woohoo pDNS GUI

Release 2019.1.0

Andreas Scherrer

Aug 27, 2019

CONTENTS:

1 Installation **1**
- 1.1 Timing 1
- 1.2 Requirements 1
- 1.3 Overview 1
- 1.4 Installing 1

2 woohoo pDNS GUI's ToDo list or whishlist **5**
- 2.1 Some things I am considering to add 5
- 2.2 Some things I am looking for from the community 5

3 Contributing **7**
- 3.1 Managing dependencies 7

4 Support **9**

5 woohoo_pdns_gui **11**
- 5.1 woohoo_pdns_gui package 11

6 Indices and tables **13**

7 What woohoo pDNS GUI is **15**

Python Module Index **17**

Index **19**

CHAPTER

ONE

INSTALLATION

1.1 Timing

A complete installation of woohoo pDNS GUI will require about 30 minutes to complete for an experienced admin.

This does not include installing woohoo pDNS itself.

1.2 Requirements

woohoo pDNS GUI is a Python 3 project, therefore you need Python 3 to run it.

Also, (access to) a pDNS database that oupututs data in Passive DNS - Common Output Format is required. Currently only databases offering API token authentication are supported (not for example the ones _guiusing basic authentication). One option is to use woohoo pDNS.

The Flask webapp is served by Gunicorn. It is strongly suggested to have a reverse proxy (like Nginx, lighttpd, Apache, . . .) in front of it.

1.3 Overview

The installation will consist of the following steps:

1) create a virtual environment (Python 3)
2) install woohoo pDNS GUI and dependencies
3) configure access to the pDNS database
4) set up the configuration in the reverse proxy
5) configure Gunicorn to serve the GUI

1.4 Installing

1.4.1 The virtual environment

Any way of virtualising the Python environment can be used to run woohoo pDNS GUI. For this guide we use Python's integrated `venv` method.

Caution: woohoo pDNS has pinned it's dependencies! This means that the exact version is noted in `requirements.txt` for all dependencies. This might have undesired side effects when installing in a non-empty environment where one of the packages woohoo pDNS depends on is already installed.

So, go ahead and choose a suitable home for your installation of woohoo pDNS GUI. For Linux/*BSD systems, something under `/usr/local` might make sense (e.g. `/usr/local/opt/woohoo-pdns`).

Once you have decided on the location and created a folder for woohoo pDNS GUI, create a new virtual environment like this:

```
$ python -m venv .pdns_gui
```

This will create a folder named `.pdns_gui` in the current directory and this folder will hold your virtual environment of the same name.

Note: on a Mac of mine, creating the virtual environment like this failed with an error like:

```
Error: Command '['/Users/<username>/tmp/.pdns/bin/python', '-Im', 'ensurepip', '--
↪upgrade', '--default-pip']' returned non-zero exit status 1.
```

which can be fixed by following advice found on Stackoverflow:

```
$ python -m venv --without-pip .pdns
$ source .pdns_gui/bin/activate
$ curl https://bootstrap.pypa.io/get-pip.py | python
$ deactivate
$ source .pdns_gui/bin/activate
```

1.4.2 Install woohoo pDNS GUI and dependencies

Go ahead and activate the new environment if not already done (your shell prompt should change):

```
$ source .pdns_gui/bin/activate
```

You should now populate this new virtual environment with woohoo pDNS GUI and the required dependencies:

```
(.pdns_gui)$ pip install woohoo-pdns-gui
```

or install it from source:

```
(.pdns_gui)$ git clone https://gitlab.com/scherand/woohoo-pdns-gui
(.pdns_gui)$ cd woohoo-pdns-gui
(.pdns_gui)$ python setup.py install
(.pdns_gui)$ pip install -r requirements.txt
```

1.4.3 Configure access to the pDNS database

To properly run the woohoo pDNS GUI, you will have to provide a config file (python file, i.e. ending in `.py`) with the following information/format:

```
TIMEZONE="UTC"  # "Europe/Zurich" or "America/New_York"
WOOHOO_APPLICATION_ROOT = ""
SECRET_KEY = "snakeoil"
```

(continues on next page)

(continued from previous page)

```
API_KEY = "MsfPfDqYMQGDc4nVcGTMS8UA"
API_ENTRYPOINT = "http://localhost:5000/api"
```

The values shown here are the default values that will be used if you do *not* provide a config file.

A list of all known timezones can be found via `pytz.all_timezones`.

You can use whatever you like for the `SECRET_KEY`; it is a Flask thing, see *woohoo_pdns_gui.config.DefaultSettings.SECRET_KEY*.

Use the `WOOHOO_APPLICATION_ROOT` variable if you are running woohoo pDNS from a subfolder of the vHost. If you have for example configured your reverse proxy to forward requests for `www.example.com/pdns/` to gunicorn, you should set `WOOHOO_APPLICATION_ROOT` to `/pdns`. This is required that woohoo pDNS GUI can construct the correct hyperlinks.

1.4.4 Set up the configuration in the reverse proxy

Again, the exact steps depend on the reverse proxy software you use and the administrative processes around it. Assuming you have all the required permissions and want to use lighttpd, the configuration should look about as follows:

```
$HTTP["host"] =~ "^pdns.example.com$" {
    $HTTP["url"] =~ "^/" {
        proxy.server = ( "" => ( (
            "host" => "localhost",
            "port" => 5000
        ) ) )
    }
}
```

1.4.5 Configure Gunicorn to serve the GUI

The GUI is served by a Flask application (WSGI application) that lives in `woohoo_pdns_gui.app.py` and is served by Gunicorn. To fire it up, you can use many different ways. For example, a startup script.

Consider using a dedicated user for Gunicorn.

You **must** provide the name of a config file via an environment variable called `WOOHOO_PDNS_GUI_SETTINGS`. This should be the python config file mentioned earlier. If only a filename is specified, the file is expected to be in a folder called `instance` in the directory you are starting flask from. In general, the path to the config file is interpreted as relative to the mentioned `instance` folder.

The following outlines the FreeBSD rc.d script (`/usr/local/etc/rc.d/pdns-api-gunicorn`) I use for this purpose (inspired by a thread in the FreeBSD forums):

```
#! /bin/sh

# PROVIDE: pdns_gui_gunicorn
# REQUIRE: DAEMON
# KEYWORD: shutdown

#
# Add the following lines to /etc/rc.conf to enable the woohoo pDNS GUI:
#
#pdns_gui_gunicorn_enable="YES"
```

(continues on next page)

(continued from previous page)

```
. /etc/rc.subr

name="pdns_gui_gunicorn"
rcvar="${name}_enable"
start_cmd="${name}_start"
stop_cmd="${name}_stop"
pidfile="/var/run/${name}.pid"
procname="daemon:"
gip="localhost"
gport="5000"

pdns_gui_gunicorn_start(){
    chdir /usr/local/opt/woohoo-pdns-gui
    . /root/.virtualenvs/pdns_gui/bin/activate
    LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8 FLASK_ENV=production WOOHOO_PDNS_GUI_SETTINGS=
↪"pdns_gui_conf.py" daemon -r -S -P ${pidfile} -T pdns-gui-gunicorn -u root /root/.
↪virtualenvs/pdns_gui/bin/gunicorn --workers 3 --bind ${gip}:${gport} "woohoo_pdns.
↪gui:create_app()"
}

pdns_gui_gunicorn_stop(){
    if [ -f ${pidfile} ]; then
        echo -n "Stopping services: ${name}"
        # MUST send TERM signal (not e.g. INT) to work properly with '-P' switch
        # check daemon(8) for details
        kill -s TERM $(cat ${pidfile})
        if [ -f ${gsocket} ]; then
            rm -f ${gsocket}
        fi
        echo "."
    else
        echo "It appears ${name} is not running."
    fi
}

load_rc_config ${name}
# this sets the default 'enable' (to no)
: ${pdns_gui_gunicorn_enable:="no"}
run_rc_command "$1"
```

CHAPTER

TWO

WOOHOO PDNS GUI'S TODO LIST OR WHISHLIST

2.1 Some things I am considering to add

- make table sortable (JavaScript)
- support for basic authentication
- filtering by `rrtype`
- searching for "first seen in last 24 hours" or similar

2.2 Some things I am looking for from the community

- Init scripts (especially for Linux)
- Reverse proxy configurations for other webservers than lighttpd
- General tips and tricks to run Python web applications (e.g. logging)

CHAPTER THREE

CONTRIBUTING

To build woohoo pDNS GUI (and the documentation), three additional dependencies exist:

```
pip-tools
nose
sphinx-rtd-theme
```

You can easily install them using the following pip command in your development environment:

```
$ pip install -r dev-requirements.txt
```

To build the documentation after cloning the repository, run the following command in the `woohoo_pdns_gui/docs` directory:

```
$ make html
```

Note: do *not* run:

```
$ sphinx-quickstart
```

To run the (inexisting) tests, issue the following command:

```
$ python setup.py test
```

3.1 Managing dependencies

Following the advice of people with (much) more experience in that field (namely Vincent Driessen and Hynek Schlawack) woohoo pDNS pins its dependencies.

The tool used is pip-tools, for the runtime dependencies in Hash-Checking Mode, and here's how.

3.1.1 Runtime dependencies

Dependencies required to *run* woohoo pDNS GUI are listed in the `install_requires` variable in `setup.py`:

```
setup(
    <snip>
    install_requires = [
        "flask",
        "gunicorn",
        <snap>
```

(continues on next page)

(continued from previous page)

```
    ]
)
```

If you want to add a new (run time) dependency for woohoo pDNS GUI, this is the place to do so.

3.1.2 Build dependencies

Dependencies required to *develop* woohoo pDNS GUI are listed in the `dev-requirements.in` file:

```
pip-tools
...
```

3.1.3 Using `pip-tools` for woohoo pDNS GUI

To *generate a requirements.txt file* (i.e. a `requirements.txt` file that listing the runtime dependencies), run the following command (you have `pip-tools` installed, right?):

```
$ pip-compile --allow-unsafe --generate-hashes
```

This will *overwrite* the current requirements.txt file with the most recent version available on PiPI for every package and will *add new dependencies* also.

To check if there are newer versions of dependencies available in PyPI, use the following command:

```
$ pip-compile --allow-unsafe --generate-hashes --upgrade
```

This will *overwrite* the current requirements.txt file with the most recent version available on PiPI for every package. It will *not* add new dependencies though.

Note: `pip-compile` has a `dry-run` command line switch.

To *generate the ‘‘dev-requirements.txt‘‘ file* (i.e. a file listing the build dependencies), run the following command:

```
$ pip-compile --allow-unsafe --output-file=dev-requirements.txt dev-requirements.in
```

This will *overwrite* the current `dev-requirements.txt` file with the most recent version available on PiPI for every package and will *add new dependencies* also.

To check if there are newer versions of build dependencies available in PyPI, use the following command:

```
$ pip-compile --upgrade --allow-unsafe --output-file=dev-requirements.txt dev-
↪requirements.in
```

This will *overwrite* the current `dev-requirements.txt` file with the most recent version available on PiPI for every package. It will *not* add new dependencies though.

References:

- Pin Your Packages
- Better Package Management
- Python Application Dependency Management in 2018
- pip-tools (GitHub)

CHAPTER FOUR

SUPPORT

If you need to get help with woohoo pDNS GUI feel free to open an issue on Gitlab and I will do my best to help out.

Please understand however that this currently is a private project run in my free time and that I can only spend as much time on it as I can.

Be assured: I will come back to you; just maybe not right now?

CHAPTER

FIVE

WOOHOO_PDNS_GUI

5.1 woohoo_pdns_gui package

5.1.1 Submodules

5.1.2 woohoo_pdns_gui.app module

5.1.3 woohoo_pdns_gui.config module

class `woohoo_pdns_gui.config.`**`DefaultSettings`**

Bases: `object`

The default configuration of the GUI just demonstrates the available options.

`API_ENTRYPOINT = 'http://localhost:5000/api'`

The URL (entry point) to call for queries.

`API_KEY = 'MsfPfDqYMQGDc4nVcGTMS8UA'`

The API key used to access the pDNS database.

`SECRET_KEY = 'snakeoil'`

Flask uses a secret key to encrypt things that sould be tamper proof (for example the Session object).

`TIMEZONE = 'UTC'`

A timezone is required to localise the display of the first seen/last seen timestamps

`WOOHOO_APPLICATION_ROOT = ''`

Set this to the subpath When the pDNS GUI is not running in the root of a vHost.

`__dict__ = mappingproxy({'__module__': 'woohoo_pdns_gui.config', '__doc__': 'The def`

`__module__ = 'woohoo_pdns_gui.config'`

`__weakref__`

list of weak references to the object (if defined)

5.1.4 woohoo_pdns_gui.meta module

class `woohoo_pdns_gui.meta.`**`LookupDict`**(*name=None*)

Bases: `dict`

Dictionary lookup object.

TODO: understand this… https://github.com/kennethreitz/requests/blob/master/requests/structures.py

`__dict__ = mappingproxy({'__module__': 'woohoo_pdns_gui.meta', '__doc__': '\n Dictio`

`__getitem__`(*key*)
x.__getitem__(y) <==> x[y]

`__init__`(*name=None*)
Initialize self. See help(type(self)) for accurate signature.

`__module__ = 'woohoo_pdns_gui.meta'`

`__repr__`()
Return repr(self).

`__weakref__`
list of weak references to the object (if defined)

`get`(*key*, *default=None*)
Return the value for key if key is in the dictionary, else default.

`woohoo_pdns_gui.meta.`**`_init`**()

5.1.5 Module contents

`woohoo_pdns_gui.`**`create_app`**(*test_config=None*)

`woohoo_pdns_gui.`**`no_app`**(*environ*, *start_response*)

INDICES AND TABLES

- genindex
- modindex
- search

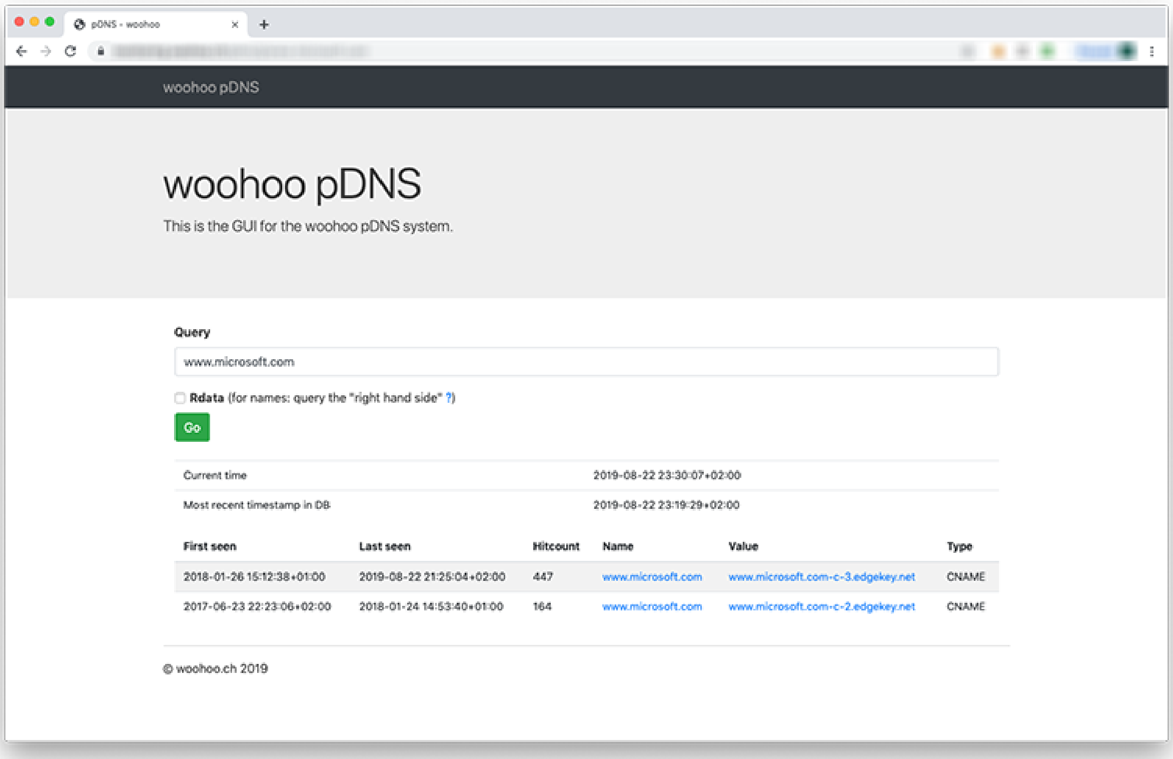
CHAPTER

SEVEN

WHAT WOOHOO PDNS GUI IS

woohoo pDNS GUI is a Python 3 web application (Flask) to display passive DNS data that is provided by a server that complies to the Common Passive DNS Output Format.

This is what it looks like:



woohoo pDNS

This is the GUI for the woohoo pDNS system.

Query

www.microsoft.com

☐ **Rdata** (for names: query the "right hand side" ?)

Go

Current time	2019-08-22 23:30:07+02:00
Most recent timestamp in DB	2019-08-22 23:19:29+02:00

First seen	Last seen	Hitcount	Name	Value	Type
2018-01-26 15:12:38+01:00	2019-08-22 21:25:04+02:00	447	www.microsoft.com	www.microsoft.com-c-3.edgekey.net	CNAME
2017-06-23 22:23:06+02:00	2018-01-24 14:53:40+01:00	164	www.microsoft.com	www.microsoft.com-c-2.edgekey.net	CNAME

© woohoo.ch 2019

There is a Database component that can be used but must be installed separately.

If you want to know in more detail what (a) passive DNS (data(base)) is, the FAQ on the Farsight Security website is a valid resource to read up on the topic.

PYTHON MODULE INDEX

W

woohoo_pdns_gui, 12
woohoo_pdns_gui.config, 11
woohoo_pdns_gui.meta, 11

INDEX

Symbols

__dict__ (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11
__dict__ (*woohoo_pdns_gui.meta.LookupDict attribute*), 11
__getitem__() (*woohoo_pdns_gui.meta.LookupDict method*), 12
__init__() (*woohoo_pdns_gui.meta.LookupDict method*), 12
__module__ (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11
__module__ (*woohoo_pdns_gui.meta.LookupDict attribute*), 12
__repr__() (*woohoo_pdns_gui.meta.LookupDict method*), 12
__weakref__ (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11
__weakref__ (*woohoo_pdns_gui.meta.LookupDict attribute*), 12
_init() (*in module woohoo_pdns_gui.meta*), 12

A

API_ENTRYPOINT (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11
API_KEY (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11

C

create_app() (*in module woohoo_pdns_gui*), 12

D

DefaultSettings (*class in woohoo_pdns_gui.config*), 11

G

get() (*woohoo_pdns_gui.meta.LookupDict method*), 12

L

LookupDict (*class in woohoo_pdns_gui.meta*), 11

N

no_app() (*in module woohoo_pdns_gui*), 12

S

SECRET_KEY (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11

T

TIMEZONE (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11

W

WOOHOO_APPLICATION_ROOT (*woohoo_pdns_gui.config.DefaultSettings attribute*), 11
woohoo_pdns_gui (*module*), 12
woohoo_pdns_gui.config (*module*), 11
woohoo_pdns_gui.meta (*module*), 11