nsupdate.info Documentation

Release 0.7.0

nsupdate.info team

2014-09-30

Contents

CHAPTER 1

Introduction

1.1 About nsupdate.info

https://nsupdate.info is a free dynamic dns service.

nsupdate.info is also the name of the software used to implement it. If you like, you can use it to host the service on your own server.

Documentation: https://nsupdateinfo.readthedocs.org/

Software project: https://github.com/nsupdate-info/nsupdate.info (build and coverage are for latest repo code, package and downloads are for pypi release)

1.2 Features

- Frontend: Dynamic DNS updates via dyndns2 protocol (like supported by many DSL/cable routers and client software).
- Backends:
  - Uses DYNAMIC DNS UPDATE protocol (RFC 2136) to update compatible nameservers like BIND, PowerDNS and others (the nameserver itself is NOT included).
  - Optionally uses dyndns2 protocol to update other services - we can send updates to configurable 3rd party services when we receive an update from the router / update client.
- Prominently shows visitor's IPs (v4 and v6) on main view, shows reverse DNS lookup results (on host overview view).
- Multiple Hosts per user (using separate secrets for security)
- Add own domains / nameservers (public or only for yourself)
- Login with local or remote accounts (google, github, bitbucket, ... accounts - everything supported by python-social-auth package)
- Manual IP updates via web interface
- Browser-based update client for temporary/adhoc usage
- Shows time since last update via api, whether it used TLS or not
- Shows v4 and v6 IP addresses (from master nameserver records)
- Shows client / server fault counters, available and abuse flags

- Supports IP v4 and v6, TLS.
- Easy and simple web interface, it tries to actively help to configure routers / update clients / nameservers.
- Made with security and privacy in mind
- No nagging, no spamming, no ads - trying not to annoy users
- Free and Open Source Software, made with Python and Django

1.3 ChangeLog

1.3.1 Release 0.7.0

Important notes:

- WE_HAVE_SSL configuration setting name was changed to WE_HAVE_TLS. Please update your configuration, if you use it.
- Django 1.6.x required now, if you want to use 1.5.x: see setup.py

New Features:

- i18n support (uses preferred language from UI or browser)
- fr/de/it translations added
- translations are on transifex, you can help there! https://www.transifex.com/projects/p/nsupdateinfo/
- add m0n0wall configuration help
- add pfSense configuration help
- implemented host delete API at /nic/delete to remove A or AAAA record in DNS (very similar to the dyndns2 update api, which does not offer this)
- host delete functionality on web UI
- custom templates (for legalese, site-specific notes, etc. - see docs for details)
- abuse / abuse blocked flags + script support (see docs)
- notification by email if host gets flagged as abusive
- show example zone file for bind9 after adding a new domain
- better display in the admin
- enabled Django's clickjacking protection middleware in settings

Fixes:

- fix some status 500 errors / unhandled exceptions:
  - when domain does not exist
  - on profile view when not logged in
  - DnsUpdateError (e.g. SERVFAIL)
  - NoNameservers exception
  - UnknownTSIGKey exception
  - "Network is unreachable" error

  - empty ?myip=
  - invalid ip address strings in updates (now: "dnserr")
- fix html validation errors
- fix login url generation in activation_complete template, issue #139
- switch off recursion when querying master dns, issue #142
- fix –reset-available cmdline option processing
- updated dd-wrt configuration with verified settings

Other changes:

- also support Python >= 3.3 (experimental, please give feedback)
- improve looks, UI / UX
- improve docs, sample configs
- remove requirements from setup.py that were only for development
- removed view for legalese (please solve locally, according to your law - you can use custom templates for this)
- added some ugly logos (if you can do better ones, please help) https://github.com/nsupdate-info/nsupdate.info/issues/78
- replaced "SSL" by "TLS" everywhere. SSL is the old/outdated name. Since 1999, it's called TLS.
- updated to latest versions on CDN: jquery, bootstrap, font-awesome

1.3.2 Release 0.6.0

Important notes:

- importing from nsupdate.settings does not work any more (nor does the nsupdate.local_settings hack work any more). in your local_settings.py, please do your imports like this:

```
from nsupdate.settings.dev import *   # for development
# alternatively:
from nsupdate.settings.prod import *  # for production
# after that, override whatever you need to override.
```

- if you run Django 1.6.x, you manually need to apply a patch for django-registration (until that package is fixed for django 1.6 compatibility), see the django-registration-dj16-fix.diff in the toplevel directory of the repo.

New Features:

- browser/javascript-based update client (the URL you need is shown in the "browser" help panel after you add a host or generate a new secret).

Other changes:

- cleaned up how settings work, improved docs about a sane settings setup
- document postgreSQL setup
- also support Python 2.6.x
- also support Django 1.6.x
- for debugging, added django-debug-toolbar

1.3.3 Release 0.5.0

Important note (only for upgrades):

There is an issue if you use "south" and the "sqlite" database - it can't add BooleanFields and set the default values correctly when using "migrate".

As we added some critical fields, you need to use these commands immediately after running "django-admin.py migrate" to make sure their initial values are correct:

```
# all hosts will be available, no host will have abuse flags set:
django-admin.py faults --reset-available --reset-abuse --reset-abuse-blocked
```

Fixes:

- use python-social-auth exception middleware to catch exceptions
- status view is for logged-in users only (it was removed from navigation, but still accessible by URL in previous releases)
- fix session cookie behaviour to be more private for not-logged-in users

New Features:

- "update other services" feature (act as dyndns2 client to update 3rd party services when we receive an update)
- added per-host fault counters for update client and dns server
- abuse handling (for clients triggering too many faults) using the "faults" management command
- abuse-blocked / abuse / unavailable counts on status view
- notfqdn and abuse dyndns2 api result codes supported
- show reverse DNS of current IPs (only on host overview)
- customizable footer (use a custom base_footer.html template)

Other changes:

- use sane field lengths in the DB
- more help texts, more hints, better docs
- workflow for adding a domain is now similar to adding a host
- improved user interface
- use travis-ci and coveralls services for the project
- updated bootstrap to 3.0.2 (from cdn)

1.3.4 Release 0.4.0

Fixes:

- fix api return value (no "noauth", just "badauth")
- fix invalid /detectip/None URL for fresh session
- make IP detection on the web UI a bit more reliable
- fix KeyErrors in logging (at least for default format)

New Features:

- use REMOTE_ADDR for one of the 2 IP detections

- add a warning on the UI if the user has no javascript enabled
- use real session cookies by default (that get cleared on browser close)
- support "keep me logged in" if user wants a permanent 14d cookie
- use html5 autofocus to put cursor into the right input field
- python manage.py testuser to reinitialize test user (see docs)

Other changes:

- document clearsessions usage
- more tests

1.3.5 Release 0.3.0

- Fixes security issue https://github.com/nsupdate-info/nsupdate.info/issues/81
- improved logging levels, added log output at some places
- dnserr dyndns2 result supported
- more safe bind9 configuration example
- support for single-host update secrets
- make dnstools unit tests work everywhere
- remove beta from version number (but keep general beta state in pypi classifier)

1.3.6 Release 0.2.0b0

First release on PyPi.

CHAPTER 2

Using the service

2.1 Requirements

2.1.1 Update client

The best way to use the service for updating a hostname with a dynamic address is to have a dyndns2 compatible update client.

Usually this kind of software is built-in in your internet router (search for "dynamic DNS", "DDNS", "dyndns" on its user interface).

Alternatively, you can also run a software on a PC / server (like ddclient for Linux).

Or even just use your browser to update your IP via the web interface of the service.

Note: please do not "update" your IP address if it did not change. Doing so is considered abusive use of the service. All sane dyndns2 clients only send an update if the IP address has changed.

2.1.2 Web interface

When using a browser for administrating your hosts / domains via the web interface of the service, please:

- use https (for security)
- have cookies enabled (we need them for keeping the session after you logged in)
- have javascript enabled
- use a sane browser, like Firefox, Chrome/Chromium or Safari

2.2 Functionality of the Web Interface

2.2.1 Your current IP(s)

At some places, we show your current IP address(es). Depending on the type of your internet connection, this can be IP v4 or v6 or both (dual stack).

At some places, we additionally show the result of a reverse DNS lookup ("rDNS") for your IP address(es). If nothing shows up for an IP, the IP does not have a reverse DNS record.

We always show you the IP addresses where your requests come from. Under some circumstances, these might not be what you expect (e.g. presence or non-presence of NAT gateways, proxies, etc.).

We detect your addresses by 2 means:

- your current remote address (where your accesses to the web interface come from) - the IP detected this way is immediately visible on the web interface.
- if we don't already have the IP address from the remote address, we use an invisible fake "image" that your browser loads from an IPv4-only or IPv6-only server - the IP detected by this method usually shows up after a few seconds.

We do some optimizations to not load these images too frequently, but also try to make sure we do not show you outdated information about your current IP addresses.

If you don't see an IP address of some kind (v4 or v6) after a few seconds, it means you don't have that kind of address (plus working connectivity of that kind).

2.2.2 Register / Login / Logout

You need to create an account to use most of the functionality of the service.

Your hosts / domains are only for you, so you need to identify to create or change them.

You need to give a valid E-Mail address, as we send you a link you need to access to complete the registration.

We'll also use that E-Mail address in case you forget your login password.

For your own safety, use https and a sane password.

Be careful: in case you lose your login username/password and you also can't receive mail sent to the E-Mail address you gave when registering, you might not be able to regain access to your account / your hosts (neither automatically nor with help from service admin) as you likely can't prove that they are really yours / you are permitted to control them.

2.2.3 Hosts

You can add hosts to all the zones (base domains) offered to you. Usually this will be one or more zone(s) offered by the service operator, but you can even add your own domains (see the separate section about domains).

After creating a new dynamic host name, we'll show you an automatically created update secret for that host. You need it for configuring your update client and we show you example configurations for some popular routers and clients on the same page.

In case you lose the update secret, just create a new one (and enter it in your router / update client).

IP v4 and v6 addresses work completely independently of each other, you need to send 2 updates if you want to update both. If you want to be specific about which IP address you update, use our IPv4-only or IPv6-only host to make sure it is the v4 (or v6) address.

After configuring a new update client, please keep an eye one the Faults column on the overview page. It shows 2 values: C: <client faults> S: <server faults>

An increasing number of client faults usually means you (or the software you use) are doing something wrong (e.g. sending updates although your IP address did not change). If you see that, please fix it!

An increasing number of server faults means there is either something wrong with the nameserver or the connection to it or it is rejecting the updates for your hostname.

2.2.4 Domains

If you control an own nameserver / zone, you can use the service to dynamically update it with your router / update client.

For this, it is required that the master nameserver of that zone accepts dynamic updates (RFC 2136) using a shared secret. If you run your own bind9 nameserver for your domain, we show you how to configure it for dynamic updates after you add a domain on nsupdate.info.

You can either privately use such an own domain or alternatively even offer them publically for all users of the service.

If you have cool domains, please offer publically!

Note: if you just register a domain at some domain seller (and the domain seller runs the DNS for you), you usually just get some web interface to manage the DNS records. Often, that nameserver is not configured to accept dynamic updates (RFC 2136) unless otherwise noted by your DNS hoster. If unsure, read their documentation, examine their web interfaces (if they allow dynamic updates, there should be some means to configure or see the update algorithm, secret and maybe even the update policy (where you can setup rules to allow/deny specific hosts) or just ask them.

If your DNS hoster does not support dynamic updates, there is some trick how you still can use them:

```
# configure this for your domain at your DNS hoster:
dynamichost.yourdomain.com  CNAME  updatedhost.nsupdate.info
```

At the nsupdate.info site, add a host "updatedhost.nsupdate.info" and keep it updated using an update client.

2.2.5 Other Services Updaters

Users can associate "other services" (3rd party services) updaters with their hosts and if we receive an update for such a host, we'll automatically send (dyndns2) updates to these other services.

You can choose which kind of IP addresses shall be sent to the other service using the "give IPv4" and/or "give IPv6" options.

Currently, Users can only use services that were made available by an admin (by adding the service record using Django's admin interface).

2.2.6 Browser-based Update Client

The service has a "built-in" browser/javascript-based update client that will query the IP and send update requests if the IP changes.

One typical scenario where this is useful:

- you are an admin for multiple, sometimes rather ad-hoc clients where you have to do remote support / maintenance
- the clients have no (working) dynamic dns host / updater configured
- you have prepared a hostname in the nsupdate.info service you use just for such scenarios, e.g. "yourname-adhoc" (+ the base domain you use)
- you need to do some remote work, but you want to avoid losing access in case you get disconnected and the IP changes
- you don't want to require the client to find out his/her current IP and communicate it to you nor do you want to remember an IP address if you can have a nice (and always same) hostname

How to optimize this scenario:

- go to the "yourname-adhoc" entry and use "Show Configuration"
- copy and paste the URL shown in the "Browser" tab of the configuration help panel, under headline "Browser-based update client"
- optional: try it yourself in your browser
- give this URL to your client (E-Mail, Chat, ...), tell the client to open it with a browser and keep that page open in the browser until you're finished.
- once the client has done that, "yourname-adhoc" will point to the client's IP

Note:

- we show 3 slightly different URLs:
  - the first one is generic and will use either IP v4 or v6,
  - the other 2 are specific and will either enforce usage of IP v4, or v6.
- this whole browser-based mechanism is only for adhoc and temporary use - if you need something permanently or repeatingly, please configure a real update client
- if you can't electronically give the URL to the client, you can also give:
  - URL: like above, but remove the "yourname-adhoc.basedomain:secret@" part
  - when clients visits that URL, it will ask for username and password:
    - User name: yourname-adhoc.basedomain
    - Password: secret
  - let the client check "Last update response". Should be "good" (or "nochg") plus same IP as shown below "My IP". If it shows something else, then there likely was a typo in the user name or password.

CHAPTER 3

Administrating the service

3.1 Installation (for development/testing)

Create and activate a virtualenv for the installation (here with virtualenvwrapper):

```
mkvirtualenv nsupdate
workon nsupdate
```

Clone the repo and cd into:

```
git clone git@github.com:nsupdate-info/nsupdate.info.git nsupdate
cd nsupdate
```

Then install the software with requirements to your virtual env:

```
pip install -r requirements.d/dev.txt
pip install -e .
```

3.2 Configuration

3.2.1 nsupdate.info Service

First, please read the nsupdate/settings/*.py files - they contain a lot of settings you can use to customize your nsupdate.info installation. dev is for a development setup, prod is for a production setup and base has settings that are common for both.

But do not change anything in there, but rather create your own local_settings.py file, import from our settings and override anything you want to change afterwards.:

```
from nsupdate.settings.dev import *
SECRET_KEY='S3CR3T'
```

IMPORTANT: you usually need to tell django what settings you want to use.

We won't document this for every single command in this documentation, but we'll assume that you either set DJANGO_SETTINGS_MODULE environment variable so it points to your settings module or that you give the –settings parameter additionally with all commands that need it:

```
export DJANGO_SETTINGS_MODULE=local_settings  # this is YOUR settings file
or
```

```
django-admin.py --settings=local_settings ...
python manage.py --settings=local_settings ...
```

Note: if Django can't import your local_settings module, make sure that your python search path contains the directory that contains local_settings.py:

```
# we assume here that local_settings.py is in current directory.
# alternatively, you could also give a specific path instead of .
export PYTHONPATH=.:$PYTHONPATH
```

3.2.2 Initialize the database

To create and initialize the database, use:

```
python manage.py syncdb
python manage.py migrate
```

3.2.3 Start the development server

```
python manage.py runserver
```

3.2.4 Nameserver

Now as the server is running, you can log in using the database administrator account you created in the syncdb step and use "admin" from the menu to start Django's admin.

You'll need to configure at least 1 nameserver / 1 zone to accept dynamic updates, see the "Domains" section in the "user" part of the manual.

3.3 Installation (for production)

You usually will use a production webserver like apache or nginx (not Django's builtin "runserver").

If you want to use a virtualenv: see the hints for development installation.

If you install from repo code, it is sufficient to use the production requirements file (will install less packages than for development):

```
pip install -r requirements.d/prod.txt
pip install -e .
```

Alternatively, you can just install the latest release from pypi:

```
pip install nsupdate
```

3.4 Configuration

As described for testing/development, but use nsupdate.settings.prod in your local_settings.py file.

Also, you will need to review the settings in the nsupdate.settings.prod module and override everything that is different for your setup into your local_settings.py file.

Note: if you do not setup ALLOWED_HOSTS correctly, your will just see status 400 errors.

3.4.1 WSGI

Module nsupdate.wsgi contains the wsgi "application" object.

Please consult the webserver / django docs how to configure it and how to run django apps (wsgi apps) with the webserver you use.

Django has nice generic documentation about this, see there:

https://docs.djangoproject.com/en/1.6/howto/deployment/

Even if you do not follow or fully read the deployment guide, make sure that you at least read the checklist:

https://docs.djangoproject.com/en/1.6/howto/deployment/checklist/

3.4.2 HTTP Basic Auth

Additionally, you need to make sure that the "authorization" http header needed for HTTP Basic Auth gets through to the nsupdate.info wsgi application. Some web servers may need special settings for this:

```
WSGIPassAuthorization On  # use this for apache2/mod-wsgi
```

3.4.3 Static Files

As soon as you switch off DEBUG, Django won't serve static files any more, thus you need to arrange /static/ file serving by your web server.

We assume here that you configured your web server to serve /static/ URL from /srv/nsupdate.info/htdocs/static/ directory.

Django helps you to put all the static files into that directory, you just need to configure STATIC_ROOT for that:

```
STATIC_ROOT = '/srv/nsupdate.info/htdocs/static'
```

And then, run this:

```
umask 0022  # make sure group and others keep r and x, but not w
python manage.py collectstatic
```

This will copy all the static files into STATIC_ROOT.

Now, you must set DEBUG=False so it doesn't leak information from tracebacks to the outside world.

Make sure your static files really work.

3.4.4 PostgreSQL

For production usage and better scalability, you may rather want to use PostgreSQL than SQLite database. Django stores its sessions into the database, so if you get a lot of accesses, sqlite will run into "database is locked" issues.

Here are some notes I made when installing PostgreSQL using Ubuntu 12.04:

First installing and preparing PostgreSQL:

```
apt-get install postgresql  # I used 9.1
apt-get install libpq-dev  # needed to install psycopg2

# within the virtual env:
pip install psycopg2

sudo -u postgres createdb nsupdate
sudo -u postgres createuser --no-createrole --no-superuser --no-createdb --pwprompt nsupdate
# enter reallysecret password, twice
sudo -u postgres psql -c 'GRANT ALL PRIVILEGES ON DATABASE nsupdate TO nsupdate;'

sudo vim /etc/postgresql/9.1/main/pg_hba.conf
# by default, postgresql on ubuntu uses only "peer" authentication for unix sockets, add "md5"
# (password hash) authentication, otherwise it might use your login user instead of the configured us
# local   all             all                                     md5
```

To make nsupdate.info (Django) use PostgreSQL, put this into YOUR settings:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'nsupdate',  # database name
        'USER': 'nsupdate',
        'PASSWORD': 'reallysecret',
        'HOST': '',  # Empty for localhost through domain sockets or '127.0.0.1' for localhost throug
        'PORT': ''  # Set to empty string for default.
    }
}
```

Now proceed with syncdb / migrate as shown above.

3.5 Customization of the Web UI

You likely will need to customize the Web UI a bit, here is how.

3.5.1 Overriding the builtin templates

If you want to add/modify footers or headers or if you need to add stuff into the HEAD element of the html, you can override some includes we made to support this usecase.

Create an custom template directory (not within the repository / code directory) and add it to your settings, e.g.:

```
TEMPLATE_DIRS = ('/srv/nsupdate.info/templates', )
```

Below that template directory, you can override the builtin templates by just using the same relative name, e.g.:

- includes/base_footer.html (footer of all web UI views)
- main/includes/home_bottom.html (bottom of main view)
- (there are more of these, just look into the code's template dirs)

Best way to start is likely to copy the original file from the template directories located below the code directory into YOUR custom template directory and then slightly modify it.

As the templates might be cached in memory, you may need to restart your wsgi process to have them reloaded.

Note: it is advised that you keep local customizations to a minimum as if you override builtin templates with your customized copies, you will have to keep your copies in sync with future changes we make to the builtin ones.

3.5.2 Custom templates

If you need to add some simple views, just showing some simple templates (like e.g. if you have some footer links that link to these views to show some site- specific content, some legalese, ...), do it like that:

- have a footer and a custom template directory like described in previous section
- add files like main/custom/foo.html to that directory:

```
{% extends "base.html" %}
{% load bootstrap %}
{% block content %}
This is content rendered from template "foo.html".
{% endblock %}
```

- link to the view made from that template like this:

```
<a href="{% url 'custom' template='foo.html' %}">
    link to custom foo.html view
</a>
```

3.6 Maintenance

3.6.1 Regular jobs

You need to run some commands regularly, we show how to do that on Linux (or other POSIX OSes) using user cronjobs (use crontab -e to edit it). Make sure it runs as the same user as the nsupdate.info wsgi application:

```
DJANGO_SETTINGS_MODULE=local_settings
# reinitialize the test user:
50 2 * * * django-admin.py testuser
# reset the fault counters:
55 2 * * * django-admin.py faults --flag-abuse=20 --reset-client --notify-user
# clear expired sessions from the database, use your correct settings module:
0  3 * * 1 django-admin.py clearsessions
# clear outdated registrations:
0  3 * * 2 django-admin.py cleanupregistration
```

3.6.2 Dealing with abuse

In the regular jobs example in the previous section, –flag-abuse=20 means that it'll set the abuse flag if the client fault counter is over 20 (and, for these cases, it'll also reset the fault counter back to 0).

–reset-client additionally sets all client fault counters back to 0, so all counts are just "per day".

–notify-user will send an email notification to the creator of the host if we set the abuse flag for it. The email will contain instructions for the user about how to fix the problem.

So, if you run this daily, it means that more than 20 client faults per day are considered abuse (e.g. if someone runs a stupid cronjob to update the IP instead of a well-behaved update client).

Hosts with the abuse flag set won't accept updates, but the user will be able to see the abuse flag (as ABUSE on the web interface and also their update client should show it somehow), fix the problem on the client side and reset the abuse flag via the web interface. If the problem was not really fixed, then it will set the abuse flag again the next day.

This procedure should make sure that users of the service run sane and correctly working update clients while being able to fix issues on their own without needing help from service administration.

For really bad cases of intentional or ongoing abuse, there is also a abuse_blocked flag that can only be set or reset manually by service administration (using django admin interface). While abuse_blocked is set, the service won't accept updates for this host. The user can see the ABUSE-BLOCKED status on the web interface, but can not change the flag.

3.6.3 Database contents

Users who are in the "staff" group (like the one initially created when creating the database) can access the admin interface (see "Admin" in the same menu as "Logout").

But be careful, the Django admin lets you do all sorts of stuff, admins are allowed to shoot themselves. Only give Django admin access ("staff" group membership) to highly trusted admins of the service.

3.6.4 Software updates / upgrades

Please read the changelog before doing any upgrades, it might contain important hints.

After upgrading the code, you'll usually need to run:

```
python manage.py migrate
```

This fixes your database schema so it is compatible with the new code.

Maybe you also need the next command (we bundle .mo files, but if you run into troubles with them, try this):

```
python manage.py compilemessages
```

Of course, you'll also need to restart the django/wsgi processes, so the new code gets loaded.

CHAPTER 4

The nsupdate.info software Project

4.1 History

The initial version of the nsupdate.info software was developed in 48h in the DjangoDash 2013 contest by:

- Arne Schauf
- Fabian Faessler
- Thomas Waldmann

4.2 Project site

Source code repository, issue tracker (bugs, ideas about enhancements, todo, feedback, ...), link to documentation is all there:

https://github.com/nsupdate-info/nsupdate.info

4.3 Translations

Translations are done on Transifex - please collaborate there to avoid double work / workflow issues:

https://www.transifex.com/projects/p/nsupdateinfo/

Translation update workflow (start from a clean workdir):

```
# pull all translations from transifex:
tx pull
# update the translations with changes from the source code:
django-admin.py makemessages -a
# push updated translation files back to transifex:
tx push -s -t
```

4.4 Contributing

Feedback is welcome.

If you find some issue, have some idea or some patch, please submit them via the issue tracker.

Or even better: if you use git, fork our repo, make your changes and submit a pull request.

For small fixes, you can even just edit the files on github (github will then fork, change and submit a pull request automatically).

CHAPTER 5

Standards used

- Frontend Update-API: dyndns2 protocol
  - dyndns2 api description on dyn.com
  - dyndns2 api description on noip.com
- Backend: dynamic DNS update
  - RFC2136

CHAPTER 6

Extensions we made to the standards

6.1 /nic/delete API url

The dyndns2 standard does not give a means to delete a DNS record (like A or AAAA), you can only update to a new address using /nic/update.

Thus, we created a /nic/delete URL that behaves just like the dyndns2 update api, but removes the A or AAAA record in DNS instead of updating it.

While the update API would actually use the given IP address to put it into an A or AAAA record, the delete API uses it only to determine the address type, whether it is IPv6 or v6 and then deletes the A or AAAA record.

CHAPTER 7

Security considerations

7.1 Transmission security

Use https for the web interface as well as for the update client (if possible).

Otherwise, your username / password (FQDN / update secret) will be transmitted in clear text (unencrypted).

The web interface will warn you if you use it via http. If WE_HAVE_TLS is set to True, it will suggest you better use the https site and link there.

Additionally, the service administrator can implement a redirect from the http to the https site within the web-server configuration for the WWW_HOST. The redirect should **not** be implemented for WWW_IPV4_HOST and WWW_IPV6_HOST as it is unknown whether all update clients can deal with a redirect.

For the router / update client configuration examples we show when creating a update secret, we use update URLs with https: (and we also tell why it might not work).

On the hosts overview page, we show whether we received the last update via TLS.

7.2 Login with remote vs. local Account

If you use a already existing remote account to log in into our service, you don't need to create a local profile (with username, E-Mail and password).

That way, we need to store less information about you - especially no password hash (and you also don't need to create a new password just for our service). So, this is a little more safe if you just consider our service.

BUT: If you use some external service to log in, you of course need to trust them for this purpose as *they* are telling "yes, this is really you".

Also, if you cancel the account on that external service and you don't have a local profile (login, E-Mail, password) with us, you will be unable to log in afterwards or recover access to your hosts/domains.

So maybe the best way is to first create a local profile (username, E-Mail, password), then log in and associate your other remote accounts with that local profile.

7.3 Passwords / Secrets / Keys

7.3.1 Interactive login password

We recommend that you use a rather strong and not guessable password for this. Do not re-use passwords, use a password system or a password manager.

The interactive login password for the web site is stored using Django's default hasher mechanism, which is currently pbkdf2 (a very strong and intentionally slow password hash). Brute-Force attacks against such hashes are very slow, much slower than against simple hashes like (s)sha1/sha256 etc.

It is NOT stored in clear text by nsupdate.info.

If you lose the password, you'll have to do a password reset via e-mail.

7.3.2 Automated update secret

The automated update secret for routers or other update clients is a random and automatically generated secret. We store it using the sha1 hasher of Django (which in fact is salted-sha1, a not very strong, but fast-to-compute hash).

Considering that a lot of routers or update clients store this secret in clear text in their configuration and often transmit it using unencrypted http (and not https), this secret is not too safe anyway. We also wanted to save some cpu cycles here and rather not use pbkdf2 for this regularly and automatically used secret.

It is not stored in clear text by nsupdate.info.

If you lose the secret, you'll have to generate a new one and change it in your update client also.

7.3.3 Nameserver Update Secret (backend, RFC 2136)

We currently store this secret (which is basically a base64 encoded shared secret, one per dynamic zone) "as is" into the database ("Domain" records there).

This is somehow critical, but also hard to do better - encryption would only help very little here as we would need to decrypt the update secret before using it, so we would need the unlocked decryption key on the same machine.

Make sure no unauthorized person gets that secret or he/she will be able to update ANY record in the respective zone / nameserver directly (without going over nsupdate.info software / service).

We support creating a random update secret, so you don't need an extra tool for this.

7.3.4 Other Services Update Secret (dyndns2 client)

We need to store this secret "as is" into the database for the same reasons as outlined above.

But: we tell you in the services overview whether we'll use TLS to transmit the update, so at least if TLS is enabled, it won't go unencrypted over the wire.

7.4 CSRF protection

We use Django's CSRF protection middleware.

7.5 Clickjacking protection

We use Django's clickjacking protection middleware.

7.6 XSS protection

Django's templating engine html-escapes inputs by default.

7.7 Cookies

The software ("as is") uses these cookies:

- "csrftoken" (host-only, for CSRF protection)
- "sessionid" (host-only, to keep the session when you have logged-in to the web interface)

If you have set WE_HAVE_TLS to True (because you run the software on a https site), you should also set *_COOKIE_SECURE to True to avoid the cookies getting transmitted via http.

We use a session cookie by default (gets cleared when you close the browser). If you check the "Keep me logged in" checkbox on the login screen, then we'll set a permanent cookie with a lifetime as configured by the site admin (SESSION_COOKIE_AGE, default: 14 days).

7.7.1 Be careful with domain cookies

The software ("as is") does not use any domain cookies.

In case you modify the software, please be extremely cautious with domain cookies and in case of doubt, do rather not use them.

If you use domain cookies (like for ".yourservice.net", the leading dot makes it a domain cookie), all hosts in that domain would be able to read and write these cookies. Your site (at e.g. www.yourservice.net), but also users' sites (like attacker.yourservice.net).

Obviously, this might lead to security issues with stealing, modifying and faking domain cookies.

7.8 Sessions

We use Django's more safe JSONSerializer to serialize session data. For Django >=1.5.3, the serializer is configurable. For Django >=1.6 json will be the default rather than the less safe pickle format.

7.9 Django's SECRET_KEY

Django's SECRET_KEY needs to be a long, random and secret string (it is usually set up by the administrator of the site).

The builtin default settings will try to read SECRET_KEY from an environment variable of same name. If there is no such environment variable, the SECRET_KEY will be undefined.

You can also define the SECRET_KEY in your local_settings.py.

If you do not define a SECRET_KEY by one of these methods, the application will refuse to start and give you an error, that a SECRET_KEY is required.

CHAPTER 8

Indices and tables

- *genindex*
- *modindex*
- *search*