DnsCherry - Dns administration Web UI

Release 0.1.3

Mar 10, 2017

Contents

Dns management web interface.

Dev dnscherry git

PyPI dnscherry package

License MIT

Author Pierre-Francois Carpentier - copyright © 2014

CHAPTER 1

Site Menu

Install

From the sources

Download the latest release from GitHub.

```
$ tar -xf dnscherry*.tar.gz
$ cd dnscherry*
$ python setup.py install
```

From Pypi

```
$ pip install dnscherry
```

or

```
$ easy_install dnscherry
```

Installed files

DnsCherry install directories are:

- **/etc/dnscherry/** (configuration)
- **dist-package** or **site-packages** of your distribution (DnsCherry modules)
- **/usr/share/dnscherry/** (static content (css, js, images...) and templates)

These directories can be changed by exporting the following variables before launching the install command:

```
#optional, default sys.prefix (/usr/ on most Linux)
$ export DATAROOTDIR=/usr/local/
#optional, default /etc/
$ export SYSCONFDIR=/usr/local/etc/
```

Deploy

Launching DnsCherry

DnsCherry can be launched using cherrypy internal webserver:

```
# dnscherryd help
$ dnscherryd -h
Usage: dnscherryd [options]

Options:
  -h, --help            show this help message and exit
  -c CONFIG, --config=CONFIG
                        specify config file
  -d                    run the server as a daemon
  -e ENVIRONMENT, --environment=ENVIRONMENT
                        apply the given config environment
  -f                    start a fastcgi server instead of the default HTTP
                        server
  -s                    start a scgi server instead of the default HTTP server
  -x                    start a cgi server instead of the default HTTP server
  -p PIDFILE, --pidfile=PIDFILE
                        store the process id in the given file
  -P PATH, --Path=PATH  add the given paths to sys.path

# launching dnscherryd
$ dnscherryd -c /etc/dnscherry/dnscherry.ini
```

Dns Configuration

This section presents the configuration of a zone **example.com**, change it with your own zone name.

Creation of the tsig key

To create a tsig key, use the following commands:

```
$ dnssec-keygen -a HMAC-SHA512 -b 512 -n HOST example.com.
# it creates two files:
$ ls K*
Kexample.com.+165+27879.key  Kexample.com.+165+27879.private

# content
$ cat K*.private
Private-key-format: v1.3
Algorithm: 165 (HMAC_SHA512)
Key: oWko9cBK6yUDnhl8R6g0drseVc2t9erYIiHD/
 ↪u9t31iMYR+rbF5Y7IXeVdGCwEDe3fpQVYWvZosUzScZ5VStLA==
```

```
Bits: AAA=
Created: 20140521080238
Publish: 20140521080238
Activate: 20140521080238
```

The important field for us is **Key**, it's this field which will be used in the dns server and DnsCherry.

Bind server TSIG configuration

Configure Bind server to use this key:

```
key example.com. {
    algorithm       hmac-sha512;
    secret "oWko9cBK6yUDnhl8R6g0drseVc2t9erYIiHD/
↪u9t31iMYR+rbF5Y7IXeVdGCwEDe3fpQVYWvZosUzScZ5VStLA==";
};

zone "example.com" {
    type master;
    file "/var/lib/bind/db.example.com";
    allow-update { key "example.com."; };

    // In case you have an 'allow-transfer { "none"; };' in options
    //allow-transfer { <dnscherry ip>; };
};
```

Warning: The user running bind must have writing rights on **/var/lib/bind/db.example.com** and **/var/lib/bind/**.

DnsCherry configuration

Configure the zone in DnsCherry:

```
[dns.zones]

#####################################
# parameters for zone "example.com" #
#####################################
# dns server ip
ip.example.com = '127.0.0.1'
# hmac algorithm
algorithm.example.com = 'hmac-sha512'
# hmac key
key.example.com = 'oWko9cBK6yUDnhl8R6g0drseVc2t9erYIiHD/
↪u9t31iMYR+rbF5Y7IXeVdGCwEDe3fpQVYWvZosUzScZ5VStLA=='
```

You can configure multiple zones by adding the following parameters in the **[dns.zones]** section:

- **ip.<your domain>**
- **algorithm.<your domain>**
- **key.<your domain>**

example:

```
[dns.zones]

ip.example.com = '127.0.0.1'
algorithm.example.com = 'hmac-sha512'
key.example.com = 'oWko9cZosUzScZ5VStLA=='

ip.mydomain.org = '192.168.0.1'
algorithm.mydomain.org = 'hmac-md5'
key.mydomain.org = 'oWko9caaaafdeZosUzScZ5VStLA=='
```

Supported algorithms

DnsCherry supports the following algorithms:

- hmac-md5
- hmac-sha1
- hmac-sha224
- hmac-sha256
- hmac-sha384
- hmac-sha512

Other dns parameters

DnsCherry has other dns parameters which must be provided:

```
# other dns parameters
[dns]
# the default selected zone
default.zone = 'example.com'
# record types to display
type.displayed = 'A, AAAA, CNAME, MX'
# record types available for new records
type.written = 'A, AAAA, CNAME, MX'
# default ttl for new records
default.ttl = '3600'
```

Logs

DnsCherry has two loggers, one for errors and actions (login, del/add, logout...) and one for access logs. Each logger can be configured to log to syslog, file or be disabled.

Warning: you can't set a logger to log both in file and syslog

Syslog configuration:

```
[global]

# logger syslog for access log
log.access_handler = 'syslog'
```

```
# logger syslog for error and dnscherry log
log.error_handler = 'syslog'
```

File configuration:

```
[global]

# logger syslog for access log
log.access_handler = 'file'
# logger syslog for error and dnscherry log
log.error_handler = 'file'
# access log file
log.access_file = '/tmp/dnscherry_access.log'
# error and dnscherry log file
log.error_file = '/tmp/dnscherry_error.log'
```

Disable logs:

```
[global]

# logger syslog for access log
log.access_handler = 'none'
# logger syslog for error and dnscherry log
log.error_handler = 'none'
```

Set log level:

```
[global]

# log level
log.level = 'info'
```

Other DnsCherry parameters

```
[global]

# listing interface
server.socket_host = '127.0.0.1'
# port
server.socket_port = 8080
# number of threads
server.thread_pool = 8
#don't show traceback on error
request.show_tracebacks = False

# session configuration
# activate session
tools.sessions.on = True
# session timeout
tools.sessions.timeout = 10
# file session storage(to use if multiple processes,
# default is in RAM and per process)
#tools.sessions.storage_type = "file"
# session
#tools.sessions.storage_path = "/var/lib/dnscherry/sessions"
```

```
# resources parameters
[resources]
# templates directory
template_dir = '/usr/share/dnscherry/templates/'

# enable cherrypy static handling
# to comment if static content are handled otherwise
[/static]
tools.staticdir.on = True
tools.staticdir.dir = '/usr/share/dnscherry/static/'
```

WebServer

Idealy, DnsCherry must be deployed behind a proper http server like nginx or apache.

The webserver must be configured to act as a reverse (ssl) proxy to a DnsCherry instance listening on localhost (127.0.0.1).

Cherrypy

Cherrypy has an embeded web sever which can be used for testing.

It has some severe limitations:

- no SSL/TLS
- no listening on the standard http port 80

To make DnsCherry listens on every IP:

```
[global]

# listing interface
server.socket_host = '0.0.0.0'
# port
server.socket_port = 8080
```

Nginx

```
server {
    listen 80 default_server;

    server_name $hostname;
    #access_log /var/log/nginx/dnscherry_access_log;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-for $proxy_add_x_forwarded_for;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Proto $remote_addr;
    }
}
```

Apache

```
<VirtualHost *:80>

    <Location />
        ProxyPass http://127.0.0.1:8080/
        ProxyPassReverse http://127.0.0.1:8080/
    </Location>

</VirtualHost>
```

Lighttpd

```
server.modules += ("mod_proxy")

$HTTP["host"] == "dns.kakwa.fr" {
    proxy.server  = ( "" =>
        (( "host" => "127.0.0.1", "port" => 8080 ))
    )
}
```

DnsCherry configuration file

```
# global parameters
[global]

# listing interface
server.socket_host = '127.0.0.1'
# port
server.socket_port = 8080
# number of threads
server.thread_pool = 8
#don't show traceback on error
request.show_tracebacks = False

# log configuration
# /!\ you can't have multiple log handlers
#####################################
#   configuration to log in files   #
#####################################
## logger 'file' for access log
#log.access_handler = 'file'
## logger syslog for error and dnscherry log
#log.error_handler = 'file'
## access log file
#log.access_file = '/tmp/dnscherry_access.log'
## error and dnscherry log file
#log.error_file = '/tmp/dnscherry_error.log'

#####################################
#  configuration to log in syslog   #
#####################################
# logger syslog for access log
#log.access_handler = 'syslog'
```

```
## logger syslog for error and dnscherry log
log.error_handler = 'syslog'

#####################################
#  configuration to not log at all  #
#####################################
# logger none for access log
log.access_handler = 'none'
# logger none for error and dnscherry log
#log.error_handler = 'none'

# log level
log.level = 'info'

# session configuration
# activate session
tools.sessions.on = True
# session timeout
tools.sessions.timeout = 10
# file session storage(to use if multiple processes,
# default is in RAM and per process)
#tools.sessions.storage_type = "file"
# session
#tools.sessions.storage_path = "/var/lib/dnscherry/sessions"

# zones to manage
[dns.zones]

# Supported algorithms
# * hmac-md5
# * hmac-sha1
# * hmac-sha224
# * hmac-sha256
# * hmac-sha384
# * hmac-sha512

#####################################
# parameters for zone "example.com" #
#####################################
# dns server ip
ip.example.com = '127.0.0.1'
# hmac algorithm
algorithm.example.com = 'hmac-md5'
# hmac key
key.example.com = 'ujeGPu0NCU1TO9fQKiiuXg=='

#####################################
# parameters for zone "example2.org" #
#####################################
# dns server ip
ip.example2.org = '127.0.0.1'
# hmac algorithm
algorithm.example2.org = 'hmac-sha512'
# hmac key
key.example2.org = 'ujeGPu0NCU1TO9fQKiiuXg=='

# other dns parameters
[dns]
```

```
# the default selected zone
default.zone = 'example.com'
# record types to display
type.displayed = 'A, AAAA, CNAME, MX'
# record types available for new records
type.written = 'A, AAAA, CNAME, MX'
# default ttl for new records
default.ttl = '3600'

# resources parameters
[resources]
# templates directory
template_dir = '/usr/share/dnscherry/templates/'

# authentification parameters
[auth]

#######################################
# parameters for auth module 'none'  #
#######################################
# this module disable authentification
# (if you use other means of authentification)

# auth module to load
auth.module = 'dnscherry.auth.modNone'
# optional http header handling.
# useful if username is transmitted by header
# (permits nominative logs).
# if activated, this header presence
# in each request is mandatory.
#auth.none.user_header_name = 'AUTH_USER'

#######################################
## parameters for auth module 'ldap'  #
#######################################
# This module is used for ldap authentification

## name of the auth module
#auth.module = 'dnscherry.auth.modLdap'
## base dn where to search user
#auth.ldap.userdn = 'ou=People,dc=example,dc=org'
## ldap login filter
#auth.ldap.user.filter.tmpl = '(uid=%(login)s)'
## base dn for group
## (if empty, all user in userdn can access dnscherry)
#auth.ldap.groupdn = 'cn=itpeople,ou=Groups,dc=example,dc=org'
## ldap group filter
#auth.ldap.group.filter.tmpl = '(member=%(userdn)s)'
## bind dn
#auth.ldap.binddn = 'cn=dnscherry,dc=example,dc=org'
## bind password
#auth.ldap.bindpassword = 'password'
## ldap uri
#auth.ldap.uri = 'ldaps://ldap.dnscherry.org'
## ldap CA file (use for ssl/tls)
#auth.ldap.ca = '/etc/dnscherry/TEST-cacert.pem'
## enable starttls (default off)
##auth.ldap.starttls = 'on'
```

```
## check certificat (default on)
##auth.ldap.checkcert = 'off'
## ldap timeout in seconds (default 1 second)
##auth.ldap.timeout = 2

########################################
#  parameters for auth module 'htpasswd' #
########################################
# This module is used for htpasswd file
# user database
#
# name of the auth module
#auth.module = 'dnscherry.auth.modHtpasswd'
# path to htpasswd file
#auth.htpasswd.file = '/etc/dnscherry/users.db'

# enable cherrypy static handling
# to comment if static content are handled otherwise
[/static]
tools.staticdir.on = True
tools.staticdir.dir = '/usr/share/dnscherry/static/'
```

Init Script

Sample init script for Debian:

```
#! /bin/sh

### BEGIN INIT INFO
# Provides:        dnscherryd
# Required-Start:    $remote_fs $network $syslog
# Required-Stop:    $remote_fs $network $syslog
# Default-Start:    2 3 4 5
# Default-Stop:
# Short-Description:    DnsCherry
### END INIT INFO

PIDFILE=/var/run/dnscherryd/dnscherryd.pid
CONF=/etc/dnscherry/dnscherry.ini
USER=www-data
GROUP=www-data
BIN=/usr/local/bin/dnscherryd
OPTS="-d -c $CONF -p $PIDFILE"

. /lib/lsb/init-functions

if [ -f /etc/default/dnscherryd ]; then
    . /etc/default/dnscherryd
fi

start_dnscherryd(){
    log_daemon_msg "Starting DnsCherryd" "dnscherryd" || true
    pidofproc -p $PIDFILE $BIN >/dev/null
    status="$?"
    if [ $status -eq 0 ]
    then
```

```
        log_end_msg 1
        log_failure_msg \
        "dnscherryd already started"
        return 1
    fi
    mkdir -p `dirname $PIDFILE` -m 750
    chown $USER:$GROUP `dirname $PIDFILE`
    if start-stop-daemon -c $USER:$GROUP --start \
        --quiet --pidfile $PIDFILE \
        --oknodo --exec $BIN -- $OPTS
    then
        log_end_msg 0 || true
        return 0
    else
        log_end_msg 1 || true
        return 1
    fi

}

stop_dnscherryd(){
    log_daemon_msg "Stopping DnsCherryd" "dnscherryd" || true
    if start-stop-daemon --stop --quiet \
        --pidfile $PIDFILE
    then
        log_end_msg 0 || true
        return 0
    else
        log_end_msg 1 || true
        return 1
    fi
}

case "$1" in
  start)
    start_dnscherryd
    exit $?
    ;;
  stop)
    stop_dnscherryd
    exit $?
    ;;
  restart)
    stop_dnscherryd
    while pidofproc -p $PIDFILE $BIN >/dev/null
    do
        sleep 0.5
    done
    start_dnscherryd
    exit $?
    ;;
  status)
    status_of_proc -p $PIDFILE $BIN "DnsCherryd" \
        && exit 0 || exit $?
    ;;
  *)
    log_action_msg \
    "Usage: /etc/init.d/dnscherryd {start|stop|restart|status}" \
```

```
    || true
    exit 1
esac

exit 0
```

This init script is available in **goodies/init-debian**.

Auth Modules

Configuration

modNone

This module is used if no authentication is needed, for example, with a Single Sign On in front of DnsCherry.

However, in order to trace who made which action, the user name can be provided in an http header, in that case, having the http header in each request is mandatory.

Configuration:

```
[auth]

# auth module to load
auth.module = 'dnscherry.auth.modNone'
# optional http header handling.
#auth.none.user_header_name = 'AUTH_USER'
```

modHtpasswd

This module uses an htpasswd file as a users database.

> **Warning:** This module requires installing **python-passlib**.

Configuration:

```
[auth]

# name of the auth module
auth.module = 'dnscherry.auth.modHtpasswd'
# path to htpasswd file
auth.htpasswd.file = '/etc/dnscherry/users.db'
```

modLdap

This module authenticates users against an ldap server.

> **Warning:** This module requires installing **python-ldap**.

Warning: As **python-ldap** is **not compatible with python 3.x**, DnsCherry must be launched with **python 2.7** if this module is used.

Configuration:

```
[auth]

# name of the auth module
auth.module = 'dnscherry.auth.modLdap'
# base dn where to search user
auth.ldap.userdn = 'ou=People,dc=example,dc=org'
# ldap login filter
auth.ldap.user.filter.tmpl = '(uid=%(login)s)'
# base dn for group (optional)
# (if empty, all user in userdn can access dnscherry)
auth.ldap.groupdn = 'cn=itpeople,ou=Groups,dc=example,dc=org'
# ldap group filter (optional, except if auth.ldap.groupdn is defined)
auth.ldap.group.filter.tmpl = '(member=%(userdn)s)'
# bind dn
auth.ldap.binddn = 'cn=dnscherry,dc=example,dc=org'
# bind password
auth.ldap.bindpassword = 'password'
# ldap uri
auth.ldap.uri = 'ldap://ldap.dnscherry.org'
# ldap CA file (optional, used for ssl/tls)
auth.ldap.ca = '/etc/dnscherry/TEST-cacert.pem'
# enable starttls (optional, default off, don't use it with ldaps)
auth.ldap.starttls = 'on'
# check certificat (optional, default on)
auth.ldap.checkcert = 'on'
# ldap timeout in seconds (default 1 second)
auth.ldap.timeout = 2
```

Implementing your own auth modules

Implementing your own modules should be fairly easy, just create a python module with a class inheriting from dnscherry.auth.Auth, and implement the **__init__** and **check_credentials** methods:

```
import cherrypy
import dnscherry.auth
import logging

class Auth(dnscherry.auth.Auth):

    def __init__(self, config, logger=None):
        """ module initialization
        initialize the auth module
        the 'auth' section of the ini file is passed by 'config'
        @hash config: the 'auth' section of the ini file
        @logger logger: the dnscherry error logger
        """
        self.logger = logger

        # set to True if you want the logout button to be displayed
        # set to False to hide it
```

```
        self.logout_button = False

        # get param1, with default value 'hello'
        self.param1 = self._get_param('auth.mymod.param1', 'hello')

        # get param2, with no default value
        # if not provided in the 'auth' section, DnsCherry will emit
        # a log telling that the parameter is missing and exit(1)
        self.param2 = self._get_param('auth.mymod.param2')

        # emit a custom log
        self._logger(
             logging.DEBUG,
             'my module is initialized'
        )

    def check_credentials(self, username, password):
        """ Check credential function (called on login)
        @str username: the login to check
        @str password: the password to check
        @rtype: bool (True if authentificated, False otherwise)
        """

        # simple module checking only one user/password
        return username == 'george' and password == 'password'
```

Changelog

Version 0.1.3

- fix configuration install setup.py if –root is specified

Version 0.1.2

- fix html templates

Version 0.1.1

- fix configuration overwrite in setup.py
- fix syslog message error

Version 0.1.0

- fix ldap auth module if using tls
- add notification mechanizism for add/delete records
- remove ugly status pages
- adding some unit tests

Version 0.0.1

- first release

CHAPTER 2

Presentation

DnsCherry is cherrypy application to manage dns zones.

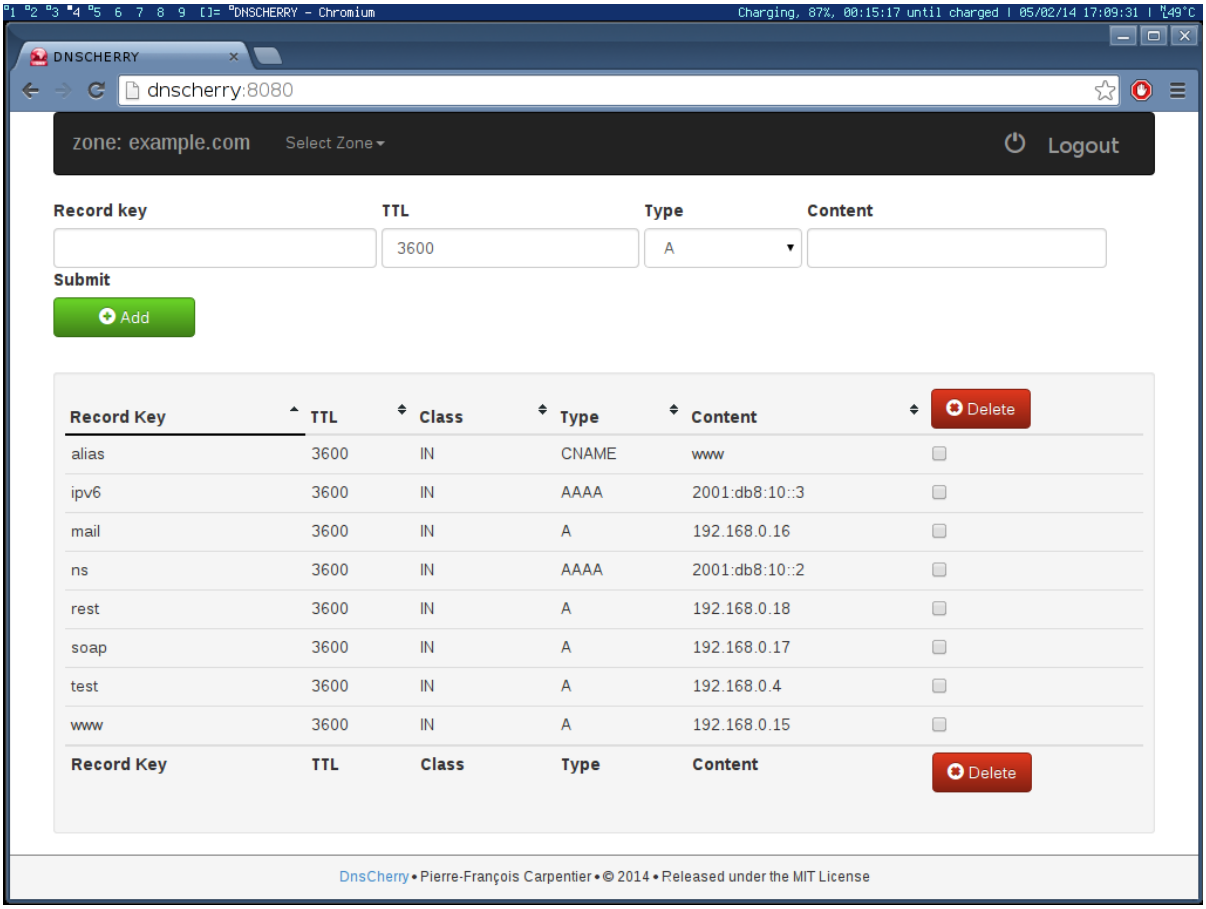
It's main functionalities are:

- add or delete records in one or several zones (using Tsig HMAC)
- access control with htpasswd files, ldap, http header (use it behind a SSO)
- modular auth mechanisms, custom authentication modules can easily be implemented
- traces for each action (add/delete record, connexions...) through logs (syslog or files)

DnsCherry aims to be a simple to deploy and to use Dns web interface.

CHAPTER 3

Screenshot



CHAPTER 4

Discussion / Help / Updates

- IRC: Freenode `#dnscherry` channel
- Bugtracker: Github