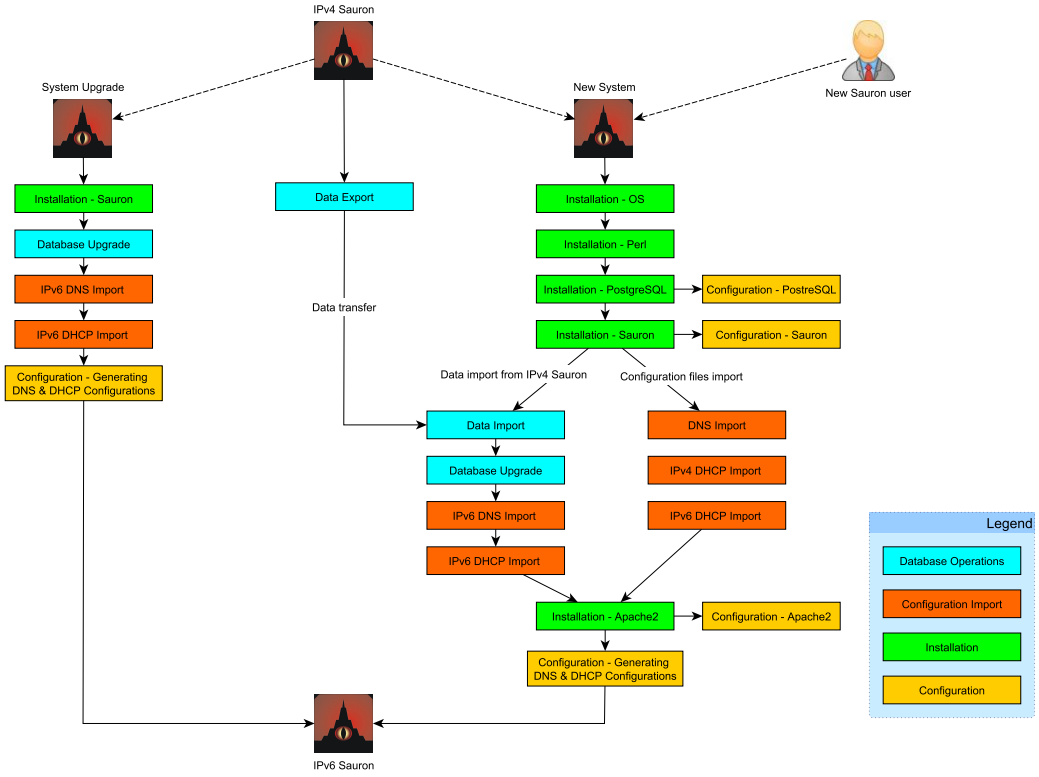

Figure 1: Deployment Scheme

Individual blocks used in Figure 1 are described in following sections. In case of upgrading IPv4 Sauron is strongly recommended to perform backup of database and Sauron files, which is described in section 3.1. Recovery procedure is also provided in the same section.

2.1 Sauron Database Operations

This section is devoted to operations with database part of Sauron.

2.1.1 Data Export

Dump of complete Sauron database can be created by following commands:

```
# su postgres
$ pg_dump -U <sauron-dbuser> <sauron-dbname> > backup.dump
```

where `<sauron-dbuser>` is username, which is used to access database named `<sauron-dbname>`.

2.1.2 Data Import

When IPv6 Sauron is installed, import of Sauron database dump can be realized by following commands:

```
# su postgres
$ dropdb <sauron-dbname>
$ createdb <sauron-dbname>
$ psql -d <sauron-dbname> -f backup.dump
```

where `<sauron-dbname>` is Sauron database name. Database version has to checked by following command:

```
$ /usr/local/sauron/status
  ...
  Database version:      1.3
  ...
```

If the database version differs form 1.5, a warning `"Database version too old, 1.5 required"` will be displayed. In this case, the upgrade of database version is necessary. Simply follow instructions provided in section 2.1.3.

Note: When upgrading IPv4 Sauron, the database will be certainly in too low version.

2.1.3 Database Upgrade

In the directory `/usr/local/sauron/sql`, there are stored database conversion scripts. Each script can upgrade the database only by one version, thus the upgrade take several steps. For example, to upgrade database from version 1.3 (used by Sauron 0.7.2) to version 1.5, following commands have to be executed:

```
# su postgres
$ cd /usr/local/sauron/
$ ./runsql sql/dbconvert_1.3to1.4
$ ./runsql sql/dbconvert_1.4to1.5
```

Actual database version, i.e. the result of conversion, can be checked by following command:

```
$ /usr/local/sauron/status
  ...
  Database version:      1.5
  ...
```

Next, actual global information has to be imported into Sauron tables. Section 3.4 provides necessary details.

2.2 Components Installation

In this section, the component installation procedures are described. Information about appropriate component configuration can be found in section 2.3.

2.2.1 Operating System

Correct functionality was tested od Linux Debian distrubution, but is should work on all other Linux distributions. In fact, it should work on any OS, where necessary components can be installed.

2.2.2 Perl

To support all necessary function, Perl in version `5.10.1` or above is required. Next, there is need for some additional Perl modules. When using APT (Advanced Package Tool), following commands should be used:

```
# apt-get install perl
# apt-get install libnet-netmask-perl
# apt-get install libnet-dns-perl
# apt-get install libnet-ip-perl
# apt-get install libpg-perl
# apt-get install libdbd-pg-perl
```

Further, the module `Crypt::RC5` is required, but is not included in standard package. Thus, manual installation is needed:

```
# wget search.cpan.org/CPAN/authors/id/S/SI/SIFUKURT/Crypt-RC5-2.00.tar.gz
# tar -zxvf Crypt-RC5-2.00.tar.gz
# cd Crypt-RC5-2.00
# perl Makefile.PL
# make
# make test
# make install
```

Other Perl modules mentioned in original Sauron documentation are standard part of Perl `5.14.2`.

2.2.3 PostgreSQL

IPv6 support is present in PostgreSQL `8.2` and above, but version `9.1` (and above) is strongly recommended. Appropriate command for APT (Advanced Package Tool) is as follows:

```
# apt-get install postgresql
```

2.2.4 Sauron

Installation of Sauron is performed in the same way as in previous versions, i.e. by following commands:

```
# wget <sauron-webpage>/sauron-0.7.4.tar.gz -O /tmp/install/sauron-0.7.4.tar.gz
# cd /tmp/install/
# tar -zxvf sauron-0.7.4.tar.gz
# cd sauron-0.7.4/
# ./configure
# make
# make docs
# make install
```

In case of Sauron upgrade, execution of above command causes overwriting of existing Sauron files to new versions.

Next, symlinks to the Sauron CGI scripts should be created in the standard webserver CGI directory. In case of standard Apache installation, following commands can be used:

```
mkdir /usr/lib/cgi-bin/sauron
ln -s /usr/local/sauron/cgi/sauron.cgi /usr/lib/cgi-bin/sauron/sauron.cgi
ln -s /usr/local/sauron/cgi/browser.cgi /usr/lib/cgi-bin/sauron/browser.cgi
```

2.2.5 Apache2

Any webserver with CGI support can be used. Apache webserver, standard solution on Linux Debian OS, can be installed by following command:

```
# apt-get install apache2
```

2.3 Components Configuration

In this section, the configuration of particular components is described.

2.3.1 PostgreSQL

At first, the database user `<sauron-dbuser>` and Sauron database `<sauron-dbname>` have to be created by following commands:

```
# su postgres
$createdb <sauron-dbname>
$ createuser --pwprompt --no-createdb --no-adduser --password <sauron-dbuser>
<enter password>
<enter password>
```

The PostgreSQL database has different configuration files for each installed version. When more then one version od PostgreSQL is installed at once, be carefull to edit right file. Editing of configuration file for version `9.1` can be done by following command:

```
# vim /etc/postgresql/9.1/main/pg_hba.conf
```

New accesses to Sauron database have to be added as follows:

```
# TYPE  DATABASE         USER                  ADDRESS         METHOD
local   <sauron-dbname>  <sauron-dbuser>                       password
host    <sauron-dbname>  <sauron-dbuser>       127.0.0.1/32    password
host    <sauron-dbname>  <sauron-dbuser>       ::1/128         password
```

where `<sauron-dbuser>` is an username, which is used by Sauron to access his own database named `<sauron-dbname>`.

Next, the PostgreSQL daemon has to be restarted in order to reflect changes in configurations, i.e.:

```
/etc/init.d/postgresql stop
/etc/init.d/postgresql start
```

Finally, the OIDs (Object Identifiers) have to be turned on, because Sauron rely on these. Prior to PostgreSQL 8.x, the OIDs were turned on by default, but since version 9.x the OIDs have been turned off by default. The process is simple:

```
# su postgres
$ psql <sauron-dbname>
sauron=> ALTER DATABASE <sauron-dbname> SET default_with_oids=on;
sauron=> \q
```

This setting is valid until database deletion (`dropdb <sauron-dbname>`). Thus the OIDs has to be turned on after each database creation (`createdb <sauron-dbname>`).

2.3.2 Sauron

Sauron main configuration file can be edited by following command:

```
# vim /usr/local/etc/sauron/config
```

In this configuration file, four following values have to be set:

```
$SERVER_ID = "<server-name>";
$DB_DSN = "dbi:Pg:dbname=<sauron-dbname>";
$DB_USER = "<sauron-dbuser>";
$DB_PASSWORD = "<sauron-dbuser-password";
```

Next, appropriate tables in Sauron database have to be created by following commands:

```
# cd /usr/local/sauron/
# ./createtables
```

Actual state of Sauron can be checked by command line utility `/usr/local/sauron/status`. If everything is fine, following result should be displayed:

```
# /usr/local/sauron/status
Sauron v0.7.4 status

Database connection:   OK
Database version:      1.5
CGI interface:         Enabled

No users currently logged in.
```

Next, global information has to be imported into empty Sauron tables. Section 3.4 provides necessary details.

At this point, creation of one superuser is recommended. To do this, use command line utility `/usr/local/sauron/adduser`. Details are provided in Sauron User Guide.

2.3.3 Apache2

The webserver configuration is standard. Usage of SSL (https) is strongly recommended. For example:

```
<VirtualHost *:443>
        ServerAdmin mr@sauron.evil
        ServerName mordor.evil

        SSLEngine on
        SSLCipherSuite ALL:...

        SSLVerifyDepth 3
        SSLCertificateFile /etc/apache2/ssl/server.crt
        SSLCertificateKeyFile /etc/apache2/ssl/server.key
        SSLCertificateChainFile /etc/apache2/ssl/ca-chain.pem

        DocumentRoot /var/www/sauron
        ErrorLog /var/log/apache2/ssl-mordor-error.log
        CustomLog /var/log/apache2/ssl-mordor-access.log combined
        ServerSignature Off

        <Directory />
```

```
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/sauron>
                Options Indexes FollowSymLinks
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/sauron/

        <Directory "/usr/lib/cgi-bin/sauron">
                AllowOverride None
                Options FollowSymLinks ExecCGI
                Order allow,deny
                Allow from all
        </Directory>

        Alias /icons/ "/usr/share/apache2/icons/"
        <Directory "/usr/share/apache2/icons">
            Options Indexes MultiViews
            AllowOverride None
            Order allow,deny
            Allow from all
        </Directory>
</VirtualHost>
```

Next, the webserver needs write permission to directory `/usr/local/sauron/logs`. In fact, the logs should not be in `/usr/local`, therefore transformation to symlink to `/var/log/sauron` will be better. Following commands will do the work for Apache (and appropriate user `www-data`

```
mkdir /var/log/sauron
chown root:www-data /var/log/sauron
chmod g+w /var/log/sauron
rm -r /usr/local/sauron/logs
ln -s /var/log/sauron /usr/local/sauron/logs
```

2.3.4 Generating DNS & DHCP Configurations

The life purpose of Sauron is to generate configuration for DNS server (APT package `bind9`) and DHCP servers (APT package `isc-dhcp-server`). Configuration can be generated by following commands:

```
# cd /usr/local/sauron/
# ./sauron --dhcp  <server-name> <dhcp-ipv4-output-dir>
# ./sauron --dhcp6 <server-name> <dhcp-ipv6-output-dir>
# ./sauron --bind  --updateserial <server-name> <dns-output-dir>
```

The configuration generator for DHCP is far more sophisticated than previous versions. The configuration handles correctly the dynamic address pools and pool access management based on DHPC classes. In fact, the changes should only fix known problems and do not cause new ones, but checking the result before production deployment is recommended. Next, the configuration for IPv6 version of ISC DHCP can be also generated.

The configuration provided by Sauron can be directly used in specific enviroments only. Probably, some extra modifications will be necessary – for example when using DNS views in BIND,

when using DNSSEC or when using the same configuration for more servers with slight modifications (*next-server option*, etc.). In this case, the finishing and distribution script has to be created according to recommendations described in section 3.3.

2.4 Configuration Import

In this section, the DNS and DHCP configuration import issues are described. Sauron is capable to maintain configurations of more servers, thus when importing more servers, the described procedures have to be repeated for each server, each one with different identifier `<server-name>`.

2.4.1 DNS Configuration Import

Following process is used to import complete DNS configuration (IPv4 and IPv6 together). Sauron utility *import* assumes ISC BIND 8.x or 9.x configuration.

1. Copy DNS configuration to directory `<dns-config-dir>`
2. Edit configuration and remove problematic parts:
   - Current version of Sauron do not support directive `view`, thus the configuration has to be converted to single-server configuration.
   - All included files `include filename`) must be specified by relative path (relative to `named.conf` files). Other option is to use `/etc/bind/` as `<dns-config-dir>`.
   - Comment default zones – standard ones according to appropriate RFCs (3849, 4193, 4291, 5735, 5737 a 6303), can be added from web GUI (Zones/Add default zones).
   - Solve other possible problems against standards (or import tool capabilities), whose will be reported during import attempts.
3. Start import by following commands:

   ```
   # cd /usr/local/sauron/
   # ./import <server-name> --dir <dns-config-dir> <dns-config-dir>/named.conf
   ```

   where `<server-name>` is selected name of server and `<dns-config-dir>` is a directory with DNS server configuration.
4. If the import script does not print `Import successfully completed!` in last row, there is an error in configuration and a correction is needed.
5. Warnings like

   ```
   ... invalid/unsupported RR type ...
   IGNORING unknown host entry ...
   ```

   point out to skipped parts of configuration. This part should by added manually – the import is done only once, thus it is usually more efective than to prepare automatic tool for importing.

If the original DNS configuration was maintained manually, the data inconsistency (mostly between forward and reverse zone) are very likely. Some of them can cause a crash of import. In fact, the import usually needs several iteration (try import – find problem – fix problem).

2.4.2 DNS Configuration Import – IPv6 Part Only

The following procedure is used to add the IPv6 records to Sauron, typically used if a previous version of Sauron was used for IPv4 configuration management, and IPv6 configuration was maintained separately.

Assumed input of script are AAAA records in separate file `<ip6-record>`, for example:

```
ipv6record      IN      AAAA    2001:db8:1801:1001::1
ipv6record2     IN      AAAA    2001:db8:1801::1001:dead
```

Then, the file is imported by following commands:

```
# cd /usr/local/sauron
# ./addhosts <server-name> <zone-name> <ip6-record> --addip --commit
```

The procedure has to be repeated for each zone. The option `--addip` has beed added to lates version of Sauron – the IP addresses are added to existing hosts (not replaced). In fact some host with IPv4 address obtain also IPv6 address.

Finally, the appropriate reverse master zone must be added via Sauron web GUI. Otherwise the DNS configuration will not be generated properly. In case of above mentioned example, the reverse master zone will be as follows:

```
1.0.8.1.8.b.d.0.1.0.0.2.ip6.arpa
```

2.4.3 IPv4 DHCP Import

Configuration of IPv4 DHCP can be imported after DNS import, because it only add information about MAC address and other settings to existing records. The global settings are also imported. The process consists of following steps:

1. Copy IPv4 DHCP configuration to directory `<dhcp4-config-dir>`
2. Start import by following commands:

   ```
   # cd /usr/local/sauron/
   # ./import-dhcp --global <server-name> <dhcp4-config-dir>/dhcpd.conf
   ```

3. If the import script does not print `All done.` in last row, there is an error in configuration and a correction is needed.
4. Current version of import tool is not capable to import records with multiple IP address (i.e. `fixed-address <IP1>, <IP2>;`). In this case, following warning appears:

   ```
   Ignoring host with multiple IPs: <domain-name>.<zone>
   ```

   Appropriate records have to be added manually via Sauron web GUI utilizing `ethernet alias` feature.
5. Sauron internal representation of dynamic pools require DNS records for all members of dynamic pools. If appropriate records are not present in DNS, following warning appears:

   ```
   Warning: <n> DNS records for pool '<start-IPv4>-<end-IPv4>' not found!
   ```

   If only few IP addresses are missing, manually correction (add missing records) is suitable. Some organizations used to have dynamic pools without corresponding DNS hostnames – in that case, Sauron command line utility `generatehosts` will help you to add appropriate records.

If the original DHCP configuration was maintained manually, the data inconsistency between DNS and DHCP is usually considerable. Thus, DHCP import will take several attempts.

Next, the Sauron command line utility `/usr/local/sauron/import-nets` can be used to define subnet structure – the results will be visible via Sauron web GUI (menu `Nets/+All`). Assume file `<nets-filename>`, which contains data about subnets:

```
<cidr>,"<name>","<description>","<vlan>","<t|f>"
...
```

Prior to run following commands, VLAN `<vlan>` has to added manually via Sauron web interface

```
# cd /usr/local/sauron
# ./import-nets --cidr=1 --name=2 --descr=3 --vlan=4 --dhcp=5 <server-name> \
    <nets-filename> --commit
```

Subnet import from file is suitable for one-time subnet definition. The mentioned tool `import-nets` is capable to work with both IPv4 and IPv6 CIDRs.

2.4.4 IPv6 DHCP Import

Configuration of IPv6 DHCP can be imported after DNS import, because it only add information about DUID address and other settings to existing records. The global settings are also imported. The process consists of following steps:

1. Copy IPv6 DHCP configuration to directory `<dhcp6-config-dir>`
2. Start import by following commands:

   ```
   # cd /usr/local/sauron/
   # ./import-dhcp --dhcp6 <server-name> <dhcp6-config-dir>/dhcpd.conf
   ```

3. If the import script does not print `All done.` in last row, there is an error in configuration and a correction is needed.
4. Sauron internal representation of dynamic pools require DNS records for all members of dynamic pools. If appropriate records are not present in DNS, following warning appears:

   ```
   Warning: <n> DNS records for pool '<start-IPv6>-<end-IPv6>' not found!
   ```

   If only few IP addresses are missing, manually correction (add missing records) is suitable. Some organizations used to have dynamic pools without corresponding DNS hostnames — in that case, Sauron command line utility `generatehosts` will help you to add appropriate records.

If the original DHCP configuration was maintained manually, the data inconsistency between DNS and DHCP is usually considerable. Thus, DHCP import will take several attempts.

The information about IPv6 subnets can be added in the same way as IPv4 subnets. Details are described at the end of section 2.4.3.

3 Recommendation

This section contains practical tips and recommendations based on nearly ten years experience with Sauron at the University of West Bohemia in Pilsen.

3.1 Backup of Sauron before Upgrade

Before upgrading the system to a new version, it is strongly recommended to make a backup of the original state. Procedures described in this subsection suppose standard installation of Sauron.

3.1.1 Create Backup

Sauron database dump has to be created by following commands:

```
# su postgres
$ pg_dump -U <sauron-dbuser> <sauron-dbname> > <backup-dir>/backup.dump
```

where `<sauron-dbuser>` is username, which is used to access database named `<sauron-dbname>`.

Next, the scripts and configuration files have to be copied by following commands:

```
# mkdir <backup-dir>/usr-local-sauron
# mkdir <backup-dir>/usr-local-etc-sauron
# cp -r /usr/local/sauron/* <backup-dir>/usr-local-sauron/
# cp -r /usr/local/etc/sauron/* <backup-dir>/usr-local-etc-sauron
```

Webserver configuration, crontab tasks, and user-defined scripts won't be affected by installation of new version of Sauron.

3.1.2 Restore from backup

Sauron database can be restored from dump by following commands:

```
# su postgres
$ dropdb <sauron-dbname>
$ createdb <sauron-dbname>
$ psql -d <sauron-dbname> -f <backup-dir>/backup.dump
```

Next, the scripts and configuration can be restored by following commands:

```
# cp -r <backup-dir>/usr-local-sauron/* /usr/local/sauron/
# cp -r <backup-dir>/usr-local-etc-sauron/* /usr/local/etc/sauron/
```

3.2 Recommended Sauron Configuration

Main configuration file `/usr/local/etc/sauron/config` contains many parameters, here is a digest of interesting ones.

If you want more restrictive mode for zone privileges (users can see hosts only in explicitly defined zones), set

```
$SAURON_PRIVILEGE_MODE = 1;
```

If you want to use external user autentization according to webserver configuration (for example WebAuth, CoSign, Shibboleth, etc.), set

```
$SAURON_AUTH_MODE = 1;
```

3.3 Scheme for generating the configuration

Sauron in installed on separate virtual server, i.e. it is detached from production DNS and DHCP server. The goal of following procedures is to ensure fully functional configuration on production servers at any time.

The scheme of configuration synthesis and its distribution is depicted in Figure 2. The final configuration arise from Sauron output and additive data for each configuration (for example SSHFP records in DNS). These configurations are subjected to syntactic and functional checks (for example daemon successfully started with provided configuration, all zones were loaded, no MX record disapeared, etc.). This test are easy to perform, because ISC BIND and ISC DHCP servers are installed. In the same version as on production servers, of course, but not accessible to clients.

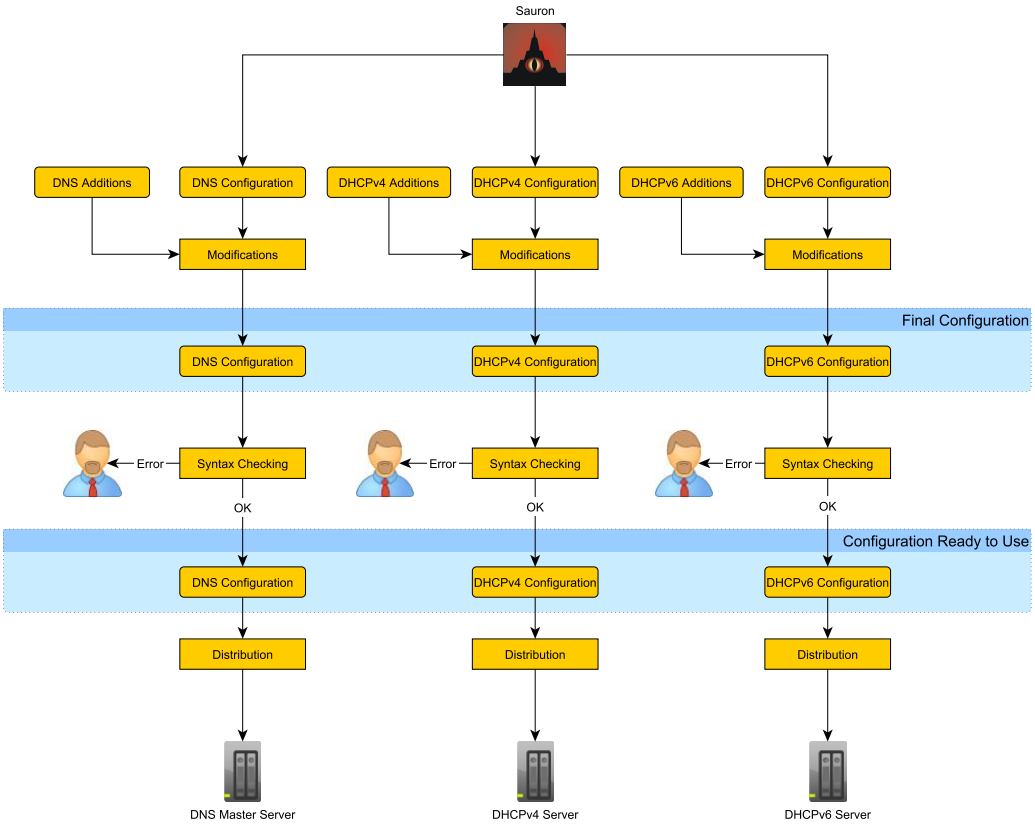

Figure 2: Scheme of recommended procedure to generate and distribute the configurations.

If some error occurs in one or more branches (DNS/DHCP IPv4/DHCP IPv6), the administrator is notified by e-mail and the process is stopped. Otherwise, the checked configuration is copied to designated storage and then distributed to production servers.

The distribution can be generally performed in two ways:

- Sauror server copy the checked configuration directly to the production servers and restart appropriate daemons.
- Sauron server copy the checked configuration to storage, which is accesible for production servers. Then, the servers downloads the appropriate configuration in regular interval or on change.

Above mentioned scheme ensure, that the production server will ever have functional configuration. Even Sauron user mistakes (for example incorrect `DHCP options`, accidentaly deleted MX record, etc.) do not cause malfunction of production DNS and DHCP servers.

3.4 (Re)initializing Global Information

Above the user-entered data, Sauron contains some global information:

- List of DNS root server allows proper generation of "." zone.
- IANA OUI (Organizationally Unique Identifier) allows to derive manufacturer from MAC addres.

Information about DNS root servers have to be imported (for new installation) or updated (for upgrade). In fact, it should be sometines updated to maintain actual data. This can be done by following commands:

```
# wget http://www.internic.net/domain/named.root /tmp/named.root
# cd /usr/local/sauron
# ./import-roots default /tmp/named.root --update
```

Information about OUI is not necessary for Sauron function. To insert fresh information about OUI, run following commands:

```
# cd /usr/local/sauron
# wget https://standards.ieee.org/develop/regauth/oui/oui.txt -O /tmp/oui.txt
# ./import-ethers /tmp/oui.txt
```